Algorithme de Chiffrement basé sur les codes de correcteurs d'erreurs

5.1 Historique

Le premier cryptosystème basé sur les codes est le cryptosystème de McEliece [66], qui proposait d'utiliser des codes de Goppa. Suite à cela, plusieurs familles de code ont été suggérées pour remplacer les codes de Goppa dans ce schéma : les codes de Reed–Solomon généralisés (GRS) [67] ou bien des sous-codes de ces derniers , des codes de Reed–Muller , des codes algébriques géométriques , des codes LDPC , des codes MDPC ou plus récemment des codes convolutifs.
Certains de ces schémas permettent d'obtenir des clés publiques plus petites que celle du système original tout en vraisemblablement conservant le même niveau de sécurité contre les algorithmes de décodage génériques. Cependant, pour plusieurs des schémas susmentionnés, il a été montré qu'une description du code sous-jacent aidant au décodage peut être obtenue, cassant par là-même le schéma. Cela s'est produit pour les codes de Reed–Solomon généralisés (GRS) dans [68] et pour leurs sous-codes dans [69]. Dans ce cas, l'attaque retrouve entièrement et en temps polynomial la structure du code à partir de la clé publique. Les codes de Reed–Muller ont également été attaqué, mais cette fois, l'algorithme trouvant la description du code permuté a une complexité sous-exponentielle , ce qui est suffisant pour casser les paramètres proposés dans [67] mais qui ne casse pas complètement le schéma. Les systèmes basés sur les codes de géométrie algébrique sont également cassé en temps polynomial mais uniquement pour les courbes hyperelliptiques de faible genre [31]. Un schéma basé sur des codes LDPC [3] a été attaqué dans [57] (le nouveau schéma présenté dans [2] semble insensible à ce genre d'attaque). Deux variantes [1] du schéma basé sur les GRS supposées résister à l'attaque de [68] ont été cassées (respectivement [34] et [24]) par une approche liée au distingueur des codes de Goppa proposé dans [28]. Le cryptosystème de McEliece d'origine reste lui intact. Une modification a été apporté dans [9, 53], utilisant les versions quasi-cycliques ou quasidyadiques des codes de

Goppa (ou plus généralement des codes alternant dans [9]) afin de réduire significativement la taille de la clé publique. Cependant, il a été montré dans [30, 70] que la structure de ces codes permet de réduire radicalement le nombre d'inconnus de l'attaque algébrique. La plupart des schémas proposés dans [9, 53] ont été cassés par cette approche. Ce genre d'attaque a une complexité exponentielle et peut être contrecarré en choisissant de petits blocs cycliques ou dyadiques afin d'augmenter le nombre d'inconnues du système algébrique. Lorsque le rendement du code de Goppa est proche de 1 (tel que dans le schéma de signature CFS [22]), il a été montré dans [29] qu'il était possible de distinguer la clé publique d'une clé aléatoire. Cela invalide les preuves de sécurité des schémas utilisant des codes de rendement proche de 1 puisque tous reposent sur l'hypothèse d'indistinguabilité des codes de Goppa.

5.2 Codes de Goppa et difficulté algorithmique

Nous avons vu section 3.3 que la difficulté algorithmique des problémes de décodage sur un code aléatoire est, dans le cas général, bien établie. Pour un code de longueur n, de dimension k et de matrice de parité G aléatoire , si les paramètres n et k sont correctement choisis, la fonction $f \rightarrow mG + e$ où $e \leftarrow W_{n,t}$ est donc à sens unique.Cependant, pour un usage dans le cadre d'un schéma de chiffrement il est nécessaire d'introduire une trappe permettant au destinataire de décoder f(m) afin de recouvrer l'information transmise m, ce qui est impossible en utilisant un code aléatoire. Les codes de Goppa permettent d'atteindre cet objectif.

5.2.1 Codes de Goppa

Les codes de Goppa ont été introduits par V. D. Goppa en 1970 [70]. Initialement étudiés pour leurs propriétés de codes correcteurs d'erreurs ce qui permit l'apparition d'algorithmes de décodage efficaces.Ils ont été ensuite étudiés pour leurs propriétés cryptographiques avec l'apparition du cryptosystème de McEliece.
Les codes de Goppa sont des codes linéaires sur un corps fini F_q, mais leur construction passe par une extension ${F_q}^m$: ce sont des codes trace d'un code de Reed-Solomon généralisé.
Un code de Goppa $\Gamma(g, L)$ est défini par un polynome g de degré t à coefficients dans ${F_q}^m$ et son support $L = \{\alpha_0, ..\alpha_{n-1}\}$ de n éléments de ${F_q}^m$ qui ne sont pas racines de g.

5.2.2 Les codes de Goppa binaires

Les codes de Goppa binaires correspondent au cas particulier où l'on choisit q = 2. La matrice de parité H du code est alors une matrice binaire de taille $mt \times n$ construite comme ci-dessus. Par rapport au cas général, les codes de Goppa binaires présentent l'avantage, si on ajoute la contrainte supplémentaire sur g de ne pas posséder de facteur multiple, d'avoir une distance minimale égale à 2t + 1, doublant ainsi sa capacité de correction.

5.2.3 Problèmes difficiles

En considérant l'objectif de construire une fonction à sens unique fondée sur la théorie des codes correcteurs d'erreurs, les codes de Goppa binaires sont particulièrement intéressants.En effet, il est calculatoirement difficile de distinguer un code de Goppa binaire de rendement proche de $\frac{1}{2}$ dont on ne connait ni le support ni le polynome générateur d'un code aléatoire de même longueur et de même dimension. Cela signifie intuitivement que la dificulté du décodage, dans un code de Goppa binaire de rendement proche de $\frac{1}{2}$ dont on ne connait ni le support ni le polynome générateur, ne sera pas fondamentalement différente de celle du décodage dans un code aléatoire.

5.3 McEliece

Le premier, McEliece eut l'idée, en 1978, d'utiliser la théorie des codes correcteurs d'erreurs à des fins cryptographiques, et plus précisément pour un algorithme de chiffrement asymétrique. Le principe du protocole qu'il décrivit consiste à faire envoyer par Alice un message contenant un grand nombre d'erreurs, erreurs que seul Bob sait détecter et corriger[51].

5.3.1 Cryptosystème de McEliece

C'est le plus ancien cryptosystème à clef publique utilisant des codes correcteurs d'erreurs. Il a été imaginé par McEliece en 1978, à peu près en même temps que RSA. Comme tous les crypto systèmes 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. On se propose de le faire avec le code cyclique de Hamming[52].

5.3.2 Cryptosystème de McEliece (idée de base)

— Générer un code que l'on sait décoder et sa matrice génératrice G. ceci est la clé privée.

— Transformer G pour obtenir G' qui semble aléatoire. ceci est la clé publique.

— chiffrer un message m en calculant : c'= $mG' \bigoplus e$
avec e une erreur aléatoire de poids la capacité de correction du code.

Données privées G : matrice génératrice d'un code C que l'on sait décoder
P : matrice de permutation de taille n × n
Q : matrice inversible de taille k × k

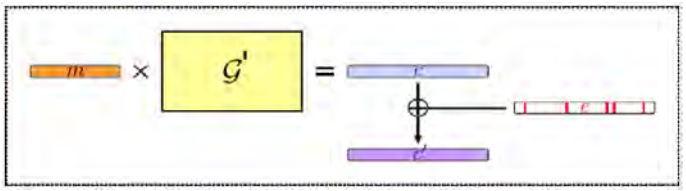

Figure 5.1 – Cryptosystème de McEliece
Source: [47]

γ_G : algorithme de décodage jusqu'à $\frac{d}{2}$ erreurs

Données publiques
$G' \stackrel{def}{=} QGP$
$t : entier \leq \frac{d}{2}$[51]

En réalité, la clef publique du système n'est autre qu'une matrice génératrice d'un autre code linéaire, C' équivalent au code C .Ce nouveau code est donc également un code de distance minimale d. En effet, effectuer le produit G'=SGP revient simplement à modifier la base choisie pour représenter le code sous forme d'une matrice génératrice (combinaison linéaire des lignes par la matrice S) et à permuter les coordonnées du code (permutations des colonnes par la matrice P). Je désignerai par la suite ce code C' par le terme code public, par opposition au code secret, C. Un message chiffré par le système, c s'écrit donc sous la forme d'un mot du code public dont t positions sont erronées. Retrouver le texte clair, ou de façon équivalente le vecteur d'erreurs e, revient alors à décoder c relativement au code public jusqu'à la distance t, i.e. résoudre, pour w = t, le problème suivant : Problème du décodage jusqu'à la distance w :

Entrée

— G : matrice binaire k × n, de rang k.
— x : vecteur binaire de longueur n

Problème
Trouver s'il existe un vecteur binaire m de longueur k tel que $d(x, mG) \leq w$[53]

5.3.3 Chiffrement

Soit m le message clair que l'on veut envoyer
1. choisir e une erreur aléatoire de poids t.
2. calculer c'=$mG' \bigoplus e$
• c' est le texte chiffré[51].
Dans la variante de McEliece, le chiffrement consiste à multiplier un vecteur de

F_2^4 par une matrice binaire k×(n-k), puis à modifier t bits choisis au hasard dans le mot de code correspondant. Bien que cela puisse constituer une difficulté dans la mise en oeuvre effective, le coût de l'ajout de l'erreur peut être néglige. En nombre d'opérations binaires, la complexité du chiffrement sera de $\frac{1}{2}k(n-k)$[52]

5.3.4 Déchiffrement

1.calculer $c'P^{-1} = (mQ)G \times PP^{-1} \bigoplus eP^{-1}$
• eP^{-1}est de poids t et (mQ)G est un mot du code ;
2. à l'aide de G on décode et on retrouve (mQ)G ;
3. prendre les k premiers bits de (mQ) G, que l'on note m_1
• on obtient alors $m_1 = mQ$ car G a été prise sous forme systématique.
4. calculer $m_1Q^{-1} = (mQ)Q^{-1}$ pour obtenir m.

5.3.5 Paramètres du système

La clé publique G' est de taille kn. La taille de la clé privée est de l'ordre de O(n)(en prenant un ensemble court générant Q et P). La complexité du chiffrement est celle d'un produit matric-evecteur, soit O(kn), la complexité du déchiffrement est celle du décodage du code soit en général de l'ordre de $O(n^2)$.
Si on prend k de l'ordre de $\frac{n}{2}$ ou $\frac{n}{3}$, on obtient alors que la taille de la clé publique (ainsi que le chiffrement et le déchiffrement) est en $O(n^2)$.
En pratique, McEliece a proposé d'utiliser la famille des **codes de Goppa binaires irréductibles** pour des paramètres $[1024; 524; 101]$ avec t=50.
Ces paramètres, proposés en 1978 pour une complexité de $\approx 2^{65}$ (opérations binaires), n'ont été cassés pratiquement qu'en 2008.
Ils conduisent à une **taille de clé de l'ordre de 500 000 bits**. Pour obtenir une complexité en 2^{80}, il conviendrait de prendre des codes de Goppa binaires irréductibles de paramètres $[2048; 1600; 46]$.
L'analyse de complexité précédente montre donc que ce système est beaucoup **plus rapide** que R.S.A, pour un même n de l'ordre de 1024 bits, on obtient un déchiffrement cubique (pour R.S.A) contre un déchiffrement quadratique (pour McEliece). Mais la **très grosse taille de clé publique** reste le problème principal pour l'utilisation pratique.
On peut potentiellement utiliser ce système avec de nombreuses familles de codes. Néanmoins, si l'on utilise une famille trop structurée comme les codes de Reed-Solomon généralisés ou les codes de Reed-Muller, il est possible de faire des attaques structurelles qui permettent de retrouver la structure du code masqué et les matrices Q et P. L'intérêt des codes de Goppa est qu'il s'agit d'une famille très importante de codes qu'on sait décoder et pour lesquels la structure (de par leur grand nombre) est plus difficile à retrouver[51].

5.3.6 Les avantages de McEliece

Le principal avantage de cette méthode est sa facilité d'implémentation : les seules opérations sont des opérations bit à bit. Par contre, l'implémentation nécessite

beaucoup de place mémoire. Ce cryptosystème, reposant sur un problème difficile de la théorie des codes, n'a pas rencontré de véritable soutien dans la communauté cryptographique. L'une des principales raisons de cet état de fait est la taille de la clé. Pourtant, le cryptosystème de McEliece possède des propriétés intéressantes, citons notamment

— la sécurité croît beaucoup plus avec la taille des clés que pour le système RSA ;

— la rapidité du chiffrement.

Un autre avantage est de reposer sur un problème très différent des algorithmes asymétriques usuels. En conséquence de quoi une percée théorique dans le domaine de la factorisation, qui ruinerait RSA, n'affecterait en rien ce cryptosystème. Le cryptosystème de McEliece résiste à ce jour à toute tentative de cryptanalyse, mais est rarement utilisé en pratique du fait de la grande taille des clés [46].

5.3.7 Les inconvénients de McEliece

1. La taille de la clef publique (G') 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 que pour la signature ou l'authentification parce que l'algorithme de chiffrage n'est pas linéaire et tout l'algorithme est vraiment asymétrique[53].

5.3.8 Cryptanalyse du système de McEliece

La sécurité repose sur l'impossibilité de déduire un algorithme de décodage efficace de la seule clé publique. Si tel est le cas, le décryptage d'une instance du système se ramènera à la résolution d'une instance d'un problème de décodage dans un code linéaire. Il en découle que toute attaque du système devra appartenir à l'une des catégories suivantes :

— les attaques par décodage : il s'agit de décoder une instance du cryptosystéme à l'aide d'un algorithme général (i.e. le code public est considère comme un code linéaire aléatoire).

— les attaques structurelles : il s'agit, à l'aide de la clé publique, de retrouver tout ou partie de la structure du code secret [52].

5.3.9 implantation

Nous présentons les différentes implantations existantes du cryptosystème de McEliece dans les Tableaux 4.1 et 4.2.

Tableau 4.1 Implantations de McEliece (jusqu'à 2009 inclus)

Titre	Auteur(s)	Année	Référence
A Software Implementation of the McEliece Public-Key Cryptosystem	Preneel Bosselaers Govaerts Vandewalle	1992	[75]
*goppa_code.c*3	"Prometheus"	1995	[76]
MicroEliece :McEliece for Embedded Devices	Eisenbarth Güneysu Heyse Paar	2009	[77]
Novel Processor Architecture for McEliece Cryptosystem and FPGA Platforms	Shoufan Wink Molter Huss Strentzke	2009	[78]

Tableau 4.2 Implantations de McEliece (depuis 2010))

Titre	Auteur(s)	Année	Référence
A Smart Card Implementation of the McEliece PKC	Strenzke	2010	[79]
Post-Quantum Cryptography on Embbeded Devices : Efficient Implementation of the McEliece Public-Key Scheme based on Quasi-Dyadic Goppa Codes	Paustjan	2010	[80]
Implementation of McEliece Based on Quasi-dyadic Goppa Codes for Embedded Devices	Heyse	2011	[81]
How SAGE helps to implement Goppa Codes and McEliece PKCs	Risse	2011	[82]
Solutions for the Storage Problem of McEliece Public and Private Keys on Memory-Constrained Platforms	Strenzke	2012	[83]
Efficient implementation of a CCA2-secure variant of McEliece using generalized Srivastava codes	Cayrel Hoffmann Persichetti	2012	[84]
Smaller Keys for Code-based Cryptography : QC-MDPC McEliece Implementations on Embedded Devices	Heyse von Maurich Güneysu	2013	[85]
A Side-Channel Secure and Flexible Platform-Independent Implementation of the McEliece PKC	Strenzke	2013	[86]
Lightweight Code-based Cryptography : QC-MDPC McEliece Encryption on Reconfigurable Devices	von Maurich Güneysu	2014	[87]
Towards Side-Channel Resistant Implementations of QC-MDPC McEliece Encryption on Constrained Devices	von Maurich Güneysu	2014	[88]
Implementing QC-MDPC McEliece Encryption	von Maurich oder Güneysu	2015	[89]
Masking Large Keys in Hardware : A Masked Implementation of McEliece	Chen Eisenbarth von Maurich Steinwandt	2016	[90]

5.4 Niederreiter

C'est une variante de McEliece reposant sur le problème de décodage par syndrome. L'idée consiste à mettre l'information dans l'erreur et à envoyer un syndrome de cette erreur. Dans la variante de Niederreiter nous devons calculer le syndrome correspondant à un mot de poids t c'est-à-dire faire la somme de t colonnes de H. En moyenne, seulement Rt de ces colonnes seront effectivement additionnées, les autres seront dans la partie ((identité)) de la matrice H et auront pour seul effet de modifier un bit dans le résultat, soit environ $Rt(n-k) + (1-R)t \approx Rt(n-k)$ opérations binaires[52].

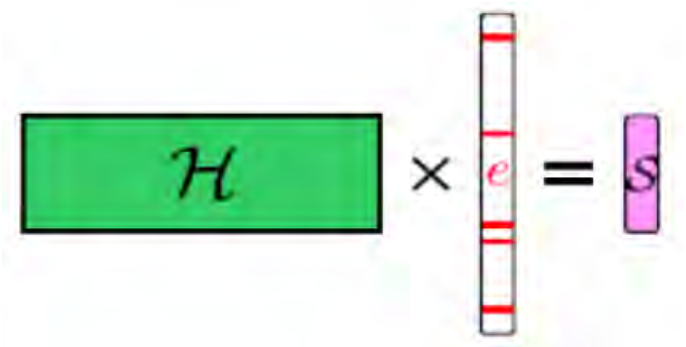

Figure 5.2 – Décodage par Syndrome
Source: [47]

Données privées

H : matrice de parité d'un code $C[n,k,d]$ sur F_q
P : matrice de permutation de taille $n \times n$
Q : matrice inversible de taille $(n-k) \times (n-k)$
γ_H : Algorithme de décodage jusqu'à $\frac{d}{2}$ erreurs

Données publiques

$H' \stackrel{def}{=} QHP$
$t : entier \leq \frac{d}{2}$

5.4.1 Chiffrement

Convertir les données en erreur e de longueur n et de poids t ; Calculer $S = H'^t e$ (somme de t colonnes de H') ;
S est le texte chiffré.

5.4.2 Déchiffrement

Calculer $Q^{-1}S = Q^{-1}QH(Pe)$;
Pe est de poids t et peut être décodé ;

Calculer $P^{-1}(Pe)$ pour obtenir e ;
Reconvertir e en texte clair[51].

5.4.3 Principe général du système de Niederreiter

Le système proposé par Niederreiter correspond à une approche duale du système de McEliece. Cette fois-ci, la clef publique est une matrice de parité du code public C'. Le texte clair correspond à un vecteur de longueur n et de poids t, et le texte chiffré à son syndrome relativement au code C'. Retrouver le texte clair à partir du chiffré revient donc ici à déterminer l'unique mot de poids t ayant pour syndrome c, i.e. résoudre, pour w = t, le problème suivant :

Problème de la recherche des mots de poids inférieur à w d'un code

Entrée

— H :matrice binaire $n \times r$, de rang r.

— s :vecteur binaire de longueur r.

Problème Trouver, s'il existe, un vecteur binaire e de longueur n et de poids inférieur ou égal à w tel que :eH=s.

5.4.4 Bilan

Niederreiter proposait d'utiliser ce système soit avec un code binaire $[104, 24, 32]$ résultant de la concaténation du code de Hamming étendu de longueur 8 et dimension 4 avec un code de Reed-Solomon poinçonné de longueur 13 et de dimension 6 défini sur F_{16}, soit avec un code de Reed-Solomon $[30, 12, 19]$ sur F_{31}. Les paramètres de ces codes sont évidemment trop petits pour que le système résiste à la cryptanalyse, comme l'ont montré Brickell et Odlyzko en utilisant l'algorithme LLL. La comparaison que j'établis entre les systèmes de McEliece et de Niederreiter, qui n'a donc de sens que si les familles de codes utilisées dans les deux cas sont identiques. Même si leurs sécurités sont équivalentes, ces deux systèmes présentent en effet des différences notables. L'utilisation du système de Niederreiter permet de réduire de moitié la taille des clefs publiques. Elle autorise en effet le stockage de la matrice de parité H' sous forme systématique, qui était impossible pour le système de McEliece puisqu'il aurait révélé une partie du texte clair[53]. Un autre avantage du système de Niederreiter, contrairement au système de McEliece, il produit un chiffrement complétement déterministe puisque les différents chiffrés d'un même texte clair sont toujours identiques. On détecte donc immédiatement que deux chiffrés correspondent au même texte clair mais on ne dispose pas d'algorithme rapide pour les déchiffrer.

5.5 Variante de Sendrier et Biswas

En 2008 N. Sendrier et B. Biswas ont proposé dans [71] une nouvelle variante du cryptosystème de McEliece qu'ils qualifient d'hybride. Cette variante introduit deux modifications du schéma : tout d'abord ils proposent d'utiliser le cryptosystème de McEliece tout en introduisant une partie de l'information à transmettre au sein meme de l'erreur qui est ajoutée au mot de code contenant l'essentiel de l'information du message. Ceci permet d'accroitre le taux de transmission du schéma de McEliece. Ensuite, ils proposent de réduire la taille des clefs en utilisant un code public décrit par une matrice génératrice sous forme systématique. Ce schéma est décrit en utilisant des codes de Goppa.

5.5.1 Génération de clés

Comme ce cryptosystème est basé sur les codes, nous avons besoin de deux entiers n et t lors de la génération des clés, qui définiront respectivement la longueur et la capacité de correction d'un code linéaire, où t sera beaucoup plus petit que n. La génération des clés de HyMES est présentée dans l'Algorithme suivant

Algorithme 12 Génération de clés de HyMES

Entrée(s): n et t deux entiers.
Sortie(s): $p_k = (\mathcal{A}, t)$ la clé publique et $s_k = (\mathscr{L}, G)$ la clé privée associée.

1: $\ell \leftarrow \lfloor \log_2 \binom{n}{t} \rfloor$.
2: Choisir un code de Goppa $\Gamma(\mathscr{L}, G)$ de longueur n et t correcteur, dont le support contient $n = 2^m$ éléments.
On notera k la dimension du code $\mathscr{C}$.
3: Prendre une matrice $\mathcal{A}$ telle que la matrice $\mathcal{G} = [\mathcal{I}_k | \mathcal{A}] \in \mathscr{M}_{k,n}(\mathbb{F}_2)$ soit génératrice de $\Gamma(\mathscr{L}, G)$.
4: $s_k \leftarrow (\mathscr{L}, G)$
5: $p_k \leftarrow (\mathcal{A}, t, \ell)$
6: **Retourner** (p_k, s_k).

Figure 5.3 – Génération de clés de HyMES

5.5.2 Chiffrement

Comme dans tout système de chiffrement à clé publique, nous avons besoin du message à chiffrer et de la clé publique pour la phase de chiffrement. Le chiffrement de HyMES est présenté dans l'Algorithme suivant

Algorithme 13 Chiffrement de HyMES

Entrée(s): $p_k = (\mathcal{A}, t, \ell)$ la clé publique, $M \in \mathbb{F}_2^k$ un message à chiffrer et $\tilde{M} \in \mathbb{F}_2^\ell$ une information supplémentaire.
Sortie(s): $\tilde{C} \in \mathbb{F}_2^n$ le texte chiffré associé à $(M, \tilde{M})$.
1: Encoder le message M : $C = M \cdot [\mathcal{I}_k | \mathcal{A}]$.
2: Encoder en mot de poids t et de longueur n le message $\tilde{M}$: $E = \varphi(\tilde{M})$.
3: Calculer $\tilde{C} = C \oplus E$.
4: **Retourner** $\tilde{C}$.

Figure 5.4 – Chiffrement

Pour l'algorithme d'encodage en poids constant, il s'agit du même que pour le cryptosystème de Niederreiter, voir l'Algorithme

5.5.3 Déchiffrement

Comme dans tout système de chiffrement à clé publique, nous avons besoin du texte chiffré à déchiffrer et de la clé privée pour la phase de déchiffrement. Le déchiffrement de HyMES est présenté dans l'Algorithme

Algorithme 14 Déchiffrement de HyMES

Entrée(s): $s_k = (\mathscr{L}, G)$ la clé privée, $\tilde{C} \in \mathbb{F}_2^n$ le texte chiffré.
Sortie(s): $M \in \mathbb{F}_2^k$ le texte clair et $\tilde{M} \in \mathbb{F}_2^\ell$ l'information supplémentaire, tous deux associés à $\tilde{C}$.
1: Décoder $\tilde{C}$ pour retrouver C et E.
2: Récupérer M : les k premiers bits de $\tilde{C} \oplus E$.
3: Calculer $\tilde{M} = \varphi^{-1}(E)$.
4: **Retourner** $(M, \tilde{M})$.

Figure 5.5 – Dechiffrement

5.5.4 Sécurité

Deux différences sont notables dans la version hybride par rapport à la version originale. La première est l'utilisation de l'erreur pour faire passer de l'information en plus. La deuxième est remarquable dans l'algorithme de chiffrement : la matrice génératrice publique est sous forme systématique. De plus, ces changements n'entraînent en aucun cas une diminution du niveau de sécurité de ce cryptosystème comparé au cryptosystème de McEliece sous les hypothèses algorithmiques de la diffculté du décodage (décodage par syndrome et du décodage à poids borné d'un code de Goppa et de la distinction entre un code de Goppa et un code aléatoire. Cependant la deuxième hypothèse n'est pas toujours vraie (lorsque le rendement est élevé) . Toutefois, la sécurité du cryptosystème d'HyMES repose sur la même fonction à sens unique que le cryptosystème de McEliece.

5.6 un schéma de signature numérique

CFS(Courtois, Finiasz et Sendrier)[72] est un schéma de signature numérique basé sur le cryptosystème de Niederreiter[66] . Il fut publié en 2001 et s'appuie sur la difficulté du problème du décodage par syndrome et sur l'indistinguabilité des codes de Goppa binaires.
Le monde de la signature numérique est peu diversifiée ; il existe relativement peu de primitive de signature et beaucoup sont basées sur la théorie des nombres. CFS, étant basé sur la théorie des codes, ne sera pas vulnérable aux améliorations algorithmiques que l'ordinateur quantique apporterait s'il venait un jour à atteindre des performances raisonnables et offrirait une alternative le moment venu.
Cependant, les problèmes liés à la mise en oeuvre de CFS ont reçus peu d'attention. Cela vient peut-être de l'aspect peu pratique apparent du système et des résultats de cryptanalyses qui ont affaibli le schéma, au moins d'un point de vue théorique. L'apparence peu pratique du système viens de la grande taille de la clé publique et des longs temps de signature. Certes la taille de la clé publique peut être un problème pour certaines applications, mais certains scénarios d'utilisation peuvent s'accommoder d'un espace de stockage de quelques mégaoctets pour vérifier des signatures. L'impression de lenteur de la primitive de signature peut s'expliquer par les premiers temps donnés dans le papier d'origine [72] qui font mention d'une mise en oeuvre logicielle générant une signature en une minute. Or il s'agissait là d'une démonstration de faisabilité. Une mise en oeuvre sur circuit logique programmable (ou FPGA) décrite dans [73] annonce une signature en moins d'une seconde. Il a été prouvé dans[74] que la clé publique de CFS pouvait être distinguée en temps polynomial d'une matrice binaire aléatoire. Cette propriété affaiblit la preuve de sécurité du système mais aucune attaque n'en a été déduite.

5.6.1 Concept

Nous considérerons uniquement les codes linéaires binaires. La plupart des faits énoncés ici pourraient se généraliser à un alphabet plus grand mais aucun schéma semblable à CFS utilisant des codes non binaires n'a été proposé jusqu'à présent.
Une instance de CFS est définie par un code de Goppa binaire Γ de longueur n capable de corriger jusqu'à t erreurs ; de matrice de parité H ; sur le corps fini F2. Nous appellerons decode la fonction de décodage de Γ. Cette fonction prend un syndrome binaire en entrée et renvoie un t-uplet de positions d'erreur correspondant à un motif d'erreur ayant l'entrée pour syndrome ou échoue si un tel motif n'existe pas. La matrice H est publique et la procédure decode est secrète. Signer un document se fait de cette façon :

1. Calculer l'empreinte du document (via une fonction de hachage).
2. Supposer que cette empreinte est un syndrome et utiliser decode pour tenter de la décode
3. La signature est le motif d'erreur obtenu.

Puisque l'empreinte du document a très peu de chance d'être un syndrome décodable (c'est-à-dire le syndrome d'un mot à distance de Hamming t ou moins d'un mot du code Γ), l'étape 2 va très sûrement échouer. Deux solutions sont proposées pour contourner cette limitation :

— Le décodage complet [voir Algorithme 1] ajoute un certain nombre de colonnes de H au syndrome jusqu'à ce qu'il devienne décodable (cela revient à tenter de deviner quelques erreurs).

— L'adjonction d'un compteur modifie le message avec un compteur puis calcule l'empreinte jusqu'à ce le syndrome associé devienne décodable. Le compteur qui a rendu le syndrome décodable est adjoint à la signature.

Les deux méthodes nécessitent une moyenne de t ! tentatives de décodage avant succès . L'adjonction d'un compteur a pour inconvénient d'inclure la fonction de hachage dans la procédure de décodage ce qui oblige à la mettre en oeuvre sur la plateforme cible, ce qui serait dérangeant pour un coprocesseur dédié. De plus, cette méthode rend la taille de la signature variable puisque celui-ci fait partie de la signature et qu'il a un écart type élevsé. Pour finir, la contre-mesure Parallel-CFS (voir 4.7.2) n'est pas applicable si cette méthode est employée.

Algorithm 1 Algorithme de Buchberger

Entrée : $G = \{g_1, g_2, ..., g_n\}$ un système générateur de I
Sortie : Base de Gröbner de $I = \langle G \rangle$
  $P \leftarrow \{\{f, g\}/f, g \in G\}$ et $f \neq g$
  Tant que $P \neq \varnothing$ **Faire**
    choisir un $\{f, g\} \in P$
    $P \leftarrow P \setminus \{f, g\}$
    $r \leftarrow$ Division de polynômes$(G, S_P(f, g))$
    Si $r \neq 0$ **Alors**
      $P \leftarrow P \cup \{\{r, f\}/f \in G\}$
      $G \leftarrow G \cup \{r\}$
    fin Si
  fin Tant que
  Retourne G

5.7 Attaques

5.7.1 Attaques théoriques

Il y a deux types d'attaques théoriques pour effectuer une cryptanalyse du cryptosyst ème de McEliece : les attaques par décodage et les attaques structurelles. Une attaque par décodage (ou attaque générique) consiste à voir la matrice génératrice publique G comme une matrice génératrice d'un code aléatoire et d'essayer

de retrouver l'erreur utilisée lors du chiffrement pour retrouver le message initial à partir d'un texte chiffré. Une attaque structurelle (ou attaque algébrique) consiste à chercher un décodeur du code publique, qui est équivalent au code privé. Le but est de retrouver la matrice de permutation (et a fortiori le code privé). Une attaque structurelle est donc plus puissante qu'une attaque par décodage, car la première cible la clé privée tandis que la seconde cible un message en clair.
Qui dit "cryptographie basée sur les codes", dit "choix d'un code". Nous présentons dans le Tableau suivant les différentes familles de codes qui ont été proposées et les attaques théoriques répertoriées.

Noms des codes	Propositions	Attaques
Goppa (classiques binaires)	[McE78]	
Reed-Solomon généralisés	[Nie86]	[SS92, Wie10]
sous-codes	[BL05, Wie06]	[Wie10, MCMMP13, CGGU+14]
Reed-Muller binaires	[Sid94]	[MS07]
Algébriques-géométriques	[JM96]	[MS07, FM08, MCMMP14, MCMMPR14, CMCP16]
sous-codes		[CMCP16]
MDPC	[MTSB13]	

Figure 5.6 – Cryptosystèmes avec différents codes
Source: `https://www.researchgate.net/publication/324476461`

On peut constater que la cryptographie utilisant les codes de Goppa reste encore non-cassée (pour des choix de paramètres judicieux), ce qui explique notre intérêt pour cette famille de codes. Nous ne nous intéressons pas dans cette thèse aux codes MDPC (Moderate Density Parity-Check en Anglais, pour des codes ayant des matrices de contrôles dont le poids de chaque ligne est constant), mais cela reste une piste pour la suite de ces travaux.

5.7.2 Attaques en pratique (par canaux auxiliaires)

La grande majorité des attaques par canaux auxiliaires contre le système de chiffrement à clé publique de McEliece portent sur l'algorithme de Patterson utilisé pour le décodage des codes de Goppa classiques binaires irréductibles lors de la phase de déchiffrement.l'étude des attaques par canaux auxiliaires contre ce cryptosystème n'a débuté qu'en 2008 soit 30 ans après l'apparition de celui-ci..

Titre	Auteur(s)	Année	Réf.	Type d'attaques
Side channels in the McEliece PKC	Strenzke Tews Molter Overbeck Shoufan	2008	[STM+08]	TA [11]
A Timing Attack *against Patterson Algorithm* *in the McEliece PKC*	Shoufan Strenzke Molter Stöttinger	2009	[SSMS10]	TA poids de l'erreur dans EEA
A Timing Attack *against the Secret Permutation* *in the McEliece PKC*	Strenzke	2010	[Str10b]	TA matrice de perm. dans EEA
Practical Power Analysis Attacks *on Software Implementations of McEliece*	Heyse Moradi Paar	2010	[HMP10]	PA [12] clé privée
McEliece/Niederreiter PKC : *sensitivity to fault injection*	Cayrel Dusart	2010	[CD10]	FI [13]
Side-Channel Attacks *on the McEliece and Niederreiter* *Public-Key Cryptosystems*	Avanzi Hoerder Page Tunstall	2011	[AHPT11]	TA amélioration de [STM+08]

Figure 5.7 – Attaques en pratique (par canaux auxiliaires)
Source: `https://www.researchgate.net/publication/324476461`

A simple power analysis attack *on a McEliece cryptoprocessor*	Molter Stöttinger Shoufan Strenzke	2011	[MSSS11]	SPA [14] poids de l'erreur dans EEA
Message-aimed side channel *and fault attacks* *against public key cryptosystems* *with homomorphic properties*	Strenzke	2011	[Str11]	FI, TA message
Timing Attacks *against the Syndrome Inversion* *in Code-based Cryptosystems*	Strenzke	2013	[Str13b]	TA polynôme de Goppa
Towards *Side-Channel Resistant Implementations* *of QC-MDPC McEliece Encryption* *on Constrained Devices*	von Maurich Güneysu	2014	[vMG14b]	(TA,) (S)PA message (ds enc.) clé privée (ds dec.)
Differential Power Analysis *of a McEliece Cryptosystem*	Chen Eisenbarth von Maurich Steinwandt	2015	[CEvMS15]	DPA [15] (1ère) clé privée

Figure 5.8 – Attaques en pratique (par canaux auxiliaires)
Source: `https://www.researchgate.net/publication/324476461`