Codes correcteurs et cryptosystème de Mc Eliece

Auclair Pierre

1er mai 2014

Table des matières

Introduction

De nos jours, l'un des rôles principaux de l'informatique est la communication, aussi bien entre particuliers sur le réseau internet qu'au niveau de l'ingénierie militaire et spatiale. Néanmoins aucun des canaux utilisés pour transmettre l'information n'est sur à 100% et des erreurs se produisent avec une probabilité non négligeable. C'est pourquoi une nouvelle branche de l'informatique : la théorie des codes, a émergée dans les années 1950 concevant des codes permettant de détecter et même de corriger les erreurs induites par les canaux de transmission.

Un exemple plus familier d'un tel code est le langage humain, celui-ci utilise deux procédés réutilisés par les codes plus informatiques. Tout d'abord, la répétition : lorsqu'on discute, tous les mots de la phrase ne sont pas nécessaires pour rendre compte de son sens. Il y a donc des informations redondantes dans la phrase, et on pourrait transposer cela en informatique en décidant d'envoyer plusieurs fois le même message. Ce n'est cependant pas la méthode utilisée à cause de son coût. Un autre procédé est celui de la distinguabilité des mots entre eux. Deux mots possèdes en général assez de syllabes différentes pour que même mal prononcés on puisse les distinguer. C'est plutôt ce procédé de séparer les mots les uns des autres qui est utilisé en informatique et que nous formaliserons.

Dans ce rapport nous nous intéresserons principalement aux codes de Goppa et à une de leurs applications en cryptographie dans le cryptosystème de Mc Eliece.

Chapitre 1

Codes correcteurs

1.1 Définition d'un code correcteur

Dans cette section nous allons formaliser l'idée de codes correcteurs telle qu'ébauchée dans l'introduction. L'idée est bien sur d'enrichir via le codage un message de taille k avec de la redondance pour obtenir un nouveau message de taille n. Il faut donc $k < n$.

Définition 1. *[Pan04] Un code en bloc* $\mathbf{C}$ *est l'image d'une application injective* $\mathbf{f}$ *de* $\mathbb{F}_q^k$ *dans* $\mathbb{F}_q^n$ *. Ce code est dit linéaire si* f *est une application linéaire.*

$$\mathbf{f} : \mathbb{F}_q^k \to \mathbb{F}_q^n$$

Nous avons aussi soulevé l'idée que pour mieux reconnaître le message d'origine, il fallait séparer les mots de code entre eux. Pour cela nous définissons une notion de distance.

Définition 2. *La distance de Hamming est l'application* $\mathbf{d}$ *de* $\mathbb{F}_q^n \times \mathbb{F}_q^n$ *dans* $\mathbb{N}$ *définie par :*

$$\mathbf{d} : (a,b) \in \mathbb{F}_q^n \times \mathbb{F}_q^n \mapsto card(i \in \mathbb{N}/a_i \neq b_i)$$

Maintenant que nous avons l'outil pour constater la séparation des mots de code, nous introduisons la notion de distance minimale entre deux mots. Cette distance donne un ensemble de boules chacune centrée sur un mot de code qui ne s'intersectent pas deux à deux.

Définition 3. *On note* t *la capacité de correction d'un code correcteur, par définition :*

$$t = \max_{r \in \mathbb{N}} \bigcap_{x \in \mathbf{C}} B(x,r) = \emptyset$$

De manière évidente on a l'inégalité : $2t + 1 \leq d$

Nous définissons enfin une application permettant de retrouver le message d'origine.

Définition 4. *On appelle un décodage l'application* $\mathbf{D} : \mathbb{F}_q^n \to \mathbb{F}_q^k$ *telle que* $\mathbf{D} \circ \mathbf{f} = Id_{F_q^k}$. $\mathbf{D}$ *est de vraisemblance maximale si :*

$$\forall x \in \mathbf{C}, \mathbf{D}(B(x,t)) = x$$

1.2 Utilisation des codes correcteurs

Nos codes correcteurs sont pour l'instant des ensembles de mots d'un espace de dimension plus grande que l'espace des messages. L'intérêt d'utiliser des applications linéaires est de pouvoir définir l'ensemble $\mathbf{C}$ avec un minimum d'informations. La seule donnée de l'image des vecteurs de base définit entièrement $\mathbf{f}$ et $\mathbf{C}$, donc le codage.

Définition 5. *On appelle G matrice génératrice du code* $\mathbf{C}$ *la matrice de* $\mathbf{f}$ *dans la base canonique.*

$$G \in M_{n,k}(\mathbb{F}_q), G = Mat(\mathbf{f})$$

Définition 6. *On appelle H matrice de parité du code* $\mathbf{C}$ *une matrice telle que :*

$$H \in M_{n-k,n}(\mathbb{F}_q), Ker(H) = \mathbf{C}$$

L'application $x \in \mathbb{F}^n \to Hx$ *s'appelle le syndrôme de* x *.*

Nous avons défini le codage de manière linéaire, il est impossible de faire de même pour le décodage. Nous allons donc utiliser la matrice de parité pour introduire la notion de syndrome qui va permettre le décodage.

Propriété 1. *A la réception d'un message de* $\mathbb{F}^n$ *, en considérant que le nombre d'erreurs est inférieur a t, donc que le vecteur d'erreur* ϵ *a un poids inferieur ou egal a t, on a :* $\forall m \in \mathbb{F}^n, \exists!(x,\epsilon) \in \mathbf{C} \times \mathbb{F}^n$ *tels que :*

$$\left\{ \begin{array}{rcl} m & = & x + \epsilon \\ \mathbf{d}(\epsilon) & \leq & t \\ Hm & = & H\epsilon \end{array} \right.$$

Le principe est de determiner ϵ . On retrouve ainsi le mot de code d'origine $x = m - \epsilon$

Propriété 2. $\forall m \in \mathbb{F}^n$ *, son syndrôme est caratéristique de l'erreur, dans l'hypothèse d'un maximum de t erreurs. Supposons deux vecteurs d'erreurs avec le même syndrome :*

$$H\epsilon = H\eta \Rightarrow H(\epsilon - \eta) = 0 \Rightarrow \epsilon - \eta \in \mathbf{C}$$

D'après l'inégalité triangulaire, $d(\epsilon - \eta) \leq 2t < d$ *donc* $\epsilon - \eta = 0$

Néanmoins cette méthode est relativement limitée en l'état. Pour un code linéaire quelconque, la reconnaissance du syndrôme est un problème jugé difficile. L'enjeu de la théorie des codes va être de construire des structures de codes rendant le problème plus simple. Dans la suite de ce TIPE nous nous intéresserons aux codes de Goppa dont l'usage principal sera le cryptosystème de Mc Eliece étudié dans la partie 3.

Chapitre 2

Codes de Goppa

2.1 Définition d'un code de Goppa

Les codes de Goppa classiques sont des codes linéaires dont on connaît une borne inférieure de la distance minimale. Ce sont les codes utilisés dans le cryptosystème de McEliece qui va nous intéresser dans une deuxième partie.

Définition 7. *On définit un code de Goppa* $\mathbb{F}_q^k \to \mathbb{F}_q^n$ *par son support* L *et un polynôme* g *définis par :*

$$\begin{cases} g & \in & \mathbb{F}_{q^m}[X] \text{ unitaire irréductible sur } \mathbb{F}_{q^m} \text{ de degré } t \\ L & = & (\alpha_1, .., \alpha_n) \text{ d'éléments de } \mathbb{F}_{q^m} \end{cases}$$

A partir de ces élements, on définit le code de Goppa $\mathbf{C}$ *par son syndrome* $\mathbf{S}$ *comme :*

$$\mathbf{C} = \{y = (y_1, .., y_n) \in \mathbb{F}_q^n / \mathbf{S}(y) = \sum_{i=1}^{n} \frac{y_i}{x - \alpha_i} = 0 \ mod \ g(x)\}$$

Propriété 3. *Le code de Goppa ainsi défini est de distance minimale* $\mathbf{d} \geq t+1$ *. En effet en supposant qu'il existe un message* $y = (y_1, .., y_n) \in \mathbf{C}$ *avec* t *ou moins coordonnées non nulles :*

$$\sum_{i=1}^{n} \frac{y_i}{x - \alpha_i} = \frac{a(x)}{b(x)} = 0 \ mod \ g(x) \ fraction \ irréductible$$

Donc $g \mid a$*, ainsi* a *de degré supérieur ou égal à* t*. Or* a *de degré = nombre de coordonnées non nulles de* y *- 1* $\leq t-1$

Par l'absurde le code de Goppa a une distance minimale $\mathbf{d} \geq t+1$ *.*

2.2 Construction de la matrice de parité

Nous souhaiterions donner à notre code de Goppa défini par son syndrome une structure plus visiblement linéaire. Nous allons construire sa matrice de parité.

Définition 8. *[Pan04] Nous allons définir une famille de polynômes* $\mathbf{f}_i$ *inverses des* $(x-\alpha_i)$ *. Cela est possible car* g *étant irréductible,* $\mathbb{F}[X]/g$ *est un corps :*

$$\mathbf{f}_i(x)(x-\alpha_i) = 1 \ mod \ g(x)$$

On a donc que :

$$\forall y \in \mathbb{F}^n \ \mathbf{S}(y) = \sum_{i=1}^{n} \frac{y_i}{x-\alpha_i} = \sum_{i=1}^{n} \mathbf{f}_i(x) y_i$$

Propriété 4. *Évaluons les* $\mathbf{f}_i$ *sous forme de fraction rationnelle :*

$$\mathbf{f}_i(x) = \frac{1}{g(\alpha_i)} \frac{g(x)-g(\alpha_i)}{x-\alpha_i}$$

Maintenant tentons de les rendre polynômiaux, pour cela utilisons une formule de factorisation bien connue :

$$x^j - \alpha_i^j = (x-\alpha_i) \sum_{k=0}^{j-1} x^k \alpha_i^{j-1-k}$$

On en déduit donc que :

$$\mathbf{f}_i(x) = \frac{1}{g(\alpha_i)} \sum_{j=1}^{t} g_j \sum_{k=0}^{j-1} x^k \alpha_i^{j-1-k} = \frac{1}{g(\alpha_i)} \sum_{k=0}^{t-1} x^k \sum_{j=k+1}^{t} g_j \alpha_i^{j-1-k}$$

Maintenant qu'on a montré leur existence et qu'on les a calculés, nous utilisons les fi pour construire la matrice de parité. Pour cela nous allons identifier un polynôme comme un vecteur de degré t.

Définition 9. *On a donc des polynômes* $(fi)_{1 \leq i \leq n}$ *de degrés t-1 que l'on va identifier comme des vecteurs colonnes de dimension t-1.*

$$\forall y \in \mathbb{F}^n \ \mathbf{S}(y) = \sum_{i=1}^{n} \frac{y_i}{x-\alpha_i} = \sum_{i=1}^{n} \mathbf{f}_i(x) y_i$$

$$\forall y \in \mathbb{F}^n \ \mathbf{S}(y) = (f_1(x), ..., f_n(x)) y$$

On se retrouve donc avec une matrice de calcul de syndrôme :

$$\begin{pmatrix} \frac{1}{g(\alpha_1)} g_t & \frac{1}{g(\alpha_2)} g_t & \cdots & \frac{1}{g(\alpha_n)} g_t \\ \frac{1}{g(\alpha_1)} (g_{t-1} + g_t \alpha_1) & \frac{1}{g(\alpha_2)} (g_{t-1} + g_t \alpha_2) & \cdots & \frac{1}{g(\alpha_n)} (g_{t-1} + g_t \alpha_n) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{1}{g(\alpha_1)} (g_1 + ... + g_t \alpha_1^{t-1}) & \frac{1}{g(\alpha_2)} (g_1 + ... + g_t \alpha_1^{t-1}) & \cdots & \frac{1}{g(\alpha_n)} (g_1 + ... + g_t \alpha_1^{t-1}) \end{pmatrix}$$

Pour construire la matrice de parité, comme seul le noyau nous intéresse, nous allons simplifier cette matrice et poser H matrice de parité égale à :

$$\begin{pmatrix} \frac{1}{g(\alpha_1)} & \frac{1}{g(\alpha_2)} & \cdots & \frac{1}{g(\alpha_n)} \\ \frac{1}{g(\alpha_1)} g_t \alpha_1 & \frac{1}{g(\alpha_2)} g_t \alpha_2 & \cdots & \frac{1}{g(\alpha_n)} g_t \alpha_n \\ \vdots & \vdots & \vdots & \vdots \\ \frac{1}{g(\alpha_1)} g_t \alpha_1^{t-1} & \frac{1}{g(\alpha_2)} g_t \alpha_2^{t-1} & \cdots & \frac{1}{g(\alpha_n)} g_t \alpha_n^{t-1} \end{pmatrix}$$

Maintenant que nous possédons sa matrice de parité, le codage est possible. La construction de G matrice génératrice de notre code de Goppa s'effectue simplement. N'étant pas spécifique aux codes de Goppa, l'algorithme utilisé est décrit plus tard.

2.3 Equation clef

Nous sommes dorénavant dotés des outils algébriques nécessaires au codage de nos messages. Nous allons maintenant étudier comment les décoder.

Définition 10. *[Fin04] Nous allons introduire le polynôme localisateur d'erreurs σ, qui par sa connaissance permet de déterminer l'emplacement des erreurs dans le message.*

$$\sigma_{y+\epsilon}(x) = \prod_{\epsilon_i \neq 0} (x - \alpha_i) \ mod \ g(x)$$

L'enjeu du décodage va être de déterminer le polynôme localisateur d'erreurs pour un message donné et d'en déterminer les racines.

Définition 11. *Nous allons définir le polynôme évaluateur d'erreurs ω défini par :*

$$\omega(x) = \sigma(x) S(x) \ mod \ g(x)$$

L'intérêt de ce second polynôme est de mettre en évidence l'équation clef du décodage :

$$\omega(x) = \sigma(x) S(x) + k(x) g(x)$$

2.4 Résolution de l'équation clef

La résolution de cette équation clef est la détermination de σ. Pour cela il existe plusieurs techniques, nous étudierons celle utilisant l'algorithme d'Euclide. L'algorithme d'Euclide étendu tel qu'utilisé est redonné plus tard.

Propriété 5. *[CL] On a unicité de la résolution de l'équation clef pour les polynômes σ et ω de degrés $deg(\sigma) \leq \frac{t}{2}$ et $deg(\omega) < \frac{t}{2}$ à un scalaire près. On suppose de plus que les deux polynômes sont de degrés minimum, c'est à dire* *premiers entre eux. En effet, soient* (σ_1, ω_1) *et* (σ_2, ω_2) *deux solutions de degrés inférieurs à* $\frac{t}{2}$*. On a :*

$$\omega_1(x) = \sigma_1(x)S(x) \ mod \ g(x) \ , \ \omega_2(x) = \sigma_2(x)S(x) \ mod \ g(x)$$

$$\omega_1(x)\sigma_2(x) = \sigma_1(x)S(x)\sigma_2(x) \ mod \ g(x) \ , \ \omega_2(x)\sigma_1(x) = \sigma_2(x)S(x)\sigma_1(x) \ mod \ g(x)$$

Ainsi :

$$\omega_1(x)\sigma_2(x) - \omega_2(x)\sigma_1(x) = 0 \ mod \ g(x)$$

Or le degré du polynôme du membre de gauche est de degré $< t-1$*. Ce polynôme est donc nul et* $\omega_1(x)\sigma_2(x) = \omega_2(x)\sigma_1(x)$*. Avec l'hypothèse de degrés minimums, on a égalité des couples à un scalaire près.*

Chapitre 3

Cryptosystème de Mc Eliece

Le cryptosystème de Mc Eliece mis au point en 1978 est est l'un des premiers systèmes de cryptage à clef publique. Son principe repose sur l'utilisation de codes correcteurs d'erreurs, plus particulièrement des codes de Goppa. Bien que l'utilisation de codes correcteurs linéaires facilement décodables le rende extremment rapide, ce système a été penalisé par la taille importante de ses clefs publiques. Néanmoins, des projets comme celui de l'OTAN [1] remettent au jour ce système pouvant resister à l'avènement d'ordinateurs quantiques. C'est ce cryptosytème que nous avons étudié.

3.1 Les clefs

Un cryptage asymétrique repose sur un principe simple : tout le monde a accès à une clef publique permettant le chiffrement et seul le créateur de cette clef possède la clef privée permettant le déchiffrement.

Pour ce faire, on commence par créer G la matrice génératrice d'un code de Goppa de paramètres :

- k dimension du bloc de message à coder
- n dimension du bloc de message à envoyer
- t la capacité de correction

La clef privée représente l'ensemble :

- G matrice génératrice (n,k)
- P une matrice (n,n) de permutation
- Q une matrice (k,k) inversible

La clef publique partagé est l'ensemble :

- G'=PGQ
- La capacité de correction t

1. Projet OTAN : SPS 984520 Secure implementation of post-quantum cryptography

3.2 Le chiffrement

Soit $m \in \mathbf{F}_2^k$ le message à chiffrer, l'envoyeur calcule l'image du message par le code G puis ajoute un vecteur $e \in \mathbf{F}_2^n$ de poids inférieur à t.

$$x = G'm + e = PGQm + e$$

C'est ce vecteur $x \in \mathbf{F}_2^n$ qui correspond au message chiffré.

3.3 Le déchiffrement

Soit $x \in \mathbf{F}_2^n$ le message chiffré reçu, on calcule :

$$P^{-1}x = GQm + P^{-1}e$$

Le vecteur $GQm \in \mathbf{F}_2^n$ est un mot du code de Goppa et donc $P^{-1}e \in \mathbf{F}_2^n$ est un vecteur d'erreur de poids inférieur à t car P matrice de permutation. On utilise notre algorithme efficace de décodage du code de Goppa, on obtient donc Qm , connaissant Q inversible on a :

$$m = Q^{-1}Qm$$

D'où le déchiffrement.

3.4 Le principe de sécurité

Le cryptosystème de Mc Eliece repose sur 2 problèmes difficiles mathématiques.

Premièrement, la difficulté à partir d'une matrice $G' = PGQ$ de retrouver la matrice G en temps polynômial. Secondement, de décoder un code correcteur qui parait aléatoire, c'est le problème de reconnaissance du syndrome que nous avons évité en utilisant un code de Goppa dont on connait la strucure et donc un algorithme efficace de décodage.

Chapitre 4

Implémentation des structures du code et du cryptosystème

4.1 Structures algébriques

4.1.1 Les matrices

```

  def __init__(self,nbligne,nbcolonne,tableau):
    """Definition de la matrice avec gestion du cas erreur de taille"""
    try:
      if len(tableau) != nbligne * nbcolonne :
        raise IndexError("Erreur de taille de la matrice durant l'initialisation
      self.nbligne=nbligne
      self.nbcolonne=nbcolonne
      self.tableau=tableau[::]
    except IndexError as ex:
      print(ex)
      print [nbligne,nbcolonne,tableau]
    except AttributeError as ex:
      print(ex)
```

Les matrices sont représentées par un tableau linéaire dont le comportement est caractérisé par le nombre de lignes et le nombre de colonnes. La flexibilité de Python nous permet d'utiliser cette classe aussi bien pour des réels que pour des éléments de corps fini.

4.1.2 Les polynômes

```
  """Classe de polynomes"""
```

```python

    def __init__(self, liste):
        """Initialisation de l'objet"""
        try:
            self.liste=liste[::]
```

Les polynômes à coefficients aussi bien dans $\mathbb{R}$ que dans des corps finis.

4.1.3 Les élements dans un corps de Galois $\mathbb{F}_{2^m}$

```python

    def __init__(self,n,p=2):
        """Initialisation"""
        diff = d(n,p)
        while diff>= 0:
            n ^= p<<diff
```

Un élement d'un corps de Galois sur $\mathbb{F}_{2^m}$ est un polynôme de $\mathbb{Z}/2\mathbb{Z}[X]/P$ avec P un polynôme irréductible de $\mathbb{Z}/2\mathbb{Z}[X]$. Notre implémentation repose sur la compréhension du binaire par l'interpréteur Python : un entier est stocké sous sa forme binaire sur laquelle on peut effectuer des opérations binaires (xor, and, or et décalages de bits). Grâce à cette caractéristique de Python on peut utiliser des entiers comme des polynômes de $\mathbb{Z}/2\mathbb{Z}$. On obtient une structure d'éléments de $\mathbb{F}_{2^m}$ très rapide.

Chapitre 5

Principaux algorithmes

5.1 Algorithme de Strassen

5.2 Pivots de Gauss et applications

5.3 Algorithme de Berlekamp

5.4 Algorithme de décodage

Bibliographie

[CL] Antoine Chambert-Loir. *Codes de Goppa - Préparation à l'agrégation - option Calcul formel.*

[Fin04] Matthieu Finiasz. *Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie à clef publique.* PhD thesis, Polytechnique, 2004.

[Pan04] A.A Pantchichkine. *Mathématiques des codes correcteurs d'erreurs.* Institut Fourier, 2004.