Table des matières

Chapitre 1

1.1 Introduction

La communication à distance (téléphone, télévision, satellite, etc.) , entre Machines et usagers nécessite des lignes de transmission acheminant l'information sans la modifier. Les lignes utilisées sont en général loin d'être parfaites . Pour cela, l'information devra être codée d'une manière spéciale permettant de déceler les erreurs, ou ce qui est encore mieux de les corriger automatiquement. On a été amené à concevoir des codes détecteurs et correcteurs d'erreurs.

Alors le problème majeur des dispositifs de télécommunication en plus de l'augmentation du débit de transmission qui peut être réglé par les méthodes de compression, est d'une part, les erreurs introduites par les supports de transmissions c.à.d. l'information reçue est erronée. On a besoin de protéger cette information. Et d'autre part, le besoin croissant de sécuriser les données dans les domaines informatique et les télécommunications. Surtout maintenant avec la venue des réseaux téléinformatiques, l'emploi des liaisons satellites et l'utilisation de l'Internet la situation a radicalement changé, dans la mesure où un même message transite par plusieurs machines avant d'atteindre son destinataire. A chaque étape, il peut être copié, perdu ou altéré. Le cryptage est donc nécessaire pour que les données soient non intelligibles sauf à l'auditoire voulu.
En 1976, avec l'invention du premier crypto système à clef publique par Diffie et Hellman . L'idée nouvelle était de faire reposer la sécurité d'un système non pas sur la connaissance d'une clef (partagée secrètement par les utilisateurs), mais sur la difficulté d'inverser une fonction à sens unique .

Une fonction à sens unique est simplement une fonction calculatoirement difficile à inverser.

Une trappe est un algorithme secret rendant facile cette inversion. Ainsi la trappe n'est connue que d'une personne, seule à pouvoir déchiffrer les messages créés en utilisant la fonction à sens unique qui est elle publique.

Dès 1978, McEliece a imaginé le premier et le plus célèbre des crypto systèmes à clef publique utilisant des codes correcteurs d'erreurs. Comme nous allons le voir dans ce

sujet, la théorie des codes contient elle aussi de multiples problèmes bien structurés et difficiles à résoudre, plus ou moins biens adaptés pour une utilisation en cryptographie.

Dans ce travail on va essayer d'introduire un crypto système à clef publique utilisant des codes correcteurs d'erreurs. Le système étudié est le crypto système de McEliece utilisant le code de Goppa.

1.2 Présentation

Le présent travail présenté dans ce manuscrit sera constitué de cinq parties précédées d'une introduction :
La première partie consiste d'une part à faire un rappel sur des outils permettant l'évaluation d'un code correcteur d'erreur, et d'autre part à élucider quelques exemples des codes correcteur d'erreur.Quant à la deuxième partie sera consacrée pour les codes de Goppa. Dans la partie suivante on va introduire la cryptographie à clef publique. La quatrième partie va mettre le point sur l'utilisation des codes correcteurs d'erreur en cryptographie à clef publique, on va voir comme des exemples le cryptosystème de McElice et La variante de Niederreiter. Une étude des fonctions de hachage cryptographiques sera l'objet de la cinquième partie. En terminant par une conclusion.

Chapitre 2

Codes correcteurs d'erreurs

En 1946, Richard Hamming travaille sur un modèle de calculateur à carte perforée de faible fiabilité. Si, durant la semaine, des ingénieurs peuvent corriger les erreurs, les périodes chômées comme la fin de semaine voient les machines s'arrêter invariablement sur des bogues. La frustration de Hamming le conduit à inventer le premier code correcteur véritablement efficace.

Cette période correspond à la naissance de la théorie de l'information. Claude Shannon formalise cette théorie comme une branche des mathématiques. Hamming développe les prémisses de la théorie des codes et décrit sa solution comme un exemple.

Les codes correcteurs d'erreurs sont des outils visant à améliorer la fiabilité des transmissions sur un canal bruité. La méthode qu'ils utilisent consiste à envoyer sur le canal plus de données que la quantité d'information à transmettre. Une redondance est ainsi introduite. Si cette redondance est structurée de manière exploitable, il sera alors possible de corriger d'éventuelles erreurs introduites par le canal. On peut alors, malgré le bruit, retrouver l'intégralité des informations transmises au départ. Une grande famille de codes correcteurs d'erreurs est constituée des codes par blocs. Pour ces codes l'information est d'abord coupée en blocs de taille constante et chaque bloc est transmis indépendamment des autres, avec une redondance qui lui est propre. La plus grande sous-famille de ces codes rassemble ce que l'on appelle les codes linéaires.

La construction d'un mot de code comportant n bits est effectuée à partir de k bits du message source k-uplet binaires $U = (u_1, u_2, u_3, ...u_k)$, appelé généralement message d'information, et de r bits de redondance. La méthode de codage la plus simple consiste à laisser inchangés les k bits d'information et à les reporter tels quels dans le mot de code en ajoutant les r bits de redondance $[a_1, a_2, ..., a_r]$. Les codes de ce type sont dits systématiques, un codeur qui transforme chaque message U indépendamment en n-uplet, le vecteur ligne V^T appelé mot code autrement dit il y a 2^k messages différents :

$$V^T = [v_1 v_2 ... v_n] = [u_1, u_2, u_3, ...u_k a_1, a_2, ..., a_r]$$

Les bits de redondance $[a_1, a_2, ..., a_r]$. Sont généralement appelés bits de contrôle. Lorsque ces derniers sont calculés uniquement à partir des bits d'information du bloc auquel ils appartiennent, le code est appelé code de bloc (n, k), c'est à dire que les n symboles des mots codes sortant dépendent seulement des k bits en entrée correspondants, on dit alors que le codeur est sans mémoire et peut être implémenté par un circuit logique combinatoire.
Lorsque les bits de contrôle sont calculés à partir des bits d'information appartenant à plusieurs blocs, le code est dit convolutionnel ou récurrent.
Le rapport R=k/n est appelé rendement.

On caractérise aussi les codes par leur capacité de correction d'erreurs. En général on a deux types de catégories :
Ceux qui luttent bien contre les erreurs isolées tels que les codes de Hamming,BCH, GOLAY, et Reed Muller, etc.
Ceux qui sont bien adaptés aux coupures (paquet d'erreurs) comme les codes de FIRE, les codes de Reed Solomon, etc.

2.1 Définitions - Encodage

Définition 2.1.1. Soit F un alphabet. L'ensemble F^n peut-être muni d'une structure d'espace métrique pour la distance, appelée distance de Hamming, définie par :

$$\forall (x, y) \in (F^n)^2, d(x, y) := |\{i, 1 \leq i \leq n \quad et \quad x_i \neq y_i\}|$$

Remarque 2.1.1. La distance de Hamming sur F^n est bien une distance sur F^n. En effet, on a pour tous a, b, c dans E^n :

$$\begin{array}{ccc} d(a,b) = 0 & \Longleftrightarrow & a = b \\ d(a,b) & = & d(b,a) \\ d(a,c) & \leq & d(a,b) + d(b,c) \end{array}$$

Définition 2.1.2. Soient F un alphabet fini de cardinal q et n un entier naturel non nul. Un code C de longueur n sur l'alphabet fini F est un sous-ensemble de F^n. Les éléments de C sont les mots de code. Si C est un code sur un alphabet fini de cardinal q, on dira que C est un code q-aire (si q = 2, on dit code binaire). Au code C sont associés trois paramètres importants (n,M,d) :
- n est la longueur de C.
- $M := |C|$ est le nombre de mots du code C.
- d est la distance minimum de C, elle est définie par

$$d = d_C := min\{d(x, y)/(x, y) \in C^2 \quad et \quad x \neq y\}$$

On dira alors que C est un $(n, M, d)_F-$code.

On définira la distance de $x \in F^n$ au code C par

$$d(x, C) := min\{d(x, y)/y \in C\}$$

On définit aussi deux paramètres relatifs pour le code C :
- Le taux de transmission de C est : $R := log_q(M)/n$.
- La distance relative de C est :$\delta := d/n$.

Remarque 2.1.2.

$$0 \leq R \leq 1 \quad et \quad 0 \leq \delta \leq 1$$

Exemple 2.1.1. Dans $\mathbb{F}_2^3$, on considère le cube d'arète 1 dont les sommets représentent tous les éléments de $\mathbb{F}_2^3$. Si $C = \mathbb{F}_2^3$, la distance minimum est 1. Si un mot reçu y est affecté d'une erreur, on a $y \in C$ et on ne pourra pas detecter cette erreur. Considérons maintenant le code :

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

La distance minimum de ce code est 2 : minimum des longueurs des chemins sur le cube (suivant les arètes) entre deux mots de code. C est une $(3, 4, 2)_{\mathbb{F}_2}$-code. Si un mot reçu y est affecté d'une erreur, on a $y \notin C$ et on pourra détecter cette erreur mais pas la corriger.

Définition 2.1.3. Soit C un $(n, M = q^k, d)_F$-code, $k < n$. F^k est l'espace des messages. Un encodage associé à C est une application injective E :

$$F^k \longrightarrow F^n$$

$$m \longrightarrow c,$$

telle que $E(F^k) = C$. Un tel code C permet de coder q^k messages différents de longueur k.

Exemple 2.1.2. : Historiquement, le premier code correcteur a été donné par Hamming. Si t est un entier, $t \geq 1$, C_t est un code de longueur $n = (t + 1)^2$ et de dimension $k = t^2$ sur l'alphabet $F = \mathbb{F}_2$, la redondance est donc égale à 2t +1. En fait C_t est un sous-espace vectoriel de F_2^n muni de sa structure de F_2-espace vectoriel. Par exemple, pour t =2, l'application d'encodage est l'application F_2-linéaire

$$F_2^4 \longrightarrow F_2^9$$

$$m \longrightarrow c,$$

Soit A la matrice de E dans les bases canoniques de F_2^4 et F_2^9 : on a donc

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \\ c_9 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{pmatrix} = \begin{pmatrix} m_1 \\ m_2 \\ m_1 + m_2 \\ m_3 \\ m_4 \\ m_3 + m_4 \\ m_1 + m_3 \\ m_2 + m_4 \\ m_1 + m_2 + m_3 + m_4 \end{pmatrix}$$

On a donc $c_i = m_i$ pour i=1,2, $c_i = m_{i-1}$ pour i=4,5 et les autres c_j s'obtiennent en faisant l'addition modulo 2 (dans chaque ligne et colonne) suivante :

c_1	c_2	c_3
c_4	c_5	c_6
c_7	c_8	c_9

On voit donc que ce code détecte et corrige 1 erreur : si on reçoit le 9-uplet (1,1,0,0,0,1,1,0,1),l'addition

1	1	0
0	0	1
1	0	1

permet de constater que les sommes correspondant à la deuxième colonne et à la deuxième ligne sont fausses donc il s'est vraisemblablement produit une erreur à la 5ème composante et le vecteur émis était (1,1,0,0,1,1,1,0,1).

2.2 Décodage - Utilité de la redondance

La situation est la suivante : le message m de longueur k a été codé par le mot de code $c \in C$ grâce à l'application E (Définition 2.1.3) . On a transmis c par le canal et le récepteur a reçu un $y \in F^n$. Comment retrouver le mot de code émis c, sachant qu'ensuite on retrouvera le message (si tout se passe bien !) par l'application E^{-1} ?

Définition 2.2.1. Soit C un $(n, M, d)_F-$code. Un décodage associé à C est une application surjective D :

$$F^n \longrightarrow C \subset F^n$$

$$y \longrightarrow D(y)$$

telle que

$$D(x) = x \Longleftrightarrow x \in C.$$

On adopte généralement le principe d'un décodage par maximum de vraisemblance, c'est à dire que pour tout mot reçu $y \in F^n$, D(y) sera égal au mot (ou un des mots) de C le plus "proche" de y au sens de la distance de Hamming : on obtient alors le message (vraisemblablement) émis $m' = E^{-1}(D(y))$.
Schématiquement, si $\mathcal{M}$ désigne l'espace des messages, $|\mathcal{M}| = |C| < |F^n|$, on a (si le décodage est correct, $c' = c$),

$$\begin{array}{ccccccc} \mathcal{M} & \overset{E}{\longmapsto} & F^n & \overset{transmission}{\longmapsto} & F^n & \overset{D}{\longmapsto} & C \overset{E^{-1}}{\longmapsto} \mathcal{M} \\ m & \longmapsto & c & & y & \longmapsto & C' \longmapsto m' \end{array}$$

Proposition 2.2.1. *Soit $C \subset F^n$ un code de distance minimum d. Alors C détecte $[d/2]$-erreurs et corrige au plus $t := [(d-1)/2]$-erreurs.*

Preuve :
Correction d'erreurs : dans F^n, on considère les boules au sens de la distance de Hamming,B(c,t), de centre les mots de C et de rayon $t := [(d-1)/2]$
$B(c,t) := \{y \in F^n; d(c,y) \leq t\}$. Puisque d est la distance minimum du code C, ces boules sont disjointes : en effet soient $c \neq c'$ deux mots de code et supposons que $y \in B(c,t) \cap B(c',t)$ alors, d'après l'inégalité triangulaire,

$$d(c,c') \leq d(c,y) + d(y,c') \leq 2t < d$$

ce qui est impossible puisque la distance minimum entre deux mots de code est d.
-Décodage correct : si $y \in F^n$ est un mot reçu affecté d'au plus t-erreurs, il appartient à une unique boule $\mathcal{B}(c,t)$, avec $c \in C$, et on décode y par c.
- Détection d'erreurs : on peut détecter qu'il y a eu des erreurs si le message reçu n'est pas un élément de C. Dans F^n, on considère les boules au sens de la distance de Hamming, de centre les mots de C et de rayon $t_1 := [d/2]$. Supposons que d soit pair puisque sinon $t_1 := [(d-1)/2]$ et on est dans le cas précédent. Soit $y \in F^n$ un mot reçu affecté de t_1-erreurs. Alors y est situé sur le bord d'une seule boule, auquel cas on connait le mot émis ou sur le bord de plusieurs boules, auquel cas on ne sait pas décoder mais on peut dire qu'il y a eu plus de t-erreurs.
-Problème de décodage : si le nombre d'erreurs affectant un mot reçu y est > t :
- soit il existe $c \in C$ tel que $y \in \mathcal{B}(c,t)$; on décodera y par c mais on commettra ainsi un décodage incorrect.
- soit y n'appartient à aucune des boules $\mathcal{B}(c,t)$; dans ce cas on détecte qu'il y a eu plus de t-erreurs et on ne peut pas connaître de façon certaine le mot émis (sauf dans certains cas..).

Définition 2.2.2. Soit C un $(n, M, d)_F$-code. La capacité de correction de C est l'entier $t := [(d-1)/2]$.
L'intérêt des codes correcteurs vient du théorème de Shannon qui prouve l'existence de "bons" codes mais qui n'est pas effectif. Pour un canal de transmission donné, on note $\gamma \in]0, 1/2[$ la probabilité (supposée fixe) qu'un symbole transmis soit entaché d'erreur à la réception et on définit l'entropie $c(\gamma)$ du canal qui, pour $F = \mathbb{F}_2$, est égale à

$$c(\gamma) = 1 + \gamma log_2 \gamma + (1-\gamma) log_2 (1-\gamma).$$

Théorème 2.2.3. *(Shannon - 1948)*
Pour tout $\epsilon > 0$ et pour tout $R < c(\gamma)$, il existe un code C dont le taux de transmission est égal à R et pour lequel la probabilité de décodage incorrect est $< \epsilon$.

2.3 Codes parfaits

Remarque 2.3.1. - Si C est un $(n, k, d)_F$-code, on peut corriger toute configuration d'au plus $t := [(d-1)/2]$-erreurs. Le prix à payer est l'envoi de n symboles pour k symboles d'information. Plus on désire corriger d'erreurs, plus on doit augmenter la redondance pour obtenir une distance minimum convenable. Si n et F sont fixés, quand M croit, la distance minimum d décroit (plus il y a de mots de code dans F^n, plus ils sont proches). Un bon code est un code tel que M et d soient grands : ces exigences sont antagonistes. Soit $C \subset F_n$ un code q-aire de distance minimum d, contenant M mots. On ne suppose pas forcément que $M = q^k$, où $k < n$, mais seulement $M \leq q^n$. Pour n fixé, si on veut augmenter le nombre de mots, la distance minimum va diminuer. Posons

$$A_q(n, d) = max\{M, \exists \quad un \quad code \quad q-aire \quad de \quad paramtres \quad (n, M, d)\}.$$

Si $C = F^n$ on a $d = 1$ et $M = q^n$, donc $A_q(n, 1) = q^n$
Un tel code n'a aucun intérêt pour la correction d'erreurs.
Même pour de petites valeurs de q,n,d, la valeur exacte de $A_q(n, d)$ n'est pas connue.

Exemple 2.3.1. Considérons le code de l'exemple 2.1.1. C'est un code $(3, 4, 2)_2$ donc $A_2(3, 2) \geq 4$. On peut montrer que $A_2(3, 2) = 4$.

Définition 2.3.1. Un code C de distance minimum d est dit parfait si d est impair, $d = 2t + 1$ et si les boules de Hamming de rayon égal à la capacité de correction t du code et de centre les mots du code,$B(c, t) := \{y \in F^n, d(c, y) \leq t\}$, recouvrent F^n : ces boules forment alors une partition de F^n.

C'est la situation idéale pour décoder : en effet tout $y \in F^n$ appartient à une boule de Hamming B(c,t) et une seule donc on peut décoder tout message reçu affecté d'au plus t-erreurs.

Exemple 2.3.2. Dans les cas suivants on obtient des codes parfaits triviaux :
-$M = 1$.
-$F = \mathbb{F}_2$, n impair, $M = 2$, $C = \{(0, ..., 0), (1, ..., 1)\}$, alors $d = n$.
Il y a très peu de codes parfaits non triviaux. Les seuls codes linéaires parfaits sont le code de Hamming binaire $\mathcal{H}_r$ et deux codes de Golay. Les codes parfaits sont utilisés dans la compression de données.

Proposition 2.3.1. *Soit F un alphabet de cardinal q.*
- Pour tout $x \in F^n$, $|B(x, r)| = \sum_{i=0}^{r} C_n^i (q-1)^i$.
-Borne d'empilement des sphères -Soit $d = 2t + 1$ un nombre impair. Alors :

$$A_q(n, d) \sum_{i=0}^{t} C_n^i (q-1)^i \leq q^n$$

Preuve :
-$y \in \mathfrak{B}(x, r) \iff d(x, y) \leq r$, c'est à dire que y a au plus r coordonnées différentes de celles de x,d'où le résultat.
- Supposons qu'il existe un code C de paramètres $(n, M, d)_q$. La capacité de correction de C est t,donc les boules de Hamming de centre les mots du code, $\mathfrak{B}(c, t)$ sont disjointes. On a donc

$$\cup_{c \in C} \mathfrak{B}(c, t) \subset F^n$$

et en prenant les cardinaux de ces ensembles, on obtient

$$M \sum_{i=0}^{t} C_n^i (q-1)^i \leq q^n$$

soit encore

$$A_q(n, d) \sum_{i=0}^{t} C_n^i (q-1)^i \leq q^n$$

Corollaire 2.3.2. *Un code C de paramètres $(n, M, d)_q$ est parfait si et seulement si d est impair et*

$$M \sum_{i=0}^{t} C_n^i (q-1)^i = q^n$$

2.4 Codes linéaires

Dans le reste de ce chapitre, on ne considèrera que des codes linéaires, c'est à dire des sous-espaces vectoriels de F^n, où $F = \mathbb{F}_q$ est un corps fini. Rappellons que $\mathbb{F}_q^n$ est muni de sa structure naturelle de $\mathbb{F}_q$-espace vectoriel produit. De plus, on se fixe une

fois pour toutes une base de F_q^n : par exemple la base canonique. Ceci veut dire que tout élément x de F_q^n (ou de tout sous-F_q-espace vectoriel de F_q^n) s'écrira $x = (x_1, x_2, ..., x_n)$ où les $x_i \in F_q$ sont les coordonnées de x sur la base canonique de F_q^n. Si $y = (y_1, y_2, ..., y_n)$ est un autre élément, la somme x+y est $x + y = (x_1 + y_1, x_2 + y_2, ..., x_n + y_n)$ et si $a \in F_q$ on a

$$ax = (ax_1, ax_2, ..., ax_n).$$

Définition 2.4.1. Un code linéaire q-aire C est un sous-espace vectoriel de F_q^n. On dit que C est un $[n, k, d]_q$-code avec
-n, longueur de C,
- $k = dim_{\mathbb{F}_q} C$, dimension de C : on a donc $M = |C| = q^k$,
- d, distance minimum de C.

Remarque 2.4.1. .-Un code linéaire $[n, k, d]_q$ permet de coder au plus q^k messages différents et de longueur k. Comme l'alphabet est sur un corps, on peut définir la distance minimum d'un code linéaire autrement.

Définition 2.4.2. Soient $x \in F_q^n$ et d la distance de Hamming sur F_q^n. Le support de x est

$$supp(x) := \{i, 1 \leq i \leq n \quad tel \quad que \quad x_i \neq 0\},$$

le poids de $x \in F_q^n$ est défini par $wt(x) := d(x, 0) = |supp(x)|$.

Lemme 1. Soit C un $[n, k, d]_q$-code. Alors la distance minimum de C est telle que

$$d = min\{wt(x), x \in C \setminus \{0\}\}$$

Preuve : Il suffit de remarquer que $0 \in C$ et que $d(x, y) = wt(x - y)$.

2.4.1 Encodage d'un code linéaire - Matrice génératrice

Définition 2.4.3. Une matrice génératrice d'un $[n, k, d]_q$-code C, notée G, est une matrice $G \in \mathcal{M}_{k,n}(\mathbb{F}_q)$ de rang maximum $(rg(G) = k)$ dont les vecteurs lignes forment une base du code C (exprimés dans la base fixée de F_q^n).

Noter qu'il existe plusieurs matrices génératrices pour un même code (autant que de bases différentes). Traditionnellement les codeurs notent les vecteurs de F_q^k ou F_q^n comme des vecteurs lignes. Soit C un $[n, k, d]_q$-code et G une matrice génératrice de C. On identifie l'espace des messages à F_q^k.
L'encodage de C associé à G est l'application linéaire

$$E : F_q^k \longrightarrow C \subset F_q^n$$

dont la matrice est la transposée de G. Tout message $u = (u_1, ..., u_k) \in F_q^k$ est codé par $E(u) = (u_1, ..., u_k)G$.
L'application $E : F_q^k \longrightarrow C$ est bien bijective parce que sa matrice est de rang maximum k.

Proposition 2.4.1. *Soit C un $[n, k, d]_q$-code et G une matrice génératrice de C. Alors* $C = \{(u_1, ..., u_k)G; (u_1, ..., u_k) \in F_q^k\}$.

Définition 2.4.4. On dit qu'une matrice génératrice G d'un $[n, k, d]_q-$code C est sous forme standard,ou normalisée, si $G = (I_k \setminus B)$; où I_k est la matrice identité $k \times k$ et $B \in \mathcal{M}_{k,n-k}(\mathbb{F}_q)$. Dans ce cas si $u := (u_1, ..., u_k) \in \mathbb{F}_q^k$ est un message, le mot de code associé à u est

$$c := uG = (u_1, ..., u_k, c_{k+1}, ..., c_n) \in C.$$

Donc les k premières composantes de c sont les symboles d'information (i.e. le message) et les $n-k$ suivantes sont les symboles de contrôle correspondant à la redondance. Si un code admet une matrice génératrice sous forme standard, on dit qu'il est systématique.

Exemple 2.4.1. :Soit $q = 2, F = \mathbb{F}_2$. Soit

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

Le rang de G est 3 et G est donc la matrice d'un $[5, 3, d]_2$ code linéaire. De plus G est sous forme standart, donc C est systématique. Si $u = (u_1, u_2, u_3) \in \mathbb{F}_2^3$ est un message, il est codé par le mot de code $uG = (u_1, u_2, u_3, u_1 + u_2, u_2 + u_3)$:Une base de C est $\{a = (1, 0, 0, 1, 0), b = (0, 1, 0, 1, 1), c = (0, 0, 1, 0, 1)\}$ et

$$|C| = |\mathbb{F}_2|^3 = 8$$

$$C = \{0, a, b, c, a + b, a + c, b + c, a + b + c\}.$$

Il est facile de voir que d = 2 (d est le plus petit poids d'un mot non nul).

Remarque 2.4.2. - Soit G une matrice génératrice d'un code C. Supposons qu'elle ne soit pas sous forme standart. On rappelle que les k lignes de G forment une base de C. La base de $\mathbb{F}_q^n$ restant fixée, il peut exister une autre base de C telle que la matrice G', dont les lignes sont les nouveaux vecteurs de base, soit sous forme standart. La matrice de G' est aussi une matrice génératrice de C.
Changer de base de C, la base de $\mathbb{F}_q^n$ restant fixée, revient à faire des opérations élémentaires sur les lignes de G (pas sur les colonnes) : échanger deux lignes, remplacer une ligne L par λL, avec $\lambda \in \mathbb{F}_q^*$,plus une combinaison linéaire des autres lignes.

Lemme 2. Soit C un $[n,k,d]_q-$code et $G = (a_{ij})_{1\leq i\leq k, 1\leq j\leq n}$une matrice génératrice de C. Le code est systématique si et seulement si le déterminant de la matrice G_k : $(a_{ij})_{1\leq i\leq k, 1\leq j\leq k}$ est non nul.

Preuve :
Si le déterminant de la matrice G_k est non nul, la méthode du pivot de Gauss sur les lignes donnera une matrice génératrice de C sous forme standart. Réciproquement, si C est systématique, C a une matrice génératrice sous forme standart $(I_k|B)$ qui vérifie la propriété demandée.
Si $\pi \in S_n$ est une permutation de {1,2,...,n}, p agit sur les éléments de $\mathbb{F}_q^n$ en permutant les composantes : si $x := (x_1, ..., x_n) \in \mathbb{F}_q^n$, $\pi(x) := (x_{\pi(1)}, ..., x_{\pi(n)})$.

Définition 2.4.5. Deux codes C_1 et C_2 de même type $[n,k,d]_q$ sont dits équivalents s'il existe une permutation $\pi \in S_n$ telle que $\pi(C_1) = C_2$. Si G_1 est une matrice génératrice de C_1, on obtient une matrice génératrice de C_2 en appliquant la permutation π aux lignes de G_1, c'est à dire en permutant les colonnes.

De la remarque 2.4.2, on déduit

Proposition 2.4.2. *Tout code linéaire est équivalent à un code systématique.*

Preuve : Supposons que C ne soit pas un code systématique. Soit G une matrice génératrice du code C. Comme le rang de G est égal à k, il existe un mineur $k \times k$ non nul. Par une permutation des colonnes de G on amène ce mineur aux k premières colonnes : on obtient une matrice génératrice d'un code équivalent et qui lui est systématique.

Exemple 2.4.2. Soit q = 2, $F = \mathbb{F}_2$ et soit

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

Le premier mineur 4×4

$$\begin{vmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{vmatrix}.$$

est nul car la somme des deux premières colonnes est égal à la troisième. Par contre

$$\begin{vmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{vmatrix} \neq 0.$$

On en déduit que rg(G) = 4, donc G est la matrice d'un$[6, 4, d]_2$ code linéaire C mais, à cause de la remarque du début, C n'est pas systématique. On permute les colonnes (pour amener le mineur non nul aux 4 premières colonnes) par

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 4 & 5 & 3 & 6 \end{pmatrix}.$$

d'où la matrice

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

puis on fait le pivot de Gauss sur les lignes de G' et on obtient

$$G_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

qui est la matrice génératrice d'un code systématique équivalent à C.

2.4.2 Matrice de contrôle

Définition 2.4.6. Soient C un $[n, k, d]_q$-code et S une application linéaire surjective

$$S : F_q^n \longrightarrow F_q^{n-k}$$

telle que Ker S =C. Soit $x \in F_q^n$. Le syndrôme de x est $S(x) \in F_q^{n-k}$. On a donc

$$x \in C \Longleftrightarrow S(x) = 0.$$

Soit H la matrice de S dans la base fixée de F_q^n et une base de F_q^{n-k}

Définition 2.4.7. Soit C un $[n, k, d]_q-$code. Une matrice de contrôle de C est une matrice $H \in \mathcal{M}_{n-k,n}(\mathbb{F}_q)$ de rang maximum (= n-k) telle que

$$x = (x_1, ...x_n) \in C \Longleftrightarrow S(x) = H \begin{pmatrix} x_1 \\ . \\ . \\ . \\ x_n \end{pmatrix} = 0.$$

Proposition 2.4.3. *Soient C un $[n, k, d]_q-$code, G une matrice génératrice de C. Si $H \in \mathcal{M}_{n-k,n}(\mathbb{F}_q)$ est de rang maximum (n-k), alors*
H est une matrice de contrôle de $C \Longleftrightarrow H^tG = 0$.

Preuve :

Considérons les applications linéaires$E : F_q^k \longrightarrow C$, E bijective de matrice ${}^tG$ et $S : F_q^n \longrightarrow F_q^{n-k}$, S surjective de matrice H. Comme

$$C = \{E(u), \forall u \in \mathbb{F}_q^k\}$$

on a H est une matrice de contrôle de $C \Longleftrightarrow \forall u \in \mathbb{F}_q^k, S(E(u)) = 0 \Longleftrightarrow S \circ E = 0$: L'application linéaire $S \circ E = 0$ est l'application nulle si et seulement sa matrice est la matrice nulle.

Corollaire 2.4.4. *Soient C un $[n, k, d]_q$-code systématique, G une matrice génératrice normalisée de C, $G = (I_k|B)$. Alors $H = (-{}^tB|I_{n-k})$ est une matrice de contrôle de C.*
Réciproquement, si C un $[n, k, d]_q-$code et H est une matrice de contrôle de C de la forme $H = (A|I_{n-k})$,alors C un code systématique et une matrice génératrice normalisée de C est$G = (I_k| -{}^t A)$.

preuve : Si $H = (A|I_{n-k})$, H est bien de rang n-k. Il suffit donc de vérifier que $H{}^tG = (-{}^tB|I_{n-k}) \begin{pmatrix} I_k \\ {}^tB \end{pmatrix} = 0$.

Par exemple, si

$$G = \begin{pmatrix} 1 & 0 & a & b & c \\ 0 & 1 & d & e & f \end{pmatrix}, H = \begin{pmatrix} -a & -d & 1 & 0 & 0 \\ -b & -e & 0 & 1 & 0 \\ -c & -f & 0 & 0 & 1 \end{pmatrix}.$$

Alors

$$H{}^tG = \begin{pmatrix} -a & -d & 1 & 0 & 0 \\ -b & -e & 0 & 1 & 0 \\ -c & -f & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ a & d \\ b & e \\ c & f \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Remarque 2.4.3. -On considère le produit scalaire habituel sur F_q^n et un $[n, k, d]_q$-code C. L'orthogonal de C, noté $C^\perp$, est un $[n, n-k, d^\perp]_q$-code, mais il n'y a pas de relation à priori entre d et $d^\perp$. Une matrice génératrice de C est une matrice de contrôle de $C^\perp$ et réciproquement. En utilisant l'orthogonal de C on peut montrer que si C est systématique de matrice génératrice $G := (I_k|B)$, alors $H := (-{}^tB|I_{n-k})$ est une matrice de contrôle de C.

2.4.3 Détermination de la distance minimum

En général c'est un problème compliqué.

Proposition 2.4.5. *Soit C un $[n, k, d]_q$-code, H une matrice de contrôle de C. Alors d est égal au nombre minimum de colonnes de H qui sont linéairement dépendantes (en tant que vecteurs de $\mathbb{F}_q^{n-k}$). Ceci sous-entend que tout ensemble de (d-1) colonnes de H est un système libre.*

Preuve : Notons $h_1, ..., h_n$ les vecteurs colonnes ($h_i \in \mathbb{F}_q^{n-k}$) de H. Pour tout $x \in \mathbb{F}_q^n$, on a

$$S(x) = \sum_{i=1}^{n} x_i h_i \in \mathbb{F}_q^{n-k}$$

Si $c \in C$, on a donc

$$S(x) = \sum_{i=1}^{n} c_i h_i = 0.$$

Supposons que $c \in C$ soit de poids r ; par suite $r \geq d$ et si $supp(c) := \{i, c_i \neq 0\} = \{i_1, ..., i_r\}$ alors

$$\sum_{j=1}^{r} c_{i_j} h_{i_j} = 0$$

donc les r colonnes $\{h_{i_1}, ..., h_{i_r}\}$ sont linéairement dépendantes. Comme d est le plus petit poids d'un mot non nul de C, on a le résultat voulu.
On retiendra de cette démonstration que :
$\exists$ un mot de poids r dans $C \iff \exists r$ colonnes de H linéairement dépendantes

Exemple 2.4.3. Code de Hamming binaire -Soit r > 1 et soit $\mathcal{H}_r$ le code de longueur $n = 2^r - 1$ sur $\mathbb{F}_2$ ayant pour matrice de contrôle la matrice dont les colonnes sont tous les vecteurs non nuls de $\mathbb{F}_2^r$ apparaissant une fois et une seule. Remarquons que $|(\mathbb{F}_2^r)^*| = 2^r - 1 = n$ et que H est bien de rang maximum puisque le mineur correspondant aux r vecteurs de la base canonique de $\mathbb{F}_2^r$ est un mineur $r \times r$ de H. La dimension de $\mathcal{H}_r$ est k tel que $n - k = r$, soit $k = n - r$. Montrons que sa distance minimum est d = 3.
1. S'il existait un mot de poids 1, une colonne de H serait nulle, ce qui est faux.
2. S'il existait un mot de poids 2, deux colonnes de H seraient égales (sur F_2, $h_{i_1} + h_{i_2} = 0 \Rightarrow h_{i_1} = h_{i_2}$), ce qui est faux.
3. Si h_{i_1} et h_{i_2} sont deux colonnes distinctes, $h_{i_3} = h_{i_1} + h_{i_2} \neq 0$ en est une autre, donc il existe un mot de poids 3.
Comme d est le plus petit poids d'un mot non nul de C, on a d = 3 par suite sa capacité de correction est t = 1. Le code $\mathcal{H}_r$ est donc un $[2^r - 1, 2^r - r - 1, 3]_2$ code qui corrige au plus une erreur. Comme d est impaire et que

$$M \sum_{i=0}^{t} C_n^i (q-1)^i = 2^{n-r}(1 + (2^r - 1)) = 2^n$$

on voit que le code de Hamming binaire est parfait.

Proposition 2.4.6 (Borne de Singleton). *Si C est un $[n, k, d]_q$-code, alors*

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

Preuve : Le rang d'une matrice de contrôle de C est égal à n-k et d'autre part il est supérieur à (d-1), puisque tout ensemble de (d-1) colonnes est un système libre d'après la proposition 2.4.5 : on obtient

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

ce qui donne le resultat.

Définition 2.4.8. Un $[n, k, d]_q$-code est dit MDS (en anglais : Maximum Distance Separable) si $n + 1 = k + d$.
Les codes MDS sont les meilleurs codes (les codes de Reed-Solomon sont MDS). La borne de Singleton montre que les paramètres asymptotiques $R := k/n$ et $\delta := d/n$ ne peuvent pas être simultanément arbitrairement proches de 1 puisque, pour un $[n, k, d]_q$-code, on a

$$R + \delta \leq 1 + 1/n.$$

2.4.4 Décodage des codes linéaires

Dans tout ce paragraphe, on considère un $[n, k, d]_q$-code C ; on note H une matrice de contrôle de C et $t := [(d - 1)/2]$ la capacité de correction. On rappelle que si $S : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^{n-k}$ est l'application linéaire de matrice H, le syndrôme de $x \in \mathbb{F}_q^n$ est $S(x) = Hx \in \mathbb{F}_q^{n-k}$
Les propriétés du syndrôme sont les suivantes.

Lemme 3. Soit $y \in \mathbb{F}_q^n$
1.$y \in C \Longleftrightarrow S(y) = 0$.
2. Si $y = c + e$, où $c \in C$ est le mot émis et e l'erreur, alors $S(y) = S(e)$.
3. Si $wt(y_1) \leq t$ et $wt(y_2) \leq t$, alors $S(y_1) = S(y_2) \Rightarrow y_1 = y_2$.

Preuve :
1. par définition de S.
2. S étant linéaire, $S(y) = S(c + e) = S(c) + S(e) = S(e)$.
3. Si $wt(y_1) \leq t$ et $wt(y_2) \leq t$, alors $wt(y_1 - y_2) \leq 2t < d$. De plus $S(y_1) = S(y_2) \Rightarrow S(y_1 - y_2) = 0 \Rightarrow y_1 - y_2 \in C$. Alors $y_1 - y_2 \in C$ et $wt(y_1 - y_2) < d$ implique $y_1 - y_2 = 0$.

Soit C un $[n, k, d]_q$-code de capacité de correction t et de matrice de contrôle H. Voici une méthode pour décoder tout mot reçu $y \in \mathbb{F}_q^n$ pourvu que y soit affecté d'au plus t

erreurs :$y = c + e_y$; avec $c \in C$ et $wt(e_y) \leq t$.
-Table de décodage - On considère toutes les erreurs éventuelles, c'est à dire tous les $e \in \mathbb{F}_q^n$ tels que $wt(e) \leq t$. Pour chaque $e \in \mathbb{F}_q^n$ tel que $wt(e) \leq t$, on calcule $S(e)$. D'après l'assertion 3. du lemme 3, si $e \neq e'$ alors $S(e) \neq S(e')$. On fait une table contenant ces informations.
-Soit $y = c + e_y \in \mathbb{F}_q^n$ un mot reçu. On calcule $s = S(y)$. On sait que $s = S(e_y)$.
$[i]$ si s figure dans la table, associé à e_0, on décode y par $y - e_0$.
$[ii]$ sinon, on peut dire que y est affecté de plus de t erreurs et on ne peut pas décoder.
Cette méthode de décodage est efficace mais coûteuse. Remarquons d'autre part que, même dans le cas $[i]$ on fera une erreur de décodage si, en fait, y est affecté de plus de t erreurs.

2.5 Quelques exemples

2.5.1 Le code de Hamming

1. Code de Hamming : $\mathcal{H}_r$, où r est un entier ≥ 2. On rappelle que c'est un code linéaire sur $\mathbb{F}_2$ de longueur $n = 2^r - 1$, de dimension $k = n - r$ et dont la matrice de contrôle a pour colonnes tous les vecteurs non nuls de $\mathbb{F}_2^r$ apparaissant une fois et une seule. La distance minimum du code $\mathcal{H}_r$ est $d = 3$. Par exemple le code $\mathcal{H}_3$ est un $[7, 4, 3]_2$-code systématique de matrice de contrôle H et de matrice génératrice G

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

Ce code permet donc de coder 4 symboles d'information et, si le message est $u := (u_1, u_2, u_3, u_4)$
le mot de code correspondant $c \in \mathbb{F}_2^7$ a pour symboles de contrôle

$$c_5 = u_1 + u_2 + u_3$$
$$c_6 = u_1 + u_3 + u_4$$
$$c_7 = u_1 + u_2 + u_4.$$

Le code $\mathcal{H}_3$ corrige $t = 1$ erreur. On remarque que, puisque le corps est $\mathbb{F}_2$, le syndrôme de $y \in \mathbb{F}_2^7$ s'obtient en faisant la somme des colonnes de H correspondant aux $y_i \neq 0$. La table de décodage contient le mot nul et les mots de poids 1 avec leur syndrôme.

$$(0, 0, 0, 0, 0, 0, 0) \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

$$(1,0,0,0,0,0,0)\begin{pmatrix}1\\1\\1\end{pmatrix},$$

Si $y := (1,1,0,1,1,1,1)$, $S(y) = (1,1,0)$, donc l'erreur est $e = (0,0,1,0,0,0,0)$. On décode y par $c = y - e = y + e = (1,1,1,1,1,1,1)$.

2. Orthogonal du code de Hamming : c'est le code simplexe binaire d'ordre r, dont les paramètres sont $[2^r - 1, r, 2^{r-1}]_2$. Il a la propriété remarquable suivante : il contient $2^r - 1$ mots non nuls,tous de poids 2^{r-1}, d'où son nom, puisque si on place les mots sur les sommets d'un cube de côté 1 dans $\mathbb{F}_2^n$, ils forment un simplexe régulier.

3. Code de Hamming étendu : on considère l'application linéaire

$$\begin{array}{ccc}\mathbb{F}_2^n & \longrightarrow & \mathbb{F}_2^{n+1}\\ (y_1,...,y_n) & \longrightarrow & (y_1,...,y_n,\sum_{i=1}^n y_i)\end{array}$$

On dit qu'on a ajouté un bit de parité. Pour $n = 2^r - 1$, l'image de $\mathcal{H}_r$ par cette application est le code de Hamming étendu dont les paramètres sont $[2^r - 1, 2^r - r - 1, 4]_2$ (il est en effet facile de voir que certains mots non nuls de poids minimum sont obtenus à partir des mots de poids minimum de $\mathcal{H}_r$ et donc sont de poids 4). Le dual de ce code est le code de Reed-Müller d'ordre 1.

2.5.2 Le code à répétitions

Ce code est la forme la plus simple de code par blocs : chaque bit à transmettre est simplement répété d fois dans le message transmis. Ainsi pour d = 3 sur $\mathbb{F}_2$ le message 1101 devient 111111000111. Ce code peut aussi être défini sur n'importe quel autre corps et sa matrice génératrice est toujours la matrice $1 \times d$ remplie de 1. Cette construction donne une distance minimale de d mais une dimension de 1 pour une longueur d aussi. On construit donc des codes de la forme [d, 1, d].
Le décodage est ensuite très simple puisqu'il suffit de garder le bit majoritaire dans chaque bloc. La capacité de correction ainsi obtenue est donc de $[\frac{d-1}{2}]$. Le principal problème est qu'avec ce code le taux de transmission est de $\frac{1}{d}$ ce qui est très peu. On peut toutefois noter que ce code est parfait quand on choisit d impair.

2.5.3 Les codes de Reed-Solomon

Les codes de Reed-Solomon ont été développés par Reed et Solomon dans les années 50 [35] mais avaient déjà été construits par Bush [7] un peu avant,dans un autre contexte. Ces codes sont certainement les codes par blocs les plus utilisés pour la correction d'erreurs en étant présents dans les CD, les DVD et la plupart des support de données numériques. Ils sont très utilisés car ils sont extrêmaux du point de vue de la capacité de correction.

a) Construction

La façon la plus simple de voir ces codes est en tant que code d'évaluation : chaque élément du support du code est associé à un élément du corps $\mathbb{F}_q$ sur lequel est défini le code et chaque mot de code est l'évaluation d'une fonction $f \in \mathcal{F}$ sur le support. Pour les Reed-Solomon on prend $\mathbb{F}_q = \mathbb{F}_{2^m}$ et $\mathcal{F}$ l'ensemble des polynômes de degré strictement inférieur à k sur $\mathbb{F}_{2^m}$. Ainsi on a bien un code linéaire de longueur $n \leq 2^m$ et de dimension k. Une matrice génératrice du code peut alors s'écrire :

$$\mathcal{G} = \begin{pmatrix} 1 & 1 & ... & 1 \\ \alpha_1 & \alpha_2 & & \alpha_n \\ . & & & . \\ . & & & . \\ . & & & . \\ \alpha_1^{k-1} & \alpha_2^{k-1} & & \alpha_n^{k-1} \end{pmatrix}$$

Par sa nature ce code a une distance minimale d'au moins $n-k+1$ car deux polynômes de degré < k distincts ne peuvent pas être égaux en plus de k - 1 positions distinctes. Cette distance est même exactement égale à $n-k+1$ puisque l'évaluation d'un polynôme de la forme $\prod_{i=1}^{k-1}(X-\alpha_i)$ est de poids n - k + 1. On a donc des codes sur $\mathbb{F}_{2^m}$ de la forme $[n, k, n-k+1]$ qui peuvent donc avoir à la fois un bon taux de transmission et une bonne capacité de correction. Notons que cette distance minimale est la meilleure que l'on puisse atteindre avec ces paramètres : une distance minimale supérieure entrerait en contradiction avec la dimension du code.

Ces codes ont aussi l'avantage de pouvoir corriger des rafales d'erreurs du fait de leur structure sur $\mathbb{F}_{2^m}$. En effet, une fois écrits en binaire, si une erreur atteint plusieurs bits à la suite il y a de bonnes chances pour que cela n'affecte qu'un seul bloc de m bits et donc ne crée au final qu'une seule erreur.

b) Décodage

Plusieurs algorithmes polynomiaux permettent de décoder efficacement $\frac{n-k}{2}$ erreurs dans un code de Reed-Solomon. Celui de Berlekamp-Massey [1 , 26] est le plus connu et le plus utilisé, mais l'algorithme le plus simple à expliquer est celui de Berlekamp-Welch [2].

On suppose que l'on a reçu un mot $c' = c + e$ où e est une erreur de poids ω et c un mot de code, évaluation d'un polynôme $\mathcal{P}_c$ de degré < k. On appelle polynôme locali-

sateur de e le polynôme unitaire $\mathcal{L}_e$ qui s'annule en tous les éléments du support où e est non nul. Le degré de $\mathcal{L}_e$ est donc ω. Puisque e n'est pas connu ce polynôme est aussi inconnu. Si on multiplie le mot reçu par l'évaluation de ce polynôme on trouve :

$$\forall x \in [1;n] \quad c'_i \times \mathcal{L}_e(\alpha_i) = (\mathcal{P}_c(\alpha_i) + e_i) \times \mathcal{L}_e(\alpha_i) = \mathcal{P}_c(\alpha_i) \times \mathcal{L}_e(\alpha_i) \qquad (2.1)$$

Le polynôme $\mathcal{P}_c$ de degré < k est lui aussi inconnu (c'est ce que l'on cherche à trouver), et on linéarise le système en introduisant le polynôme $\mathcal{N} = \mathcal{P}_c \times \mathcal{L}_e$ qui est donc de degré au plus $k - 1 + \omega$.On obtient alors le système linéaire suivant :

$$\forall x \in [1;n] \quad c'_i \times \mathcal{L}_e(\alpha_i) = \mathcal{N}(\alpha_i). \qquad (2.2)$$

Il est composé de n équations en $k+2\omega$ inconnues (ω inconnues pour $\mathcal{L}_e$ qui est unitaire de degré ω et $k + w$ pour$\mathcal{N}$). De plus, toute solution du système 2-1 donne aussi une solution pour ce système. Il suffit donc qu'il y ait un nombre fini de solutions à 2-2 pour pouvoir retrouver une solution. Ce sera le cas si $n \geq k+2\omega$ et donc si $\omega \leq \frac{n-k}{2}$. On peut donc facilement corriger jusqu'à la moitié de la distance construite des Reed-Solomon. C'est le maximum que l'on puisse faire si on veut garder un décodage unique.
Au-delà de cette borne de $\frac{n-k}{2}$ erreurs on peut encore décoder en temps polynomial en utilisant l'algorithme de Guruswami-Sudan [21 , 40]. Cet algorithme permet d'effectuer du décodage par liste, c'est-à-dire qu' il renvoie la liste de tous les mots plus proches que la distance fixée. Avec, on peut corriger jusqu'à $n - \sqrt{nk}$ erreurs, et même s'il peut renvoyer une liste de solutions,dans la pratique la liste ne contiendra en général qu'une seule solution.
Quand on essaye de corriger plus d'erreurs, il n'existe pas d'algorithme polynomial pour décoder : on arrive dans les instances difficiles du problème de polynomial reconstruction .

2.5.4 Les codes de Goppa

Ces codes seront étudiés en détail dans le chapitre 3 . Ils sont célèbres car ce sont les codes utilisés dans les cryptosystèmes de McEliece et Niederreiter et qu'ils sont aussi de très bons codes binaires.

2.5.5 Les codes cycliques, BCH, de Reed-Muller, de Golay...

Il existe beaucoup d'autres classes de codes par blocs, toutes présentant leurs intérêts, soit du point de vue de la capacité de correction, soit de la simplicité du codage ou du décodage. . . On ne fais que mentionner leur existence car leur connaissance n'est pas nécessaire à la compréhension de ce projet. Ils n'est en revanche pas exclu que des applications cryptographiques de la théorie algébrique des codes puissent passer par l'utilisation de ces codes : c'est d'ailleurs déjà le cas pour certains problèmes (recherche de fonctions booléennes, traitor tracing. . .).

2.5.6 Les codes aléatoires

Cette dernière catégorie n'est pas vraiment une famille de codes puisqu'il s'agit en fait de tous les codes construits sans structures particulières. Pour obtenir un code aléatoire il suffit de tirer une matrice génératrice aléatoire et de chercher son image. Bien sûr, une fois choisi, le code n'est plus aléatoire, mais de façon générale, un code construit de cette façon aura de bonnes propriétés en moyenne : il a en général une bonne distance minimale.
Malheureusement pour un tel code il n'existe pas d'algorithme de décodage polynomial.

Remarque 2.5.1. Afin d'être plus complet, il pourrait être judicieux de parler un peu des autres variétés de codes utilisées de nos jours. En effet les codes par blocs sont faciles à analyser et sont, historiquement, les codes les plus utilisés. Cependant,ils tendent à être souvent remplacés par des codes comme les turbo codes, LDPC ou autre codes convolutifs ayant de meilleures capacités de correction ou des algorithmes de décodage plus efficaces. Ils sont en général conçus pour effectuer du décodage souple, c'est-à-dire quand on associe à chaque bit reçu une probabilité d'erreur propre.
Ces codes sont cependant beaucoup plus difficiles à analyser, et il est même en général assez difficile de connaître leur exacte capacité de correction. Ils ont pourtant quand même déjà des applications en cryptographie, pour des attaques sur des chiffrements à flots [23 , 24] par exemple. Il faudra par contre certainement encore attendre un peu avant de voir apparaître les premiers cryptosystèmes les utilisant.

Chapitre 3

Les codes de Goppa

Introduits en 1970, les codes de Goppa [3, 19] sont des codes linéaires sur un corps fini $\mathbb{F}_p$. Ils ont été beaucoup étudiés dans un premier temps pour leurs propriétés en tant que codes correcteurs d'erreurs et des algorithmes de décodage efficaces qui ont été mis au point [26, 12]. Ensuite, avec l'apparition du cryptosystème de McEliece [28] ils ont été étudiés pour leurs propriétés cryptographiques.

Notons que le terme de code de Goppa désigne aussi parfois des codes appartenant à la famille des codes géométriques [20, 41]. Ces dernieres ont aussi de très bonnes propriétés et de multiples applications. Le terme de code de Goppa étudiés dans ce qui suit désignera les codes de Goppa classiques.

Dans ce chapitre nous verrons tout d'abord comment sont construits les codes de Goppa. Ensuit nous allons parlé de quelques algorithmes permettant de les décoder efficacement. Enfin, nous allons voir les propriétés particulières des codes de Goppa binaires, qui en font de bons codes pour la cryptographie.

3.1 Construction

Les codes de Goppa sont des codes linéaires sur un corps fini $\mathbb{F}_p$. Cependant, leur construction passe par l'utilisation d'une extension $\mathbb{F}_{p^m}$. Un code de Goppa $\Gamma(\mathcal{L}, g)$ est défini par son polynôme de Goppa g de degré t à coefficients dans $\mathbb{F}_{p^m}$ et son support $\mathcal{L} \subset \mathbb{F}_{p^m}$ de n éléments. Si on note $\alpha_0, ..., \alpha_{n-1}$ les éléments de son support, sa matrice de parité est alors obtenue à partir de la matrice suivante :

$$\mathcal{H}_{aux} = \begin{pmatrix} \frac{1}{g(\alpha_0)} & \cdots & \frac{1}{g(\alpha_{n-1})} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \frac{\alpha_0^{t-1}}{g(\alpha_0)} & \cdots & \frac{\alpha_{n-1}^{t-1}}{g(\alpha_{n-1})} \end{pmatrix}$$

Chaque élément de cette matrice est ensuite décomposé en m éléments de $\mathbb{F}_p$, placés en colonnes, en utilisant une projection de $\mathbb{F}_{p^m}$ dans $\mathbb{F}_p^m$. On passe ainsi d'une matrice de taille $t \times n$ sur $\mathbb{F}_{p^m}$ à une nouvelle matrice de parité $\mathcal{H}$ de taille $mt \times n$ sur $\mathbb{F}_p$.

Les éléments du code $\Gamma(\mathcal{L}, g)$ seront donc tous les éléments c de $\mathbb{F}_p^n$ tels que $\mathcal{H} \times c^T = 0$. C'est donc un code de longueur n et dimension $k \geq n - mt$. De plus, un tel code a une distance minimale au moins égale à t+1. En effet, toute sous-matrice carrée $t \times t$ de $\mathcal{H}_{aux}$ est inversible car $\mathcal{H}_{aux}$ s'écrit comme le produit d'une matrice de Vandermonde et d'une matrice diagonale inversible :

$$\mathcal{H}_{aux} = \begin{bmatrix} 1 & \dots & 1 \\ . & & . \\ . & & . \\ . & & . \\ \alpha_0^{t-1} & \dots & \alpha_{n-1}^{t-1} \end{bmatrix} \times \begin{bmatrix} \frac{1}{g(\alpha_0)} & & & 0 \\ & . & & \\ & & . & \\ & & & . \\ 0 & & & \frac{1}{g(\alpha_{n-1})} \end{bmatrix}$$

Donc, il n'existe pas de mot de code de poids inférieur ou égal à t. Les codes de Goppa sont donc de la forme $[n, k \geq n - mt, d \geq t + 1]$ sur $\mathbb{F}_p$, avec comme seule contrainte de longueur $n \leq 2^m$.

Par rapport aux codes de Reed-Solomon vus section 2.5.3 ils ont donc l'avantage de ne pas être limités pour leur longueur en fonction de la taille du corps sur lequel on veut travailler. De plus, si on fixe m = 1 on se retrouve avec des codes qui ont exactement les même performances que les Reed-Solomon.

Il est important de remarquer qu'on peut caractériser l'appartenance au code $\Gamma(\mathcal{L}, g)$ sans avoir besoin de faire appel à une matrice de parité du code. En effet,si comme pour la construction de $\mathcal{H}_{aux}$ on indexe les coordonnées des mots de code par les éléments du support $\mathcal{L}$ et que l'on note c_β la coordonnée du mot c associée à l'élément $\beta \in \mathcal{L}$, on a :

$$c \in \Gamma(\mathcal{L}, g) \iff \sum_{\beta \in \mathcal{L}} \frac{c_\beta}{z - \beta} = 0 \quad mod \quad g(z).$$

3.2 Décodage d'un code de Goppa

Afin de décoder les codes de Goppa, plusieurs techniques existent. ces techniques fonctionnent toutes sur le même principe : si on cherche à décoder un mot $c' = c + e$, où c est un élément du code de Goppa et e une erreur de poids inférieur ou égal à $\frac{t}{2}$, on va commencer par calculer un syndrome sur $\mathbb{F}_{p^m}$ de ce mot bruité, à partir de ce syndrome on va écrire ce que l'on appelle une équation clef et on finira le décodage en résolvant l'équation clef pour retrouver e.

3.2.1 L'algorithme de Patterson

Cet algorithme, mis au point en 1975 [34], est le plus ancien pour décoder les codes de Goppa. Il ne fait pas appel à la matrice de parité $\mathcal{H}$ décrite précédemment et n'a besoin que de la connaissance du support $\mathcal{L}$ et du générateur g du code $\Gamma(\mathcal{L}, g)$.

a) Le syndrome

Pour cet algorithme, le syndrome $\mathcal{R}_c$ d'un mot c est un polynôme sur $\mathbb{F}_{p^m}$ modulo $g(z)$. Il s'écrit :

$$\mathcal{R}_c(z) = \sum_{\beta \in supp(c)} \frac{c_\beta}{z - \beta} \quad mod \quad g(z).$$

Si ce syndrome est nul, le mot appartiendra au code : c'est la caractérisation alternative que nous venons de voir.

b) L'équation clef

Afin de pouvoir décoder $\frac{t}{2}$ erreurs dans le code de Goppa, il faut, à partir du syndrome $\mathcal{R}_e$ d'une erreur e de poids $\leq \frac{t}{2}$, retrouver les $\frac{t}{2}$ coefficients e_β dans la somme modulo $g(z)$.

Supposons que l'on ait donc reçu le syndrome $\mathcal{R}_e$ d'un mot du poids $\leq \frac{t}{2}$.
On appellera polynôme localisateur d'un mot e, le polynôme σ_e unitaire qui a pour racines tous les éléments du support de e. On a donc :

$$\sigma_e(z) = \prod_{\beta \in supp(e)} (z - \beta).$$

Ce polynôme est donc de degré $\leq \frac{t}{2}$. On introduit alors le polynôme $\omega_e(z)$ défini par :

$$\omega_e(z) = \sigma_e(z)\mathcal{R}_e(z) \quad mod \quad g(z). \tag{3.1}$$

Ce polynôme sera appelé polynôme évaluateur de l'erreur e car si on connaît sa valeur en un point β du support de e on peut déterminer la valeur de l'erreur en ce point. De même l'équation 3-1 sera appelée équation clef modulo g. Si on est capable de résoudre cette équation, c'est-à-dire de trouver des polynômes ω_e et σ_e de degrés suffisamment petits la vérifiant, on pourra retrouver e et décoder.

c)Résolution de l'équation clef

La résolution peut se faire de deux façons différentes : soit à l'aide de l'algorithme de Berlekamp-Massey [26] (comme expliqué dans l'article original de Patterson [34]),

soit avec un algorithme d'Euclide étendu, qui a l'avantage d'être plus simple à présenter. En effet, on cherche à trouver ω_e et σ_e de degrés $\leq \frac{t}{2}$ qui vérifient :

$$\omega_e(z) = \sigma_e(z)\mathcal{R}_e(z) \quad mod \quad g(z) = \sigma_e(z)\mathcal{R}_e(z) + k(z)g(z).$$

Si on essaye de calculer le PGCD de $\mathcal{R}_e$ et g avec l'algorithme d'Euclide étendu, on va calculer à chaque étape des polynômes u_i, v_i et r_i vérifiant : $\mathcal{R}_e u_i + g v_i = r_i$.
A chaque étape les polynômes u_i et v_i seront de degré inférieur ou égal à i et le polynôme r_i sera de degré au plus $t - i$. Il existe donc une étape i_0 pour laquelle si on arrête l'algorithme on va trouver une solution à l'équation : $\sigma_e = u_{i_0}$ et $\omega_e = r_{i_0}$ à un coefficient scalaire près.

3.2.2 Les codes de Goppa sont des codes alternants

On peut également décoder les codes de Goppa en utilisant la matrice de parité telle qu'elle a été définie précédemment. En effet, cette matrice correspond à une matrice de parité de code alternant, et le code peut donc être décodé comme un code alternant (voir [25] pour plus de détails). Dans ce contexte, on va tout de même procéder de façon similaire, en calculant d'abord un syndrome, puis en établissant une équation clef et en la résolvant.

a) Le syndrome

Supposons encore une fois que l'on a reçu un mot bruité $c' = c + e$. On part du syndrome sur $\mathbb{F}_p$ défini par $\mathcal{R}_e = \mathcal{H}c' = \mathcal{H}e$. Ce syndrome est de taille mt et on va le réécrire sous la forme d'un polynôme de degré t - 1 à coefficients dans $\mathbb{F}_{p^m}$: les m premiers termes constituent le terme constant, les m suivants le coefficient de degré 1, et ainsi de suite. . .

Notons que, vue la forme de $\mathcal{H}_{aux}$, on peut aussi écrire ce syndrome sous la forme suivante :

$$\mathcal{R}_e = \sum_{\beta \in supp(e)} \frac{e_\beta}{g(\beta)} \frac{1}{1 - \beta z} \quad mod \quad z^t.$$

b) L'équation clef

La forme de l'équation clef vient alors très facilement et on écrira :

$$\widetilde{\sigma}_e(z)\mathcal{R}_e(z) = \omega_e(z) \quad mod \quad z^t.$$

En notant toutefois que le localisateur $\widetilde{\sigma}_e$ est cette fois-ci le polynôme qui a pour racines les inverses des éléments du support :

$$\widetilde{\sigma}_e(z) = \prod_{\beta \in supp(e)} (1 - \beta z).$$

c)Résolution de l'équation clef

Pour résoudre cette nouvelle équation tout se passe exactement comme avec l'algorithme de Patterson, soit en appliquant l'algorithme de Berlekamp-Massey [26], soit un Euclide étendu.

3.3 Propriétés des codes de Goppa binaires

Les codes de Goppa binaires correspondent au cas particulier où l'on choisit p = 2. La matrice de parité $\mathcal{H}$ du code est alors une matrice binaire de taille $mt \times n$ construite à partir de la matrice $\mathcal{H}_{aux}$ de taille $t \times n$ à coefficients dans $\mathbb{F}_{2^m}$ décrite précédemment. Comme nous allons le voir, par rapport au cas général, les codes binaires présentent certains avantages les rendant, par la même occasion, bien adaptés à des usages cryptographiques.

3.3.1 Une nouvelle caractérisation des éléments du code

Comme nous l'avons vu section 3-1, les mots du code sont les mots vérifiant :

$$\sum_{\beta \in \mathcal{L}} \frac{c_\beta}{z - \beta} = 0 \quad mod \quad g(z).$$

Dans le cas binaire c_β ne peut être égal qu'à 0 ou 1 et l'équation peut donc se réécrire :

$$\sum_{\beta \in supp(c)} \frac{1}{z - \beta} = 0 \quad mod \quad g(z).$$

or, si on dérive le polynôme localisateur de c on trouve une écriture similaire :

$$\frac{d\sigma_c(z)}{dz} = \sigma_c(z) \sum_{\beta \in supp(c)} \frac{1}{z - \beta}.$$

Le polynôme g ayant été choisi sans racines sur $\mathcal{L}$ et σ_c étant scindé, $\frac{d\sigma_c(z)}{dz}$ sera égal à 0 modulo $g(z)$ si et seulement si la somme l'est aussi.

Un mot appartient donc à un code de Goppa binaire $\Gamma(\mathcal{L}, g)$ si et seulement si la dérivée de son polynôme localisateur est divisible par le générateur g du code.

3.3.2 Capacité de correction

Pour l'instant, la seule contrainte sur g était de ne pas avoir de racine parmi les éléments de $\mathcal{L}$. Si on lui ajoute la contrainte supplémentaire d'être sans facteurs multiples,

on va pouvoir doubler la capacité de correction du code. En effet, quand on utilise la caractérisation précédente, on veut que la dérivée du localisateur d'un mot du code soit divisible par g. Or, sur $\mathbb{F}_{2^m}$, la dérivée d'un polynôme ne contient pas de facteurs de degré impair, et il existe donc toujours un polynôme f vérifiant :

$$\frac{d\sigma_c(z)}{dz} = f^2(z).$$

De ce fait, si g divise la dérivée de σ_c, il divisera aussi f^2. Si g est sans facteurs multiples, cela veut donc dire qu'il doit nécessairement diviser f.

Un mot qui appartient au code $\Gamma(\mathcal{L}, g)$ avec g sans facteurs multiples appartiendra donc aussi au code $\Gamma(\mathcal{L}, g^2)$ qui a lui une distance minimale de 2t+1 et un algorithme de décodage jusqu'à t erreurs. Un code de Goppa binaire a donc une distance minimale plus grande et une capacité de correction doublée.

a) L'algorithme de décodage

Si on reprend la caractérisation avec la matrice de parité, cela signifie que la matrice de parité $\mathcal{H}_{aux}$ de taille $t \times n$ du code $\Gamma(\mathcal{L}, g)$ engendre toutes les lignes de la matrice de parité du code $\Gamma(\mathcal{L}, g^2)$. On peut donc calculer une matrice $\mathcal{K}$ de taille $2t \times t$ telle que :

$$\mathcal{K} \times \mathcal{H}_{aux}(g) = \mathcal{H}_{aux}(g^2).$$

Pour décoder t erreurs dans le code $\Gamma(\mathcal{L}, g)$ il suffit donc de calculer le syndrome normalement avec $\mathcal{H}_{aux}(g)$, de l'étendre à un syndrome double à l'aide de $\mathcal{K}$ et de résoudre l'équation clef modulo z^{2t}.

b) L'algorithme de Patterson pour le cas binaire

Dans son papier d'origine, Patterson donne un autre algorithme pour faire cela sans avoir à étendre le syndrome, et en restant modulo $g(z)$. Il faut pour cela d'abord remarquer que dans le cas binaire, le polynôme ω_e que l'on cherche est exactement la dérivée de σ_e. L'équation clef peut donc se réécrire :

$$\sigma_e(z)\mathcal{R}_e(z) = \frac{d\sigma_e(z)}{dz} \quad mod \quad g(z).$$

Pour la résoudre on décompose σ_e en une partie paire et une partie impaire :
$\sigma_e(z) = u(z)^2 + zv(z)^2$. On procède ensuite en trois étapes :
1. on calcule $h(z) = \frac{1}{\mathcal{R}_e(z)}$ mod $g(z)$.
2. on calcule $S(z)$ tel que $S(z)^2 = z + h(z)$mod g(z) (c'est l'étape qui correspond à l'extension du syndrome).

3. on résout la nouvelle équation clef : $v(z)^2 S(z)^2 = u(z)^2 \bmod g(z)$ qui se réécrit pour g sans facteurs multiples : v(z)S(z) = u(z) mod $g(z)$
On se ramène ainsi à résoudre l'équation pour des polynômes de degré deux fois plus bas qui permettent de reconstituer ensuite le polynôme localisateur entier.

3.3.3 Indistinguabilité

L'indistinguabilité des codes de Goppa binaires désigne le fait qu'il est calculatoirement difficile de distinguer un code de Goppa dont on ne connaît ni le support, ni le générateur d'un code aléatoire de même longueur et même dimension.

Un distingueur de code de Goppa sera un algorithme capable de dire en temps τ, avec une probabilité supérieure à $\frac{1}{2} + \varepsilon$, si un code qu'on lui donne en entrée est un code de Goppa ou non. Il est conjecturé qu'un tel distingueur ne peut pas exister avec un quotient $\frac{\tau}{\varepsilon}$ polynomial en les paramètres du code.

a) Code de Goppa permuté

La propriété d'indistinguabilité n'a donc pas besoin de faire appel à une matrice de parité particulière pour avoir un sens. Cependant, plutôt que de parler d'un code dont on ne connaît pas le support on a en général tendance à parler d'un code de Goppa permuté, ce qui désigne en fait le code associé à une matrice de parité $\mathcal{H}'$, obtenue en < mélangeant > la matrice $\mathcal{H}$.

Les deux façons de voir les choses sont pourtant à peu près les mêmes : pour représenter un code $\Gamma(\mathcal{L}, g)$ sans donner $\mathcal{L}$ et g, le seul moyen est de donner une matrice génératrice ou une matrice de parité $\mathcal{H}'$ de ce code.

Si $\mathcal{H}_g$ désigne la matrice de parité obtenue directement à partir de $\mathcal{H}_{aux}$ pour un g donné et quand les α_i sont ordonnés, alors la matrice de parité $\mathcal{H}'$ utilisée pour représenter $\Gamma(\mathcal{L}, g)$ est nécessairement de la forme :

$$\mathcal{H}' = \mathcal{Q}\mathcal{H}_g\mathcal{P}$$

où $\mathcal{P}$ est une permutation et $\mathcal{Q}$ une matrice inversible.
En effet, $\mathcal{P}$ sera la permutation qui transforme les α_i bien ordonnés en $\mathcal{L}$, et une fois permutée $\mathcal{H}_g\mathcal{P}$ ayant le même noyau que $\mathcal{H}'$ (ce sont deux matrices de parité d'un même code $\Gamma(\mathcal{L}, g)$, il existe une application linéaire $\mathcal{Q}$ inversible pour passer de l'une à l'autre.

Avec ce point de vue, un distingueur de code de Goppa permuté sera un algorithme prenant en entrée une matrice binaire $\mathcal{H}'$ et décidant en temps τ avec une probabilité d'être exacte de $\frac{1}{2} + \varepsilon$ s'il existe deux matrices $\mathcal{Q}$ et $\mathcal{P}$ et un polynôme g permettant d'avoir

$$\mathcal{H}' = \mathcal{Q}\mathcal{H}_g\mathcal{P}$$

b) Les meilleurs distingueurs

Aucun bon distingueur de code de Goppa binaire n'est connu à ce jour. Les meilleures techniques connues consistent à énumérer tous les codes de Goppa ayant les paramètres adéquats, et pour chacun d'eux, tester s'il est équivalent au code à distinguer.

Afin d'avoir un meilleur distingueur il faudrait être capable d'exhiber une propriété des codes de Goppa, invariante par passage à un code équivalent , qui ne soit pas vérifiée (avec une probabilité de $\frac{1}{2} + \varepsilon$ au moins) pour un code aléatoire. Malheureusement, les invariants que l'on sait calculer facilement pour un code ne nous apportent aucune information. Même l'énumérateur des poids, qui est certainement le meilleur invariant qui soit en matière de codes, est en général très proche de la distribution binomiale d'un code aléatoire.

Chapitre 4

Cryptographie à clef publique

4.1 Définitions

Définition 4.1.1. La cryptographie est l'ensemble des techniques permettant d'écrire un message de façon brouillée afin que seul son destinataire légitime soit capable de comprendre la teneur du message.

En dépit de l'évolution des moyens de communication depuis l'antiquité, il a toujours été difficile de garantir la sécurité du canal par lequel transite un message. Un cavalier transportant un rapport sur la position d'une armée peut être intercepté par des éclaireurs ennemis ; un employé du télégraphe transmettant des consignes d'investissements boursiers peut être à la solde d'un actionnaire concurrent ; un ordre militaire transmis par les ondes radios peut être capté par n'importe quel récepteur réglé sur la bonne fréquence ; un courriel qui transite sur Internet peut être lu par n'importe quel ordinateur sur la chaîne reliant l'expéditeur au destinataire. Aussi, la problématique d'établir des communications sécuriséees en utilisant un médium non sécurisé a toujours été d'importance en vue d'applications militaires, financières, ou simplement du respect de la vie privée.

Définition 4.1.2. Le chiffrement est la fabrication du message chiffré à partir du clair et de la clé de chiffrement.
Le déchiffrement est l'extraction du message clair à partir du chiffré en utilisant la clé de déchiffrement.
Le décryptage ou l'attaque est l'extraction du message clair à partir du chiffré sans connaître la clé de déchiffrement.
La cryptanalyse est l'étude théorique d'un système de chiffrement en vue de mettre au point des algorithmes de décryptage.

Définition 4.1.3. Un système cryptographique est la donnée d'un 5-uplet (P,C,K,E,D)vérifiant :
1) P est un ensemble fini de blocs de texte claires possibles.

2) C est un ensemble fini de blocs de texte chiffrés possibles.
3) K que l'on appelle espace des clefs est un ensemble fini de clefs possibles.
4) $\forall k \in K, \exists$ une fonction de chiffrement $e_k \in E$ et une fonction de déchiffrement $d_k \in D$ avec :

$$\begin{array}{rccc} e_k : & P & \longrightarrow & C \\ & m & \longrightarrow & e_k(m) = c \end{array}$$

$$\begin{array}{rccc} d_k : & C & \longrightarrow & P \\ & c & \longrightarrow & d_k(c) = m \end{array}$$

Et $d_k o e_k(m) = m; \forall m$ texte claire de P

Remarque 4.1.1. Un bon cryptosystème doit permettre un chiffrement et un déchiffrement rapide, tout en interdisant toute possibilité de décryptage. Pour alléger les charges en mémoire informatique, on préfère aussi que les clés soient de petite taille.

Il existe deux types d'algorithmes de chiffrement :

Les algorithmes symétriques (ou schémas à clé secrète), ils sont pour la plupart connus et utilisés depuis longtemps (dès l'Antiquité pour certains) : l'envoyeur A (appelé Alice par convention) et le receveur B (appelé Bob) possèdent la même clé.

Le principal inconvénient de ce type d'algorithmes est que la clé doit rester secrète pour toute personne autre que A et B ; elle ne doit notamment pas être captée par un espion lorsque A et B la communiquent entre eux lors de la création de celle-ci (en fait, il faut assurer la confidentialité même de la clé qui est censée assurer la confidentialité du message !). Pour pallier ce type de problème on utilise un algorithme de mise à la clé. Cependant, un algorithme symétrique permet d'assurer simultanément confidentialité et intégrité, à condition, bien sûr, que deux personnes seulement connaissent la clé.

Les algorithmes asymétriques (ou schémas à clé publique), ils ont été introduits par Diffie et Hellman en 1976 : deux clés différentes sont nécessaires, une clé secrète s et une clé publique p de telle sorte que la connaissance de p ne permette pas de retrouver facilement s. Plus précisément, il ne doit pas exister d'algorithme efficace permettant de calculer s à partir de p.

Un algorithme asymétrique est généralement basé sur un problème complexe, c'est-à-dire difficile à résoudre, de sorte qu'il n'existe pas d'algorithme efficace permettant de trouver la solution.

L'efficacité d'un algorithme est susceptible d'évoluer en fonction des progrès des ordinateurs (gain en puissance et en rapidité), mais aussi en fonction des découvertes et des améliorations d'algorithmes permettant de résoudre les problèmes difficiles sous-jacents plus efficacement.

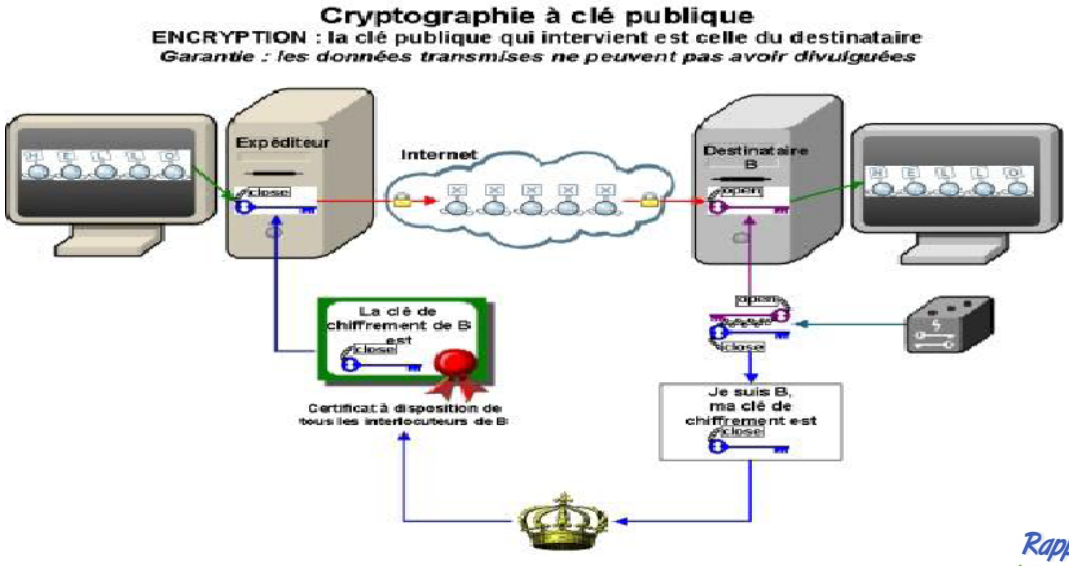

FIGURE 4.1 – Cryptographie à clé publique

Dans le but d'envoyer un message confidentiel à B, A utilise la clé publique de B pour chiffrer ce message ; B étant le seul possesseur de la clé secrète (seule clé autorisant le déchiffrement), il est le seul à même de déchiffrer le message envoyé par A.

Les problèmes complexes les plus fréquemment utilisés sont la factorisation de grands nombres ou l'inversion de certaines fonctions en arithmétique modulo n : Par exemple la fonction puissance $x \rightarrow x^e \mod n$ la recherche des racines e-ième est un problème difficile lorsque l'on ne sait pas factoriser n, ou la fonction exponentielle $x \rightarrow b^x \mod n$ (recherche du logarithme discret). On remarque que, s'il est difficile de résoudre l'un de ces problèmes, vérifier une solution est par contre très simple. C'est sur cette dissymétrie que sont basés les systèmes à clés publiques.

Ceci permet, contrairement aux algorithmes symétriques, de publier les clés (publiques, bien entendu), ce qui a pour avantage de permettre à n'importe qui d'envoyer un message à la personne qui a publié sa clé, ou de vérifier qu'un message a bien été rédigé par une personne dont la clé est dans l'annuaire. En contrepartie, cet annuaire étant accessible en lecture à tous, A voulant se faire passer pour B n'a qu'à remplacer la clé publique de B par la sienne pour parvenir à ses fins. Pour empêcher une telle fraude, il faut généralement une autorité qui certifie les clés publiques.

Les principaux schémas à clé publique sont RSA (du nom de ses auteurs Rivest, Shamir et Adleman en 1977), basé sur les racines e-ième d'un nombre, ce qui revient à un problème de factorisation de grands nombres, ainsi que les schémas de Diffie-Hellman et de Fiat-Shamir.

4.2 Fonction à sens unique

Considérons deux ensembles arbitraires X et Y et une fonction $f : X \to Y$. Soit $f(X)$ l'ensemble image de l'ensemble X par f.

Définition 4.2.1. La fonction f est dite à sens unique si pour tout x de X, il est facile de calculer f(x) (autrement dit, f peut être calculée en temps polynomial) et s'il est difficile de trouver, pour la plupart des $y \in f(X)$ un $x \in X$ tel que $f(x) = y$.

Nous avons cependant des candidats sous la forme de fonctions que nous savons calculer efficacement mais pour lesquelles aucun algorithme efficace pour les inverser n'est connu.

Les fonctions à sens unique ne peuvent pas servir directement comme système de chiffrement en utilisant $f(M)$ pour chiffrer M puisque même le destinataire légal ne serait pas en mesure de déchiffrer le cryptogramme.

La notion de fonction à sens unique à trappe est d'une utilité plus immédiate pour la cryptographie à clé publique.

Définition 4.2.2. Une fonction à sens unique $f : X \to Y$ est dite à trappe si elle peut être calculée efficacement dans le sens direct. Le calcul dans le sens inverse est aussi efficace pourvu qu'on dispose d'une information secrète -la trappe- qui permet de construire une fonction g telle que $g \circ f = Id$. Ainsi, il est facile de calculer l'image par f de n'importe quelle entrée mais calculatoirement impossible d'inverser f sans connaître g. De plus, il doit être facile de générer des couples (f, g). En outre, la publication de f ne doit rien révéler sur g.

4.3 Le principe général d'un cryptosystème public

Le principe général d'un cryptosystème public est qu'il est lié à une fonction à sens unique à trappe f de la manière suivante :
- Quand A veut envoyer à une seconde B un message $m \in \mathcal{M}$ (disons un entier modulo N), elle envoie en clair $m' = f_B(m)$; noter qu'elle connaît f_B (qui est dans l'annuaire).
- Pour déchiffrer le message, B calcule $f_B^{-1}(m')$ qui lui redonne m.

Confidentialité : La sécurité du système repose sur le choix de fonctions f "à sens unique" c'est-à-dire telles que f soit facile (rapide), f^{-1} soit impossible en pratique à calculer.
La confidentialité est assurée du fait qu'une tièrce personne C ne pourra déchiffrer le

message qu'en calculant f_B^{-1} (ce qui est supposé impraticable).

Identification : Quand A veut identifier B,
A envoie à B un message $m \in \mathcal{M}$ (disons un entier modulo N), elle l'envoie d'une façon claire chiffré de la sorte $m' = f_B(m)$; et demande à B de lui renvoyer le message reçu après son déchiffrement (i.e $m'' = f_A(f_B^{-1})(m'))$). La personne A déchiffre le message m'', ou encore elle calcule $m^* = f_A^{-1}(m'')$ et compare avec son message de déppart m. Si $m^* = m$ alors la personne qui est en contact avec lui est bien B, car c'est la seule personne qui est capable d'effectuer $f_B^{-1}(m')$ (Car f_B^{-1} est le secret de B).

La non répudiation : Le cryptosystème associé à une fonction à trappe s'utilise aussi pour générer des signatures électroniques. Une signature électronique garantit la non-répudiation d'un document électronique.
Pour signer un message $M \in \mathbb{Z}/N\mathbb{Z}^*$ avec un tel système, le signataire utilise sa clé privée f_A^{-1} pour signé un message m pour obtenir la signature : $S = f_A^{-1}(m)$.Seul le signataire possédant la clé privée f_A^{-1}, est donc capable de signer le message m. On vérifie la validité de la signature S en utilisant la clé publique f_A, en s'assurant que : $m = f_A(S)$.
Ainsi, n'importe qui peut vérifier la signature S de m. En réalité, comme pour le chiffrement, il faut pour des raisons de sécurité appliquer au message m à signer un format particulier.

4.4 Le problème NP-complet

4.4.1 Problème de décision

Chaque problème informatique peut se réduire à un problème de décision, c'est-à-dire un problème formulé comme une question dont la réponse est Oui ou Non, plutôt que par exemple ; le problème du voyageur de commerce, qui cherche, dans un graphe, à trouver la taille du cycle le plus court passant une fois par chaque sommet, peut s'énoncer en un problème de décision ainsi : Existe-t-il un cycle passant une et une seule fois par chaque sommet tel que la somme des coûts des arcs utilisés soit inférieure à B, avec $B \in \mathbb{N}$.

La théorie de la complexité repose sur la définition de classes de complexité qui permettent de classer les problèmes en fonction de la complexité des algorithmes qui existent pour les résoudre.
-Classe L : Un problème de décision qui peut être résolu par un algorithme déterministe en espace logarithmique par rapport à la taille de l'instance est dans L.

-Classe NL : Cette classe s'apparente à la précédente mais pour un algorithme non-déterministe.
-Classe P : Un problème de décision est dans P s'il peut être décidé par un algorithme déterministe en un temps polynomial par rapport à la taille de l'instance. On qualifie alors le problème de polynomial.
-Classe NP : Un problème NP est Non-déterministe Polynomial (et non pas Non polynomial, erreur très courante).
Les problèmes dans NP sont tous les problèmes qui peuvent être résolus en énumérant l'ensemble des solutions possibles et en les testant avec un algorithme polynomial. Par exemple, la recherche de cycle hamiltonien dans un graphe peut se faire avec deux algorithmes :

- le premier génère l'ensemble des cycles (ce qui est exponentiel)
- le second teste les solutions (en temps polynomial).

-Classe Co-NP (Complémentaire de NP) : C'est le nom parfois donné pour l'équivalent de la classe NP, mais avec la réponse non.

Les problèmes complets les plus étudiés sont les problèmes NP-complets. La classe de complexité étant par définition réservée à des problèmes de décisions, on parlera de problème NP-difficile pour les problèmes d'optimisation sachant que, pour ces problèmes d'optimisation, on peut construire facilement un problème qui lui est associé et est dans NP et qui est donc NP-complet.

4.4.2 Exemple d'un problème NP-complet

Voici un problème NP-complet, celui du sac à dos dont l'énoncé est le suivant :
Données :
Soient $A = \{a_1, ..., a_n\}$ et k un entier.
Question :
Existe-t-il un sous-ensemble de A dont la somme des toute ses éléments est égale à k ?

Exemple 4.4.1. Pour $A = \{43, 129, 215, 473, 903, 302, 561, 1165, 697, 1523\}$ et $k = 3231$,On remarque que $3231 = 129 + 473 + 903 + 561 + 1165$,donc $B = \{129, 473, 903, 561, 1165\}$ est une solution.
En principe, on peut toujours trouver la réponse en testant la somme de chaque sous-ensemble de A et en identifiant ceux qui ont une somme égale à k. Dans l'exemple, cela signifie qu'il faudrait tester 210 sous- ensembles. Si cette valeur n'est pas encore suffisante, que peut-on dire d'un n-uplet de plusieurs centaines d'entiers ?Par exemple, pour n = 300 sur une machine qui effectue un million d'opérations par seconde, il faudrait 6, 4.1076 années de temps de calcul !

Le problème du sac à dos nous permet de définir une fonction à sens unique comme

suit :
- Tout entier x tel que $0 \leq x \leq 2^{n-1}$ peut être représenté en binaire sur n bits.
- On définit $f(x)$ comme le nombre obtenu à partir de A, un n-uplet d'entiers, en sommant les a_i pour lesquels le bit i de la représentation binaire de $x = x_{n-1}x_{n-2}...x_0$ est égal à 1.
En notation matricielle, si A est un vecteur ligne de dimension n et si B_x représente le vecteur colonne de dimension n qui représente l'écriture binaire de x sur n bits,alors : $f(x) = A.B_x$. Ainsi si

$$A = (a_1, a_2, ..., a_n)$$

alors

$$\begin{array}{rcccc} f(1) & = & f(0...01) & = & a_n \\ f(2) & = & f(0...10) & = & a_{n-1} \\ f(3) & = & f(0...11) & = & a_n + a_{n-1} \end{array}$$

Exemple 4.4.2. Pour

$$A = (43, 129, 215, 473, 903, 302, 561, 1165, 697, 1523) \texttt{ et } k = 3231,$$

on a

$$f(364) = f(0101101100) = 129 + 473 + 903 + 561 + 1165 = 3231.$$

Alors, pour trouver x avec la seule connaissance de $f(x)$, on a autant de travail que pour résoudre le problème du sac à dos qui est NP-complet. La fonction f est un bon candidat de fonction à sens unique si n est un entier suffisamment grand. Comment fonctionnerait un tel système de chiffrement ?
- On applique f sur des blocs de n bits correspondants à une suite de bits du texte clair.
- Par exemple si on code a par $1 = 00001...z$ par $26 = 11010$ on aurait sur le texte sauna and health :

$$\begin{array}{cccccccc} sa & un & a & an & d & he & al & th \\ (2942 & 3584 & 903 & 3326 & 215 & 2817 & 2629 & 819) \end{array}$$

qui représente le cryptogramme.

Pour le moment notre système de chiffrement n'est pas à clé publique. Il peut toujours être utilisé comme chiffre à clé secrète. Mais le problème se pose pour le destinataire qui devrait résoudre le problème du sac à dos pour déchiffrer le cryptogramme.

Transformation en un système de chiffrement à clé publique :

Pour transformer la fonction définie ci-dessus en une fonction à sens unique à trappe,il nous faut replonger dans l'étude des problèmes de sac à dos pour trouver des classes de problèmes de sac à dos faciles.

Cas où les éléments du vecteur A forment une suite appelée sup-croissante

Définition 4.4.1. Une suite $(a_i)_{i\in\mathbb{N}}$ est dite super-croissante si elle verifie :

$$(\forall j \in \mathbb{N}^*)(\sum_{i=1}^{j-1} a_i \leq a_j).$$

Dans ce cas, le problème du sac à dos a une solution facile. Il suffit de parcourir A de la droite vers la gauche de la façon suivante :
Etant donné k, on compare k et $N_A := \sum_{i=1}^{n} a_i$.
Premier cas : Si $k > N_A$ alors $S = \emptyset$.
second cas : Si $k \leq N_A$. On pose $r = 0$ et $k = k_0$ alors il existe un $j > 1$ tel que $k_0 > a_{n-j}$, on considère i_1, le plus petit entier j tel que $a_{n-j} \leq k_0$ et on pose $k_1 = k - a_{n-i_1}$.
a)$k_1 = k_0 - a_{n-i_1} = a_{n-i_2}$, alors on a $k = a_{n-i_1} + a_{n-i_2}$ et le procédé est terminé.
b)$k_1 = k - a_{n-i_1} > a_{n-i_1-1}$, alors $S = \emptyset$ et le procédé s'arrête.
c)Il existe un $j > 1$ tel que $k_1 > a_{n-i_1-j}$, on considère le plus petit entier i_2 tel que $a_{n-i_2} \leq k_1$ et on pose $k_2 = k - a_{n-i_1} - a_{n-i_2}$, et on applique le procédé à partir à k_2.
Ce procédé se termine au maximum après n étapes, et il se termine quand on a atteint au maximum l'élément a_1.
Cet algorithme montre aussi que pour tout k le problème du sac à dos a au plus une solution.
Mais maintenant, si on publie A, déchiffrer sera aussi facile pour le destinataire que pour un cryptanalyste ! Pour éviter cela, on va perturber A de telle sorte que le n-uplet résultant B ressemble à un problème du sac à dos arbitraire.

Perturbation du problème :

Pour perturber A de telle sorte que le n-uplet résultant B ressemble à un problème du sac à dos arbitraire, on va utiliser une multiplication modulaire.
1. On choisit un entier $m > \sum_{i=1}^{n} a_i$. Comme A est sup-croissant, m est grand comparé aux a_i.
2. On choisit un autre entier t premier avec m i.e. qui vérifie $pgcd(t, m) = 1$. L'entier t sera le multiplicateur et m le module. Le choix de t premier avec m garantit l'existence d'un inverse t^{-1} modulo m obtenu par l'algorithme d'Euclide étendu.
3. On forme alors les produits $b_i \equiv a_i \times t \bmod m$ pour tout i ; $1 \leq i \leq n$ et on obtient un nouveau n-uplet B qui est utilisé comme clé publique.

Exemple 4.4.3. Pour A = (1,3, 5,11, 21, 44, 87, 175,349, 701),m = 1590 et t = 43 on obtient :
$t^{-1} = 37 \bmod m$, B = (43, 129, 215, 473, 903, 302, 561, 1165, 697,1523). Les entiers t, t^{-1} et m constituent l'information secrète (la trappe).

On chiffre de la même manière que précédemment :
pour chiffrer un message clair $p = (p_1, ..., p_n)$. On utilise la fonction de chiffrement associée la fonction a sens unique avec trape associée au vecteur n bits B(qui est publique) on calculant $Bp = c = b_1p_1 + ... + b_np_n$ qui sera le message chiffré a envoyer au destinataire qui a gardé comme secret propre à lui le vecteur A et les entiers m et t qui vont lui servir à dechiffrer tous les messages qui va recevoir comme des message chiffrés par sa clé publique B .

Comment déchiffrer ?

Comme le destinataire connaît les entiers t, t^{-1} et m, il peut retrouver A à partir de B. Après avoir reçu un bloc chiffré $c = \sum ci = \sum b_ip_i$:
1. il calcule $c' := t^{-1}c = t^{-1}\sum b_ip_i = \sum a_ip_i \bmod m$
2. il résout le problème du sac à dos défini par A et c' . La solution définit une suite unique p de n bits. Il s'agit aussi d'un bloc du clair car toute solution p du problème pour B et c est égale à p :
$c' \equiv t^{-1}c \equiv t^{-1}B.p \equiv t^{-1}tAp \equiv Ap \bmod m.$
Observons que $Ap < m$ car $m > \sum_{i=1}^{n} a_i$, ce qui implique que la congruence ci-dessus se simplifie en $c' = Ap$. Comme le problème défini par A et c' ne peut avoir plusieurs solutions, on a nécessairement $p = p'$.

Exemple 4.4.4. Déchiffrons
(2942,3584,903,3326,215,2817, 2629, 819) : On multiplie par $t^1 = 37 \quad modulo \quad m$ où $m = 1590$ on obtient :

$$(734, 638, 21, 632, 4, 879, 283, 93) :$$

Déchiffrons le bloc 734 pour l'exemple avec A = (1, 3, 5,11, 21, 44, 87, 175, 349, 701) qui est sup-croissant. 734 > 701, le dernier bit de p sera 1,
on recommence avec 734 - 701 = 33 > 21, le bit 6 de p sera 1,
on recommence avec 33 - 21 = 12 > 11, le bit 7 de p sera 1,
on recommence avec 12 - 11 = 1 = 1, le bit 10 de p sera 1 et on a fini.
Donc p = 1001100001 qui correspond au bloc de deux lettres sa.

4.5 Protocole d'échange de Clé de Diffie-Hellmann

Alice et Bob veulent s'échanger des informations de façon confidentielle en utilisant un système de communication non sécurisé (téléphone, internet,...), Charlie quant à lui désire savoir ce qu'ils veulent se dire. Alice et Bob vont donc devoir crypter (ou chiffrer) leur messages. La plupart des algorithmes de chiffrement des messages nécéssitent l'accord préalable d'Alice et Bob sur une clé qu'ils seront les seuls à connaitre

(dite clé secrète). Cela pose évidemment un problème si Alice et Bob sont très éloignés et ne disposent pas de moyens de communication sûrs. Bob et Alice doivent donc se mettre d'accord sur un chiffre (qui leur servira de clé secrète) de telle sorte que même si Charlie entend toute la conversation il ne puisse pas le connaitre. En 1976, Diffie et Hellmann ont proposé une solution pour y parvenir.

Tout d'abord Alice et Bob se mettent d'accord sur un grand nombre premier appelé cryptographique p, et un générateur g de $\mathbb{Z}/p\mathbb{Z}$. Charlie connait lui aussi p et g. Le protocole repose sur l'hypothèse que dans $\mathbb{Z}/p\mathbb{Z}$ l'application

$$\{0, ..., p-1\} \to \mathbb{Z}/p\mathbb{Z}$$

$$x \to g^x$$

est à sens unique. Pour tout nombre x, on sait comment calculer rapidement g^x modulo p, mais à partir de cette valeur on ne sait pas calculer rapidement x (on ne sait pas non plus prouver que c'est impossible !). Si on prend x assez grand, la méthode consistant à essayer tout les nombres jusqu'à tomber sur x.

Alice choisit alors secrêtement un nombre privé x et calcule g^x dans $\mathbb{Z}/p\mathbb{Z}$ elle envoie le résultat à Bob.

Calculer x est difficile pour Charlie et pour Bob également.

Bob choisit lui aussi secrêtement un nombre privé y, calcule g^y dans $\mathbb{Z}/p\mathbb{Z}$ et l'envoie à Alice. Charlie ne sait pas en déduire y.

Finalement Alice éléve à la puissance x le nombre que Bob lui a envoyé elle obtient : $(g^y)^x \quad modulo \quad p$

Bob, de même, élève à la puissance y le résultat que lui a envoyé Alice et obtient $(g^x)^y \quad modulo \quad p$, comme

$$(g^y)^x = (g^x)^y = g^{xy}$$

ce nombre :g^{xy} peut leur servir de clé secrête, et Charlie ne peut pas le calculer à partir de g^x et g^y.

Exemple 4.5.1. Supposons qu'Alice et Bob choisissent le nombre premier $p = 1259$ et $g = 3$.

Alice choisit $x = 144$ et calcule $3^{144} = 572[1259]$

Alice envoie 572 à Bob.

Bob choisit $y = 731$ et calcule $3^{731} = 900[1259]$

Bob envoie 900 à Alice.

Alice calcule $900^{144} = 572^{731} = 26[1259]$

Donc Alice et Bob peuvent utiliser la clé $K = 26$.

4.6 RSA

C'est l'algorithme cryptologique à clé publique le plus connu et le plus utilisé (plus de 2 millions de clés en circulation en 2002). Il tire son nom des initiales de ses inventeurs : Rivest, Shamir et Adleman. Il date de 1977.
Il repose sur la suite d'applications :

$$f_n : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$$

$$x \to x^c$$

où $c \in \mathbb{N}$, $c < \varphi(n)$ et $n = pq$ où p et q sont des nombres premiers.
Notons tout d'abord que le nombre d'éléments de $\mathbb{Z}/n\mathbb{Z}$ est n et donc qu'on peut représenter tous ces éléments par des chaines de r bits où $r = E(log_2(n)) + 1$.
Le calcul de f_n est une exponentiation modulaire, l'algorithme classique nécéssite un nombre de calcul qui vaut Cr^3 où C est une constante. Par ailleurs, on va voir ci-dessous qu'inverser f (si c est bien choisi) est aussi difficile que de factoriser n. Donc la suite (f_n) est à sens unique sous l'hypothèse que la factorisation de n est un probleme difficile, c'est à dire qu'il n'existe pas d'algorithme de complexité polynômial pour factoriser n.
En pratique Alice choisit deux grands nombres premiers p et q. Alice calcule $n = pq$, ensuite elle choisit un nombre c premier avec $\varphi(n)$, appelé exposant de chiffrement. Alice publie n et c. Bien sûr le "secret" d'Alice consiste en la factorisation de n et donc elle garde p et q secrets.

1. **Cryptage**
   Pour crypter le message $m \in \mathbb{Z}/n\mathbb{Z}$,Bob calcule $m' = m^c \bmod n$ et l'envoie à Alice.
2. **Décryptage**

Il faut montrer que la suite (f_n) est à trappe, plus précisément on va donner un algorithme qui permet à Alice de calculer rapidement m à partir de m' grâce à la connaissance de p et q.
Alice calcule $\varphi(n) = (p-1) \times (q-1)$. Elle calcule ensuite l'inverse de c dans $\mathbb{Z}/\varphi(n)\mathbb{Z}$ par l'algorithme d'Euclide (le nombre de calculs effectués par cet algorithme est Cr^2 où C : constante), on note d ce nombre appelé l'exposant de déchiffrement.
Pour décrypter m' Alice calcule m'^d.
En effet par défnition de d on a

$$cd \equiv 1[\varphi(n)]$$

donc il existe un entier k tel que

$$cd = 1 + k\varphi(n)$$

donc

$$m'^d = (m^c)^d = m^{cd} = m^{1+k\varphi(n)}$$

On utilise alors le résultat suivant :

Proposition 4.6.1. *Si $n = pq$ où p et q sont deux nombres premiers, alors pour tout entier k et tout nombre m de $\mathbb{Z}/n\mathbb{Z}$ on a*

$$m^{1+k\varphi(n)} = m$$

Démonstration.

Si m est inversible alors $m^{\varphi(n)} = 1$
donc

$$\begin{aligned} m^{1+k\varphi(n)} &= m \times (m^{\varphi(n)})^k \\ &= m \end{aligned}$$

Si m n'est pas inversible alors,on utilise le théorème des restes chinois et l'isomorphisme

$$\mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}.$$

On a donc $m = xy$ avec $x \in \mathbb{Z}/p\mathbb{Z}$ et $y \in \mathbb{Z}/q\mathbb{Z}$
On va montrer que dans $\mathbb{Z}/p\mathbb{Z}$ on a $x^{1+k(p-1)} = x$ pour tout entier k. En effet si x = 0 c'est évident et sinon d'après le petit théorème de Fermat on a $x^{p-1} = 1$ et donc $x^{1+k(p-1)} = x$. Comme $\varphi(n)$ est multiple de $p-1$ on a dans $\mathbb{Z}/p\mathbb{Z}$

$$x^{1+k\varphi(n)} = x$$

Le même raisonnement avec q montre que

$$y^{1+k\varphi(n)} = y$$

dans $\mathbb{Z}/q\mathbb{Z}$ .Donc

$$\begin{aligned} m^{1+k\varphi(n)} &= (xy)^{1+k\varphi(n)} \\ &= x^{1+k\varphi(n)} y^{1+k\varphi(n)} \\ &= xy \\ &= m \end{aligned}$$

Exemple 4.6.1. Si Alice a choisi p = 101 et q = 113 donc n = 11413. Alors $\varphi(n) = 112 \times 100 = 11200$. Supposons qu'Alice ait choisi c = 3533 (on vérifie facilement que pgcd(3533, 11200) = 1). On calcule d par l'algorithme d'Euclide, on trouve d = 6597. Bob veut transmettre le message 9726 à Alice, il calcule $9726^{3533} \bmod 11413 = 5761$ et envoie ce résultat. Alice reçoit 5761 et calcule $5761^{6597} \bmod 11413 = 9726$

Il reste donc à voir qu'il n'est pas possible d'inverser RSA sans factoriser n.

Proposition 4.6.2. *Si quelqu'un réussit à calculer d à partir de n et c alors il peut factoriser n.*

Démonstration.

Décomposons $cd-1$ sous la forme $2^k r$ avec r impair pour tout $m \in (\mathbb{Z}/n\mathbb{Z})^*$, on a $m^{cd-1} = 1$ on a alors trois possibilités :
1. $m^r = 1$
2. il existe l, $0 \leq l < k$ tel que $m^{2^l r} = -1$
3. il existe l, $0 \leq l < k$ tel que $m^{2^l r} \neq \pm 1$ et $m^{2^{l+1} r} = 1$.
En fait, le troisième cas se produit pour la moitié des messages m, dans ce cas $m^{2^l r}$ est une racine carrée de 1 autre que ± 1. Notons $x = m^{2^l r}$, on a donc $x^2 = 1$ dans $\mathbb{Z}/n\mathbb{Z}$ et donc $(x-1)(x+1) = 0$ et donc $pgcd(x-1, n)$ est soit p soit q !
En fait on peut aussi montrer que si quelqu'un peut déterminer pour tout message m son premier bit (position de m par rapport à $\frac{n}{2}$) ou son dernier bit (sa parité) à partir de m^c alors il peut factoriser n.Par conséquent, une cryptanalyse totale de RSA revient à factoriser n. Mais il n'est pas prouvé que cette factorisation est nécéssaire pour une cryptanalyse partielle (déchiffrer un message).

4.6.1 Une mauvaise utilisation de RSA

Supposons que Bob et Ted utilisent la même clé RSA : n = pq avec deux exposants de chiffrements différents : c_1 et c_2. On suppose que c_1 et c_2 sont premiers entre eux.
Si Alice envoie le même message m à Bob et Ted qu'elle crypte en utilisant RSA et que Charlie intercèpte les messages cryptés, alors Charlie peut calculer m.
On voit donc la nécéssité de générer une nouvelle clé RSA pour chaque nouvel utilisateur.
Comment procède Charlie ?
Il connait $m_1 = m^{c_1}[n]$ et $m_2 = m^{c_2}[n]$, il connait aussi n, c_1 et c_2 qui sont publics et il doit calculer m.
Avec l'algorithme d'Euclide il calcule a et b tel que $ac_1 + bc_2 = 1$, en suite il calcule $m_1^a \times m_2^b = m^{ac_1+bc_2} = m$.

Chapitre 5

Codes correcteurs et cryptographie à clef publique

5.1 Le cryptosystème de McEliece

C'est le plus ancien cryptosystème à clef publique utilisant des codes correcteurs d'erreurs. Il a été imaginé, comme son nom l'indique, par McEliece [28] en 1978, juste après l'invention des premiers cryptosystèmes à clef publique, à peu près en même temps que RSA [36], le système le plus utilisé aujourd'hui.

Comme tous les cryptosystèmes à clef publique, ce système est constitué de 3 algorithmes :

1. La génération de clefs
2. Le chiffrement (utilisant la clef publique)
3. Le déchiffrement (utilisant la clef secrète).

McEliece a suggéré d'utiliser les codes de Goppa, qui sont des codes linéaires avec un algorithme rapide de décodage.

5.1.1 Génération de clef

On commence par générer un code de Goppa corrigeant t erreurs et sa matrice de parité $\mathcal{G}$ de taille $k \times n$. On va maintenant mélanger cette matrice pour la rendre indistinguable d'une matrice aléatoire : pour cela on a besoin d'une matrice de permutation aléatoire $\mathcal{P}$ de taille $n \times n$ et d'une matrice inversible aléatoire $\mathcal{Q}$ de taille $k \times k$.

Le clef publique sera la matrice $\mathcal{G}_{pub} = \mathcal{Q} \times \mathcal{G} \times \mathcal{P}$ qui est indistinguable d'une matrice aléatoire. Par contre, la clef secrète est composée des trois matrices $\mathcal{Q}$, $\mathcal{P}$ et $\mathcal{G}$ qui permettent de retrouver la structure du code de Goppa et donnent donc accès à l'algorithme de décodage.

5.1.2 Chiffrement

Soit m un message de k bits qu'Alice veut chiffrer.
Alice calcule le mot c de longueur n associé à m :

$$c = m \times \mathcal{G}_{pub}$$

Ensuite, elle génère une erreur aléatoire e de longueur n et de poids t. Le chiffré sera simplement le mot de code bruité :

$$c' = c + e.$$

5.1.3 Déchiffrement

Pour déchiffrer, Bob connait $\mathcal{Q}$, $\mathcal{P}$ et $\mathcal{G}$, il calcule :

$$c' \times \mathcal{P}^{-1} = m\mathcal{G}_{pub}\mathcal{P}^{-1} + e\mathcal{P}^{-1} = m\mathcal{Q} \times \mathcal{G} + e\mathcal{P}^{-1}.$$

$m\mathcal{Q} \times \mathcal{G}$ est un mot du code de Goppa et $e\mathcal{P}^{-1}$ est une erreur de poids t, Bob peut décoder cette erreur et retrouver le message initial $m\mathcal{Q}$. Il ne reste plus qu'à multiplier par $\mathcal{Q}^{-1}$ pour retrouver le message m et avoir fini de déchiffrer.

5.1.4 Exemples

Exemple 5.1.1. Soit $\mathcal{C}_s$ le code de Goppa de paramètres [15, 3, 7] obtenu avec l'extension $m = 3$

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

$\mathcal{C}_s$ est capable de corriger jusqu'à 3 erreurs.
On considère la permutation $\pi = (1, 5, 7, 9, 11, 6, 14, 15, 4, 13, 10, 2, 12, 3)$ et

$$\mathcal{Q} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Matrice publique $\mathcal{G}_{pub}$ est alors :

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

On suppose que le message à chifrer m= (111). Le mot de code C associé est

$$c = m \times \mathcal{G}_{pub} = (00110 \quad 10001 \quad 10101)$$

Si on suppose que le canal de transmission introduit une erreur simple de poids 3 de valeur $e = (00001 \quad 00100 \quad 00010)$. Au lieu d'envoyer le message c c'est un autre message c' qui est envoyé :

$$c' = c + e = (00111 \quad 10101 \quad 10111)$$

Exemple 5.1.2. Soit G la matrice géneratrice du code binaire C :

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

On choisit la matrice de brouilleur $\mathcal{Q}$ et une matrice de permutation $\mathcal{P}$:

$$\mathcal{Q} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \qquad \mathcal{P} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

On calcul la clé publique qui correspond à la matrice $\mathcal{G}_{pub}$

$$\mathcal{G}_{pub} = \mathcal{Q} \times \mathcal{G} \times \mathcal{P}$$

On trouve

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

Chiffrement Soit $m = (111011101)$ le message à envoyer. Pour crypter le message m on va le chifrer par blocs : $m_1 = (111)$ $m_2 = (011)$ et $m_3 = (101)$. Si on suppose que le canal de transmission introduit une erreur simple de poids 1. Au lieu d'envoyer le message chiffré $c = m \times \mathcal{G}_{pub}$ c'est un autre message c' qui est envoyé :

$$c' = m \times \mathcal{G}_{pub} + e \quad avec \quad e = (00100)$$

Donc on obtient :

$$c'_1 = m_1 \times \mathcal{G}_{pub} + e = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$
$$c'_2 = m_2 \times \mathcal{G}_{pub} + e = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$
$$c'_3 = m_3 \times \mathcal{G}_{pub} + e = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

D'où le message chiffré est :

$$c' = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Déchiffrement A la réception de $X = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \end{pmatrix}$. on calcul d'abord :

$$d = X \times \mathcal{P}^{-1}$$

On a

$$\mathcal{P}^{-1} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad et \quad \mathcal{Q}^{-1} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Alors

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

Or

$$\begin{array}{rcc} d & = & X \times \mathcal{P}^{-1} \\ & = & (m \times \mathcal{G}_{pub} + e) \times \mathcal{P}^{-1} \\ & = & (m \times \mathcal{Q} \times \mathcal{G} \times \mathcal{P} + e) \times \mathcal{P}^{-1} \\ & = & m \times \mathcal{Q} \times \mathcal{G} + e \times \mathcal{P}^{-1} \\ & = & m \times \mathcal{Q} \times \mathcal{G} + e' \end{array}$$

Où $e' = e \times \mathcal{P}^{-1}$ est un vecteur du poids 1 (puisque $\mathcal{P}^{-1}$ est également une matrice de permutation).
Le syndrome de d trouvé est :

$$\begin{array}{rcc} S & = & d \times (\mathcal{Q} \times \mathcal{G})^T \\ & = & \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} \end{array}$$

Puisque e' est de poids 1 et

$$\begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \times \mathcal{G}_{pub} + e = X$$

On déduit que

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

5.2 La variante de Niederreiter

Cette variante du cryptosystème de McEliece a été mise au point par Niederreiter [31] en 1986. Elle est exactement équivalente du point de vue de la sécurité (mis à part qu'avec cette version, chiffrer deux fois le même message ne présente pas de risque) et est un peu plus efficace en temps de calcul. Elle fonctionne comme le chiffrement de McEliece, mais en utilisant la matrice de parité du code et en utilisant l'erreur pour contenir le message.

5.2.1 Génération de clef

Comme pour McEliece, on commence par générer un code de Goppa et sa matrice de parité $\mathcal{H}$ de taille $(n-k) \times n$. On génére une permutation aléatoire $\mathcal{P}$ de taille $n \times n$ et une matrice inversible $\mathcal{Q}$ de taille $(n-k) \times (n-k)$. La clef publique est :

$$\mathcal{H}_{pub} = \mathcal{Q} \times \mathcal{H} \times \mathcal{P}.$$

5.2.2 Chiffrement

Pour chiffrer Alice commence par coder le message m en un mot e_m de poids t et de longueur n. On peut donc mettre au plus $log_2(\binom{n}{t})$ bits d'information dans m.
Le chiffré transmis est le syndrome S de ce mot :

$$S = \mathcal{H}_{pub} \times e_m^T.$$

5.2.3 Déchiffrement

Pour déchiffrer on procède comme avec le système de McEliece. Bob commence par calculer :

$$\mathcal{Q}^{-1} \times S = \mathcal{H} \times \mathcal{P} e_m^T.$$

Encore une fois $\mathcal{P}e_m^T$ est de poids t lui aussi et il sait donc retrouver l'erreur correspondante au syndrome $\mathcal{Q}^{-1} \times S$. Il retrouve donc $\mathcal{P}e_m^T$ et donc aussi e_m. Bob en déduit alors le message clair m.

Ici, la clef publique fera $n(n-k)$ bits (ou $k(n-k)$ sous forme systématique), les blocs sont de longueur $log_2(\binom{n}{t})$ et le taux de transmission est $\frac{log_2(\binom{n}{t})}{n-k}$. Le chiffrement est lui beaucoup moins coûteux avec juste le codage en mot de poids constant (qui sera malheureusement souvent l'étape la plus coûteuse) et la somme de t colonnes de $\mathcal{H}$ pour un coût de $t(n-k)$. Le déchiffrement a lui un coût très similaire, mais un certain gain sur les produits de matrice puisque l'on travaille sur des tailles un peu plus petites.

Pour les paramètres [2048, 1685, 67], cela donne un taux de transmission de 0.66 (donc un peu moins que McEliece), des blocs de 363 bits pour 240 bits d'information (donc beaucoup plus courts que ceux de McEliece) et une matrice de 750 kbits (ou encore 600 kbits sous forme systématique).

Ce système a donc l'avantage d'avoir des blocs beaucoup plus courts et un chiffrement bien plus rapide, sans pour autant perdre beaucoup sur les autres points vis-à-vis du cryptosystème de McEliece.

5.2.4 Exemple

Soit $\mathcal{H}$ une matrice de parité de C_s

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

On considère la permutation $\pi = (1, 5, 7, 9, 11, 6, 14, 15, 4, 13, 10, 2, 12, 3)$
On tire la matrice inversible $\mathcal{Q}$ suivante :

$$\mathcal{Q} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

La matrice publique est

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

On suppose que le clair à chiffrer m est le mot de poids 3 suivant :

$$m = (010000010000001)$$

Le chiffré $S = 0110$

5.3 Remarques générales

Comme on peut le voir, les systèmes de McEliece et de Niederreiter sont très proches, l'un utilisant les matrices génératrice, et l'autre, les matrices de parité. En fait, il a été montré dans [12] que la sécurité des deux systèmes est équivalente. Toute attaque structurelle sur l'un se traduit par une attaque structurelle sur l'autre. La variante de Niederreiter présente cependant deux avantages :
- Le système de Niederreiter provoque asymptotiquement une expansion moins importante du message. L'expansion du message peut cependant également être évitée dans le système de McEliece.
- Le système de McEliece est vulnérable à une attaque par réémission de message : Si le même message est chiffré deux fois, les deux chiffrés seront distants d'au plus 2t. On peut alors trouver facilement les positions d'erreurs en comparant ces deux chiffrés. La

variante de Niederreiter n'est pas affectée par ce problème.
Malgré ces légères différences, puisque les sécurités des deux systèmes sont équivalentes, nous allons nous limiter par la suite à l'étude du système de McEliece.

Par rapport à des cryptosystèmes à clé publique issus d'autres primitives, le système de McEliece présente un avantage et deux inconvénients :

- Le chiffrement et le déchiffrement correspondent à des opérations classiques de codage et de décodage. Or on connaît des algorithmes très rapides pour effectuer ces deux types d'opérations. Le cryptosytème de McEliece peut donc être implémenté efficacement, pour un système asymétrique.
- Cependant, le message chiffré subit une expansion. Pour un clair de k caractères, on envoie un chiffré de n > k caractères. Néanmoins, cette expansion n'est pas très gênante en pratique. De plus, comme cela a été montré dans [5], il est possible d'y remédier en incluant une partie du message dans le vecteur d'erreur e.
- La clé publique est une matrice, elle occupe donc $nklog_2(q)$ bits de mémoire. Pour des valeurs raisonnable de n et k, cette clé publique risque de devenir très grosse, comparativement à ce que l'on sait faire avec d'autres systèmes asymétriques. C'est probablement, en bonne partie, à cause de la taille de sa clé publique que le cryptosystème de McEliece n'est pas utilisé en pratique.

5.4 Sécurité

Un attaquant connaît $\mathcal{G}_{pub}$ sans connaitre sa décomposition, et connait le chiffré c. Son but est de trouver une décomposition

$$c = m\mathcal{G}_{pub} + e,$$

c'est-à-dire de résoudre le problème Décodage borné$(\mathcal{C}, c, t)$ où $\mathcal{C}$ est le code engendré par $\mathcal{G}_{pub}$ La sécurité du système repose sur deux suppositions :

- Premièrement, que le code $\mathcal{C}$ est difficile à décoder au rayon t, ce qui est le cas par exemple si ce code ne peut pas être algorithmiquement distingué d'un code aléatoire.
- Deuxièmement, que la structure du code a bien été masquée, c'est-à-dire qu'il est difficile de retrouver les matrices $\mathcal{Q}$,$\mathcal{G}$, $\mathcal{P}$ à partir de $\mathcal{G}_{pub}$.

L'attaquant peut donc tenter d'attaquer le système par deux méthodes : Ou bien il tente directement de décoder le chiffré c dans le code $\mathcal{C}$ (attaque sur le chiffré), ou bien il essaye de retrouver une décomposition valide de la clé $\mathcal{G}_{pub} = \mathcal{Q}' \times \mathcal{G}' \times \mathcal{P}'$, qui lui permet de mettre au point son propre algorithme de décodage (attaque sur la clé).

La résistance du système à ces deux méthodes d'attaque dépend intrinsèquement de la famille de codes dans laquelle est choisie $\mathcal{G}$. Le choix de cette famille est le point

délicat dans la conception du cryptosystème. En effet, on veut utiliser une famille de codes que l'on sait décoder (et si possible rapidement), ce qui impose que ces codes soient structurés. D'un autre coté, des codes trop structurés prêtent le flanc à des attaques tirant parti de cette structure. Il faut donc atteindre un compromis difficile.

La famille des codes de Goppa est actuellement la meilleure candidate pour satisfaire ces deux contraintes. La famille des codes géométriques semblait prometteuse, mais ces codes trop structurés rendent le cryptosystème vulnérable à une attaque sur la clé.

Il est supposé dans [10] que les codes de Goppa sont indistinguables de codes aléatoires, et cette hypothèse n'a jamais été infirmée depuis. Ce sont donc de bons candidats pour implémenter le système de McEliece, car ils résistent à toutes les attaques structurelles. La meilleure attaque est encore de décoder le chiffré par un algorithme du type Ensemble d'information. La complexité d'une telle attaque est approximativement $O(n^3 \frac{C_n^t}{C_{n-k}^t})$.

Malheureusement, les codes de Goppa sont loin d'être optimaux, on a $t << \frac{n-k}{2}$. Aussi, pour obtenir une distance de décodage suffisante pour neutraliser l'attaque par ensemble d'information, il faut une clé publique de taille importante.
Des paramètres du cryptosystème correspondant à une sécurité de 280 sont donnés dans [8] ou [17]. On peut par exemple choisir $n = 2048$, $k = 1685$, $t = 33$. Cela donne des blocs message de 1685 bits, un taux de transmission de 0.82, et une clé publique d'environ 3Mbits (que l'on peut éventuellement réduire à 600 kbits). Pour comparaison, une clé RSA de 1024 bits atteint le même paramètre de sécurité.

Les codes de Reed-Solomon sont optimaux, et permettraient donc de résister aussi bien à un décodage par Ensemble d'information, pour une clé de taille beaucoup plus faible. Cependant, ils ne sont pas utilisables, car ils sont soumis à une attaque structurelle en temps polynomial(cf.[39]).

Les codes géométriques auraient également permis de réduire la taille de la clé par leur quasi-optimalité (Janwa et Moreno proposent l'utilisation d'une clé de 13 kbits dans [22]), mais ils sont aussi victimes d'une attaque structurelle rapide.

Chapitre 6

Fonctions de hachage

Une grande majorité des fonctions de hachage utilisées aujourd'hui en cryptologie est construite sur un même modèle : l'itération d'une fonction de compression selon le schéma présenté en 1989 par Damgard et Merkle [11, 29].Dans la plupart des cas, ces fonctions de compression sont construites dans le but d'être très rapides et utilisent pour cela des mécanismes empruntés aux systèmes de chiffrement symétrique [37, 30]. D'autres constructions, empruntant des techniques de la cryptographie à clef publique, permettent d'obtenir des fonction de compression dont la sécurité est prouvée [18], cependant, cela se fait toujours au détriment de la vitesse et ces fonctions sont beaucoup trop lentes pour être d'une utilité pratique.

6.1 Généralités sur les fonctions de hachage

Une fonction de hachage $\mathcal{H}$ est une fonction de $\mathbb{F}_2^*$ dans $\mathbb{F}_2^r$ qui prend en entrée un document de longueur quelconque et retourne en sortie une suite binaire de longueur donnée. Le but d'une telle fonction est de garantir l'intégrité d'un document, c'est-à-dire, permettre d'être certain qu'un document donné correspond bien à ce que l'on attend et qu'il n'a pas été modifié. Pour cela, la personne qui cherche à diffuser un document va rendre public le haché du document : son image par la fonction de hachage. Si la fonction de hachage vérifie les deux propriétés suivantes, alors il suffira de vérifier que le document reçu a le bon haché pour être certain de son intégrité. Pour cela elle doit donc être :

- non-inversible : pour un haché h donné, il doit être calculatoirement impossible de trouver un document D tel que $\mathcal{H}(D) = h$.
- sans collisions : il doit être calculatoirement impossible de trouver deux documents D_1 et D_2 tels que $\mathcal{H}(D_1) = \mathcal{H}(D_2)$.

Dans certains cas on peut être amené à ne considérer que la première de ces deux propriétés et à ne chercher que la non-inversibilité, mais on n'a plus alors à proprement

parler une fonction de hachage.

6.2 la construction de Merkle-Damgard

Dans cette section, on présente une manière particulière de construire une fonction de hachage à partir d'une fonction de de compression. Suivant cette construction, la fonction de hachage obtenue possède des propriétés de sécurité, telles que la résistance aux collisions, pour autant que la fonction de compression offre de telles propriétés.cette technique est souvent appelée construction de Merkle-Damgard.

C'est sur cette construction que sont basées à peu près toutes les fonctions de hachage utilisées de nos jours. Au coeur du schéma se trouve une fonction de compression : une fonction qui prend une entrée de longueur donnée s et retourne une sortie de longueur r. Comme son nom l'indique cette fonction compresse et il faut donc que s soit strictement plus grand que r. C'est en itérant cette fonction de compression que l'on obtient une fonction qui prend une entrée de taille quelconque.

Dans ce schéma chaque élément a son importance :

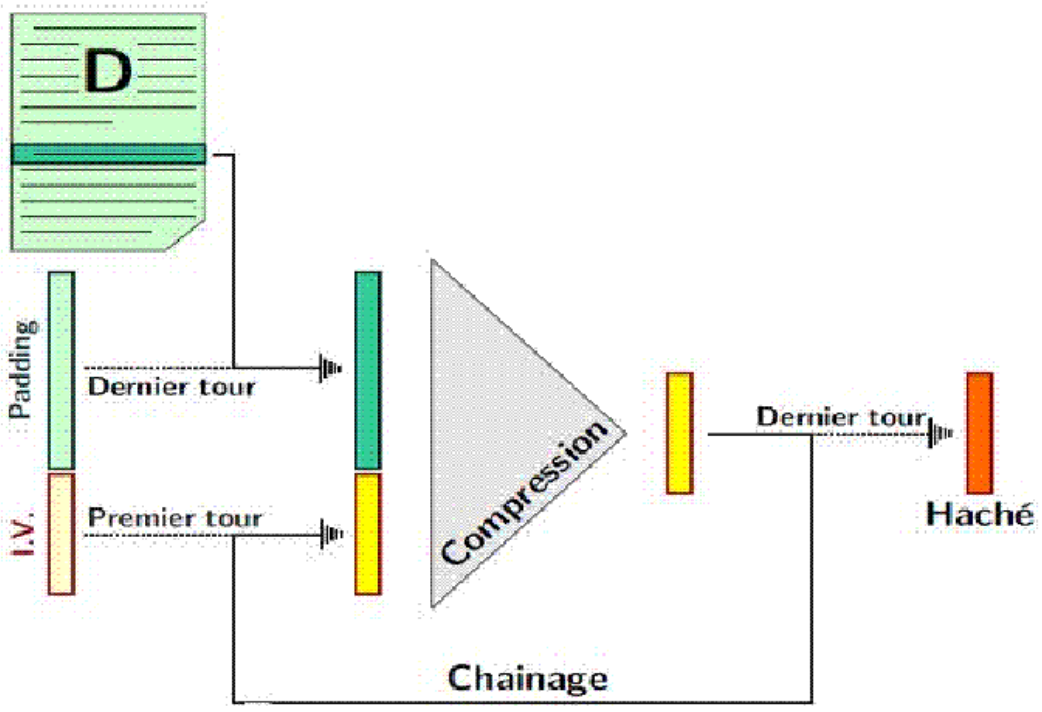

FIGURE 6.1 – Schéma général de construction d'une fonction de hachage de Damgřard et Merkle

le document va être lu par tranches de longueur fixe en entrée de la fonction de compression.

le chaînage consiste à réutiliser la sortie du tour $n-1$ de la fonction de compression en entrée du n^{ime} tour. Cela fait que la sortie du n^{ime} tour dépend de tout le début du document D et pas seulement de la dernière tranche lue.

l'initial value (I.V.) sert uniquement à remplacer les bits de chaînage lors du premier tour. Elle peut être choisie de n'importe quelle façon : par exemple entièrement nulle.

le padding permet de gérer des document de longueur quelconque. En effet la fonction de compression prend s bits d'entrée, dont r de chaînage. Si la longueur du document n'est pas multiple de $s-r$ on n'a pas un nombre de tours entier. Le padding consiste à ajouter des 0 à la fin du document, puis un denier bloc de longueur $s-r$ contenant la longueur initiale du document (afin d'éviter de créer des collisions avec le padding)pour atteindre une longueur adéquate.

Enfin, l'une des plus intéressantes propriétés de cette construction concerne la sécurité : il est prouvé que la sécurité d'une fonction de hachage construite avec ce schéma est au moins aussi élevée que la sécurité de la fonction de compression utilisée. Cela signifie qu'inverser le schéma global nécessite forcément d'inverser au moins une fois la fonction de compression, et trouver des collisions nécessite de trouver des collisions sur la fonction de compression. Cela permet donc de prouver la sécurité du schéma complet en ne prouvant que la sécurité de la fonction de compression.

En effet, on peut imaginer facilement que si on a une collision sur la fonction globale on a aussi une collision sur le dernier tour de la fonction, ou alors, si les deux documents sont égaux, sur celui d'avant...On peut ainsi prouver que si les deux documents sont différents, il y a nécessairement une collision sur un tour de la fonction de compression et on en déduit que si on sait trouver des collisions sur la fonction globale, on saura donc aussi le faire sur la fonction de compression.

6.3 Quelques fonctions connues

Les fonctions de hachage les plus utilisées de nos jours sont MD5 et SHA-1.Elles utilisent toutes les deux le schéma de chaînage de la figure 6-1 : MD5 une sortie de 128 bits pour une entrée de 128+512 et SHA-1 avec une sortie de 160 bits pour une entrée de 160+512.

MD5 (Message Digest 5) a été proposée par R. Rivest en 1992 [37] et est une évolution de MD4 [38] (pour laquelle des collisions ont été trouvées depuis [14]).Quelques attaques existent sur cette fonction [6, 13] mais elle est encore très utilisée pour des applications où la sécurité n'est pas primordiale. Elle est de nos jours essentiellement utilisée pour garantir l'authenticité de fichiers sur internet. Sa construction repose sur

le schéma de Davies-Meyer, c'est-à-dire que la fonction de compression est en fait une fonction de chiffrement de 128 bits vers 128, utilisant une clef de 512 bits.

SHA-1 est une évolution de SHA (Secure Hash Algorithm) publiée par le gouvernement américain en 1993 [30]. Sa construction repose sur celle de MD5, mais est plus sûre puisqu'elle utilise 160 bits. De ce fait c'est cette fonction qui est la plus utilisée dans les schémas de signature. Il en existe aussi trois autres versions : SHA-256, SHA-384 et SHA-512 hachant respectivement en 256, 384 et 512 bits. Une faiblesse présente dans SHA et mise à jour par F. Chabaud et A. Joux [9] a disparu dans chacune de ces versions.

Une grande variété d'autres fonctions de hachage existent mais sont beaucoup moins utilisées. Certaines font appel à des principes similaires [15, 27, 4], d'autres constructions cherchent, au contraire, à atteindre des buts différents : par exemple, la fonction décrite dans [18], en s'appuyant sur des problèmes de logarithmes discrets, permet d'obtenir une preuve de sécurité au détriment de la vitesse de hachage.

6.4 Une nouvelle fonction de compression

Matthieu Finiasz [17] a construit une fonction de hachage h rapide à sécurité prouvée, il a construit une fonction de compression f_c dont a prouvé la sécurité et qui est suffisamment rapide. Pour cela il faut non seulement qu'un tour de la fonction de compression soit rapide, mais aussi que la fonction compresse beaucoup (prenne beaucoup de bits d'entrée) afin de ne pas avoir à faire un trop grand nombre de tours au total.

6.4.1 Utilisation du chiffrement de Niederreiter

Le cryptosystème de Niederreiter [31] a, par rapport à la plupart des autres cryptosystèmes à clef publique, l'avantage d'avoir un chiffrement très rapide.

On peut donc imaginer utiliser une fonction similaire à f_c et ainsi bénéficier à la fois de la rapidité du système et de sa sécurité.Comme vu section 5-2.2, dans le système de Niederreiter le chiffrement se fait en utilisant une matrice de parité d'un code de Goppa. On convertit le message à chiffrer en un mot de longueur n et de poids w donné que l'on multiplie par la matrice de parité. Le résultat est un syndrome qui va servir de chiffré. La matrice de parité est supposée être indistinguable d'une matrice aléatoire et seule la connaissance de sa structure de code de Goppa et de la permutation utilisée pour la brouiller permet de déchiffrer le syndrome.

Pour utiliser cette fonction de chiffrement en tant que fonction de compression on n'a pas besoin de connaître la trappe (au contraire, on veut même être certain que personne ne peut inverser la fonction de compression) et on peut donc prendre une véritable matrice aléatoire $\mathcal{H}$ de taille $r \times n$. La fonction de compression se décompose alors en deux étapes :

1. convertir les $s = log_2(\binom{n}{w})$ bits d'entrée en un mot de longueur n et poids w.
2. multiplier ce mot par la matrice $\mathcal{H}$ pour obtenir un syndrome de longueur r.

6.4.2 Sécurité d'une telle fonction de compression

Pour une fonction de hachage, la sécurité se mesure en terme de résistance à l'inversion et à la recherche de collisions. L'inversion consiste à retrouver une entrée de la fonction de compression correspondant à une sortie donnée. Dans notre cas il s'agit, à partir d'un syndrome S donné de longueur r, de retrouver un mot e de poids w tel que $\mathcal{H} \times e^T = S$. C'est exactement le problème de syndrome decoding (SD) .

Pour ce qui est de la recherche de collisions, il va s'agir de trouver deux mots e_1 et e_2 de poids w ayant le même syndrome. Cela revient à trouver un mot $e = e_1 + e_2$ de poids $2w$ ayant un syndrome nul. Ici aussi on se retrouve exactement avec une instance du problème SD à résoudre.

D'un point de vue théorique, la sécurité de la fonction de compression (et donc de la fonction de hachage construite avec) repose sur un problème NP complet bien identifié : le problème SD [4]. Cependant cela ne suffit pas pour garantir la sécurité du schéma, il faut aussi choisir des paramètres tels que les instances du problème SD qu'un attaquant aurait à résoudre soient suffisamment difficiles pour être considérées comme calculatoirement impossibles.

a) Pour l'inversion

On est dans le cas le plus général du problème SD, c'est-à-dire qu'on s'attaque à des instances quelconques, sans aucune particularité : la matrice est bien quelconque, le syndrome aussi. Les meilleures attaques pour l'inversion seront donc les mêmes attaques que sur le problème SD général.

b) Pour les collisions

On ne s'intéresse qu'à la résolution du problème SD avec des syndromes nuls. C'est donc à un sous-ensemble des instances du problème que l'on s'attaque et des algorithmes plus spécifiques pourraient donc exister. Cependant, on peut en fait ramener n'importe quelle instance au cas où le syndrome est nul et ces instances ne sont pas plus faciles que les autres. Encore une fois on sera donc amené à utiliser des attaques génériques.

Conclusion

On a introduit dans ce travail une application de la théorie des codes correcteurs d'erreurs en cryptographie. Le cryptosystème classique à base de codes correcteurs est le système de McEliece utilisant les codes de Goppa. Comparativement aux systèmes issus d'autres primitives cryptographiques, le chiffrement et le déchiffrement par McEliece sont rapides. Mais elle nécessite certainement encore quelques petits ajustements avant d'être réellement utilisable en pratique. Car on a pu recensé trois inconvénients pour le crypto système de McEliece :
1. La taille de la clef publique $(\mathcal{G}_{pub})$ est grande. Ceci posera certainement des problèmes d'exécution.
2. Le message chiffré est plus long que le message clair. Cette augmentation de la largeur du message chiffré rend le système plus sensible aux erreurs de transmission.
3. Le crypto système n'est employé pour la signature ou l'authentification parce que l'algorithme de chiffrage n'est pas linéaire et tout l'algorithme est vraiment asymétrique.

La sécurité d'un système s'évalue grâce au coût des meilleures attaques, mais l'effort fourni pour essayer d'en trouver de meilleures est aussi une part importante de cette sécurité.

La version de McEliece utilisant des codes de Goppa semble parfaitement sûre d'un point de vue cryptographique. Cependant, elle n'est pas utilisée en pratique, principalement car la taille de sa clé publique est beaucoup plus importante que ce que l'on sait faire avec des systèmes issus d'autres domaines.

Si l'on veut diversifier notre panel de primitives cryptographiques, il faut rendre les systèmes asymétriques basés sur des codes compétitifs vis-à-vis d'autres cryptosystèmes. Pour cela, il est important de trouver un moyen de réduire la taille de leur clé publique.

La fonction de hachage est le travail pour lequel l'étude de la sécurité semble la moins achevée. En effet, la sécurité du système repose directement, et de façon relativement claire, sur un problème bien précis, voisin du célèbre problème de syndrome decoding (SD). Cependant, même si ce problème proche est lui aussi NP-complet, l'étude de la complexité des meilleures attaques en est encore à son début.

Finalement, on peut déduire que la cryptographie basée sur les codes correcteurs d'erreurs constitue une alternative aux systèmes de chiffrement à clé publique classiques à ne pas négliger.

Bibliographie

[1] E. R. Berlekamp. Algebraic Coding Theory. Aegen Park Press, 1968.

[2] E. R. Berlekamp and L. R. Welch. Error correction for algebraic block codes. US Patent 4 633 470, 1986.

[3] E. R. Berlekamp. Goppa codes. IEEE Transactions on Information Theory, 19(5) :590-592, September 1973.

[4] E. R. Berlekamp, R. J. McEliece, and H. C. van Tilborg. On the inherent intractability of certain coding problems. IEEE Transactions on Information Theory, 24(3), May 1978.

[5] B. Biswas and N. Sendrier. McEliece Cryptosystem Implementation : Theory and Practice. In Post-Quantum Cryptography, number 5299 in LNCS, pages 47-62. Springer-Verlag, 2008.

[6] B. den Boer and A. Bosselaers. Collisions for the compression function of MD5. In T. Helleseth, editor, Advances in Cryptology - EUROCRYPT'93, volume 765 of Lecture Notes in Computer Science, pages 293-304. Springer-Verlag, 1994.

[7] K. A. Bush. Orthogonal arrays of index unity. Annals of Mathematical Statistics, 23 :426-434, 1952.

[8] A. Canteaut and N. Sendrier. Cryptanalysis of the original McEliece Cryptosystem. In Advances in Cryptology - ASIACRYPT'98, number 1514 in LNCS, pages 187-199. Springer-Verlag, 1998.

[9] F. Chabaud and A. Joux. Differential collisions in SHA-0. In H. Krawczyk, editor, Advances in Cryptology - CRYPTO'98, volume 1462 of Lecture Notes in Computer Science, pages 56-71. Springer-Verlag,1998.

[10] N. Courtois, M. Finiasz, and N. Sendrier. How to achieve a McEliece-based Digital Signature Scheme. Technical report, INRIA, http ://eprint.iacr.org/2001/010/, 2001.

[11] I.B. Damgard. A design principle for hash functions. In Gilles Brassard, editor, Advances in Cryptology - Crypto' 89, Lecture Notes in Computer Science, pages 416-426. Springer-Verlag, 1989.

[12] Y. X. Li, R. H. Deng, and X. M. Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. IEEE Transactions on Information Theory, 40(1) :271-273, 1994.

[13] H. Dobbertin. The status of MD5 after a recent attack. Cryptobytes, 2(2) :1-6, 1996.

[14] H. Dobbertin. Cryptanalysis of MD4. Journal of Cryptology, 11(4) :253-271, 1998.

[15] H. Dobbertin, A. Bosselaers, and B. Preneel. RIPEMD-160 : a strengthened version of RIPEMD. In D. Gollmann, editor, Advances in Cryptology - FSE 96, volume 1039 of Lecture Notes in Computer Science, pages 71-82. Springer-Verlag, 1996.

[16] Douglas Stinson,tradiction de Serge Vaudenay,Gildas Avoine et Pascal Junod,Cryptographie Théorie et pratique,2nd edition.

[17] M. Finiasz. Nouvelles Constructions utilisant des Codes Correcteurs d'Erreurs en Cryptographie à Clef Publique. PhD thesis, Ecole Polytechnique,Oct. 2004.

[18] J. K. Gibson. Discrete logarithm hash function that is collision free and one way. Computers and Digital Techniques, IEE Proceedings E, 138(6) :407-410, November 1991.

[19] V. D. Goppa. A new class of linear error-correcting codes. Probl. Inform. Transm., 6 :207-212, 1970.

[20] V. D. Goppa. Codes on algebraic curves. Soviet Math. Dokl., 24 :170-172, 1981.

[21] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and Algebraic-Geometric codes. IEEE Transactions on Information Theory, 45(6) :1757-1767, September 1999.

[22] H. Janwa and O. Moreno. Public Key Cryptosystems Using Algebraic- Geometric Codes. Designs, Codes and Cryptography, 8(3), 1996.

[23] T. Johansson and F. Jonsson. Fast correlation attacks based on turbo code techniques. In M. J. Wiener, editor, Advances in Cryptology - CRYPTO'99, volume 1666 of Lecture Notes in Computer Science, pages 181-197. Springer-Verlag, 1999.

[24] T. Johansson and F. Jonsson. Improved fast correlation attacks on stream ciphers via convolutional codes. In J. Stern, editor, Advances in Cryptology - EUROCRYPT'99, volume 1592 of Lecture Notes in Computer Science, pages 347-362. Springer-Verlag, 1999.

25

[25] F. J. MacWilliams and N. J. A. Sloane. The Theory of Error-Correcting Codes. North-Holland, 1977. 26

[26] J. L. Massey. Shift-register synthesis and BCH decoding. IEEE Transactions on Information Theory, 15(1) :122-127, January 1969. 27

[27] J. L. Massey. SAFER K-64 : A byte-oriented block-ciphering algorithm. In R. J. Anderson, editor, Advances in Cryptology - FSE 93, volume 809 of Lecture Notes in Computer Science, pages 1-17. Springer-Verlag, 1994. 28

[28] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. DSN Prog. Rep., Jet Prop. Lab., California Inst. Technol., Pasadena, CA, pages 114-116, January 1978. 29

[29] R. C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, Advances in Cryptology - Crypto' 89, Lecture Notes in Computer Science. Springer-Verlag, 1989. 30

[30] National Insitute of Standards and Technology. FIPS Publication 180 : Secure Hash Standard, 1993. 31

[31] H. Niederreiter. Knapsack-type crytosystems and algebraic coding theory. Prob. Contr. Inform. Theory, 15(2) :157-166, 1986. 32

[32] Ayoub Otmani "Cryptographie fondée sur la théorie des codes" Ayoub.Otmani@info.unicaen.fr 33

[33] A. A. Pantchichkine,Mathématiques des codes correcteurs d'erreurs (Master-2 de mathématiques (M2P), "Cryptologie, Sécurité et Codage d'Information",2004/2005, Module 506a),Institut Fourier, B.P.74, 38402 St.-Martin d'Hères, FRANCE. 34

[34] N. J. Patterson. The algebraic decoding of Goppa codes. IEEE Transactions on Information Theory, 21(2) :203-207, March 1975. 35

[35] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. Journal of the SIAM, 8(2) :300-304, June 1960. 36

[36] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2) :120-126, February 1978. 37

[37] R. L. Rivest. The MD5 message-digest algorithm. Request for Comments (RFC) 1321, April 1992. Internet Activities Board, Internet Privacy Task Force. 38

[38] R. L. Rivest. The MD4 message digest algorithm. In A.J. Menezes and S.A. Vanstone, editors, Advances in Cryptology - CRYPTO '90, Lecture Notes in Computer Science, pages 303-311. Springer-Verlag, 1991. 39

[39] V. M. Sidel'nikov and S. O. Shestakov. On cryptosystems based on generalized Reed-Solomon codes. Discrete Mathematics, 4(3) :57-63,1992. en russe. 40

[40] M. Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. Journal of Complexity, 13(1) :180-193, March 1997. 41

[41] M. A. Tsfasman, S. G. Vląadut, and T. Zink. Modular curves, Shimura curves, and Goppa codes, better than the Varshamov-Gilbert bound. Math. Nachrichten, 109 :21-28, 1982.