THESE de DOCTORAT de l'UNIVERSITE PARIS 6

Spécialité : INFORMATIQUE

présentée par

Anne CANTEAUT

pour obtenir le grade de DOCTEUR de l'UNIVERSITE PARIS 6

Sujet de la thèse :

ATTAQUES DE CRYPTOSYSTÈMES À MOTS DE POIDS FAIBLE ET CONSTRUCTION DE FONCTIONS t-RÉSILIENTES

Soutenue le 10 octobre 1996, devant le jury composé de :

Paul CAMION	Directeur
Claude CARLET	
Pascale CHARPIN	
Alain LANUSSE	
James L. MASSEY	Rapporteur
Michel MINOUX	Président
Jacques STERN	Rapporteur

Introduction

L'invention des techniques de chiffrement à clef publique a suscité un certain émoi dans le monde de la cryptographie puisqu'elle a montré que l'on pouvait chiffrer des données sans avoir au préalable échangé un secret par un canal sécurisé, que l'on pouvait produire une signature numérique permettant d'identifier formellement l'auteur d'un message, et même jouer à pile ou face par téléphone... Aujourd'hui de nombreux problèmes demeurent toutefois sans solution ; d'autres difficultés sont apparues depuis que les progrès de l'informatique et les efforts acharnés des cryptanalystes sont venus à bout de certains systèmes considérés comme sûrs quelques années auparavant. Désormais les systèmes de chiffrement à clef publique résistant encore à la cryptanalyse se comptent sur les doigts d'une main ; la famille des fonctions de hachage cryptographiquement sûres est réduite à une peau de chagrin ... Curieusement la recherche de nouvelles primitives et l'évaluation de leur sécurité n'ont jamais semblé aussi primordiales qu'aujourd'hui.

A la suite des travaux de Rivest, Shamir et Adleman émergèrent plusieurs systèmes de chiffrement à clef publique exploitant la complexité de divers problèmes mathématiques. Les principaux furent successivement le système de Merkle et Hellman [MH78] reposant sur le problème du sac-à-dos, celui de McEliece [McE78] utilisant les codes correcteurs d'erreurs, ceux de Chor et Rivest [CR84] et de ElGamal [ElG84] exploitant de différentes manières le problème du logarithme discret, ceux de Matsumoto et Imai [MI83, MI88] qui manipulaient des polynômes sur des corps finis ... Mais les travaux des cryptanalystes eurent peu à peu raison d'une telle richesse et cette belle diversité est aujourd'hui réduite aux seuls systèmes fondés sur la théorie des nombres. "Hors du RSA, point de salut" semble être la conclusion de ces vingt dernières années. A travers elle apparaît tout le danger de la situation : la cryptographie à clef publique est devenue entièrement dépendante de deux problèmes voisins, celui de la factorisation et celui du calcul du logarithme discret. Il est donc aujourd'hui indispensable d'imaginer et d'améliorer des algorithmes à clef publique reposant sur d'autres problèmes afin de prévenir d'éventuels progrès par exemple dans les techniques de factorisation. Il serait de surcroît souhaitable de pallier l'extrême lenteur des systèmes communément employés, qui effectuent tous de nombreuses multiplications sur de très grands entiers. C'est pourquoi la première partie de cette thèse est consacrée à l'analyse de la sécurité de certains

cryptosystèmes fondés sur la théorie des codes. Ces algorithmes à clef publique, dont le plus connu, celui de McEliece [McE78], résiste depuis 18 ans à la cryptanalyse, sont actuellement la seule alternative crédible aux systèmes issus de la théorie des nombres. Dans cette perspective, j'ai conçu un nouvel algorithme de recherche de mots de poids faible dans un code linéaire de grande taille qui m'a en particulier permis de montrer que les paramètres initialement proposés par McEliece n'apportaient pas une sécurité satisfaisante à ces systèmes de chiffrement.

Malgré l'émergence des algorithmes à clef publique, les primitives conventionnelles ne tombèrent pas en désuétude puisqu'elles seules permettent d'atteindre des débits de chiffrement satisfaisants. Le monde de la cryptographie à clef secrète est plus diversifié que celui de la cryptographie à clef publique mais il paraît également plus fragile car la sécurité de ces systèmes n'a jamais fait l'objet d'une preuve formelle. En effet, le principal critère utilisé pour la conception de la plupart de ces primitives, construites à partir d'un réseau dont chaque sommet est une boîte de calculs, est simplement que le réseau soit assez inextricable et les boîtes de calcul suffisamment irrégulières. Une telle condition semble bien peu fiable et elle ne permet en aucun cas de cerner la sécurité de ces algorithmes par un problème mathématique précis. Contrairement aux systèmes à clef publique, leur capacité de résistance à la cryptanalyse est tout à fait imprévisible : les fonctions de hachage qui étaient employées dans la plupart des systèmes, MD4 et MD5, viennent ainsi d'être cryptanalysées [Dob95, Dob96]. Les critères à respecter pour qu'un générateur pseudo-aléatoire ou une fonction de hachage soit solide restent encore à déterminer. Quelques travaux récents, en particulier ceux de Serge Vaudenay [Vau95a], ont cependant ébauché une théorisation du problème et ont mis en évidence l'importance de certains objets mathématiques, telles les fonctions courbes et les fonctions résilientes étudiées en détail dans la seconde partie de ce mémoire. J'y montre notamment combien l'emploi de ces fonctions est primordial tant pour réaliser de bons générateurs pseudo-aléatoires, et donc des algorithmes de chiffrement à flot solides et rapides, que pour analyser la sécurité des primitives cryptographiques conventionnelles.

Les travaux présentés dans cette thèse s'articulent donc autour de deux thèmes : l'évaluation de la sécurité des cryptosystèmes à mots de poids faible et l'étude et la généralisation de la propriété de fonction résiliente. Ces deux sujets faisant appel à certains résultats de théorie des codes, le lecteur trouvera en annexe les quelques définitions et propriétés élémentaires des codes correcteurs d'erreurs nécessaires à la compréhension de ce mémoire.

Chapitre 1

Introduction à la cryptographie

Il fut un temps où remplacer chaque lettre d'un message par celle située trois positions plus loin dans l'alphabet suffisait aux empereurs romains pour se mettre à l'abri des regards indiscrets. Mais, depuis que les curieux ont appris à lire, à compter jusqu'à 3 puis jusqu'à 26 et à utiliser un ordinateur, les secrets de Jules César se sont passablement éventés. Face aux progrès technologiques, ses successeurs durent alors recourir à l'ingéniosité des cryptographes. Nourrie par les mathématiques discrètes, la théorie de la complexité et la théorie de l'information, leur imagination est toujours à la recherche de techniques efficaces et infaillibles permettant de protéger les données confidentielles d'éventuels indiscrets. L'avènement de la cryptographie moderne, depuis la découverte en 1976 des systèmes à clef publique, n'a pas apporté de solution parfaite aux cryptographes dans leur quête effrénée de systèmes toujours plus rapides et plus sûrs.

Les solutions apportées par la cryptographie pour une meilleure sécurité de l'information se répartissent en deux grandes catégories suivant leur fonctionnalité : les *algorithmes de chiffrement*, qui assurent la confidentialité des données, et les *algorithmes d'authentification*, qui garantissent en l'authenticité et la provenance.

1 Le chiffrement

Il est possible de mettre un message hors de portée des oreilles indiscrètes en le transformant en un *texte chiffré* à l'aide d'une *fonction de chiffrement* C, paramétrée par une clef K_C. Un interlocuteur privilégié peut alors déchiffrer le message en utilisant la *fonction de déchiffrement* D s'il connaît la clef K_D (cf. figure 1.1).

Un tel système n'est sûr que s'il est impossible à un intrus de déduire le texte clair du message chiffré, et *a fortiori* de retrouver la clef de déchiffrement.

On distingue deux grands types d'algorithmes de chiffrement.

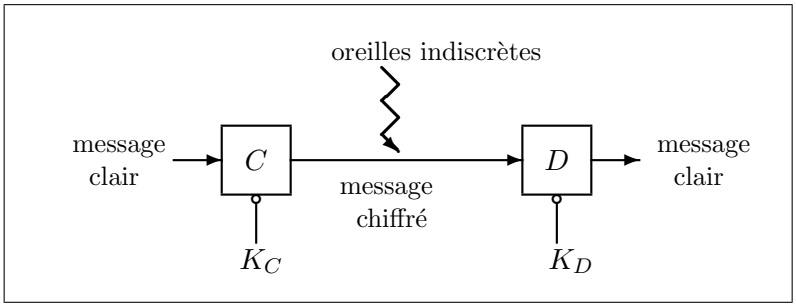


FIG. 1.1 –: *Principe général d'un algorithme de chiffrement*

1.1 Le chiffrement à clef secrète

Les *algorithmes de chiffrement à clef secrète* (ou *symétriques* ou encore *conventionnels*) sont ceux pour lesquels émetteur et destinataire partagent une même clef secrète — autrement dit, les clefs de chiffrement et de déchiffrement sont identiques.

La technique de chiffrement à clef secrète la plus élémentaire et la plus sûre est le *le chiffrement à flot*, désigné plus explicitement en anglais par l'expression *one-time pad*. Il s'agit d'un des rares algorithmes qui assure une sécurité parfaite, *i.e.* qu'il est théoriquement impossible de retrouver le message clair à partir du chiffré sans connaître la clef secrète, à condition que celle-ci soit une suite complètement aléatoire. Cependant, comme il n'est en général pas envisageable de partager une clef secrète qui soit aussi longue que le message à chiffrer, on utilise dans la pratique une *suite pseudo-aléatoire* générée de façon déterministe à partir d'un secret commun court qui, lui, peut être échangé plus facilement.

Pour contourner le problème difficile de la génération de suites pseudo-aléatoires, on fait souvent appel à des techniques de chiffrement à clef secrète dites *par blocs*. Elles consistent généralement à effectuer, à partir d'un bloc de message (de 64 ou 128 bits) et d'une clef secrète, une succession suffisamment inextricable de calculs pour qu'il soit impossible d'avoir une vision d'ensemble de la fonction. Les plus employées sont le DES, développé par le gouvernement américain en 1977, et l'algorithme IDEA, conçu par Lai et Massey [LM90].

Leur principal intérêt est leur rapidité : un prototype développé par DEC, mettant en œuvre le DES avec plus de 50 000 transistors, atteint un débit de 1 Gbit/s [Ebe92]. Leur vitesse impose cependant que la taille des clefs soit relativement importante car il est essentiel de se prémunir des attaques exhaustives, qui passent en revue toutes les clefs possibles. Alors qu'aucune attaque réellement concluante du DES n'avait émergé en 20 ans — la plus performante, la *cryptanalyse linéaire* [TCG91, Mat93], nécessite la connaissance de 2^{43} couples clair-chiffré —, sa sécurité est aujourd'hui compromise par son débit. La clef secrète n'étant que de 56 bits, il est possible d'en essayer toutes les possibilités en

un temps raisonnable dès lors que l'on dispose de suffisamment de cartes DES. Actuellement, la solidité d'un chiffrement DES est alors évaluée par le coût financier du matériel nécessaire pour passer en revue les 2^{56} clefs en un temps donné (cf. [Sch96, page 153]). L'algorithme IDEA emploie en revanche des clefs de 128 bits, ce qui le met à l'abri de ce type d'attaques.

D'un point de vue algorithmique, la sécurité de ces systèmes ne repose pas sur une théorie mathématique mais simplement sur la constatation empirique qu'ils sont difficiles à cryptanalyser. Cette absence de théorie est d'autant plus perceptible qu'une grande partie des travaux effectués sur les systèmes conventionnels est restée jusqu'à ce jour secrète.

1.2 Le chiffrement à clef publique

La *cryptographie à clef publique* (ou *asymétrique*), inventée en 1976 par Diffie et Hellman [DH76], évite le partage d'un secret entre les deux protagonistes. Ainsi, pour assurer la confidentialité de données lors d'une transmission, il suffit à l'émetteur de chiffrer le message avec la clef publique du destinataire, généralement disponible dans un annuaire. Ce dernier, à l'aide de la clef secrète correspondante, est seul en mesure de déchiffrer le message reçu.

Les algorithmes de chiffrement à clef publique sont donc construits à partir d'une fonction facile à calculer mais que l'on ne peut inverser en un temps raisonnable que si l'on connaît un certain secret, la clef de déchiffrement. De telles fonctions, appelées *fonctions à sens unique avec trappe*, sont fournies par des problèmes mathématiques réputés difficiles. Ainsi le système RSA, du nom de ses auteurs Rivest, Shamir et Adleman [RSA78], repose sur la difficulté de la factorisation : la clef secrète d'un utilisateur est composée de deux grands nombres premiers, chacun ayant à peu près 300 bits, et sa clef publique est égale au produit de ces deux nombres (cf. figure 1.2).

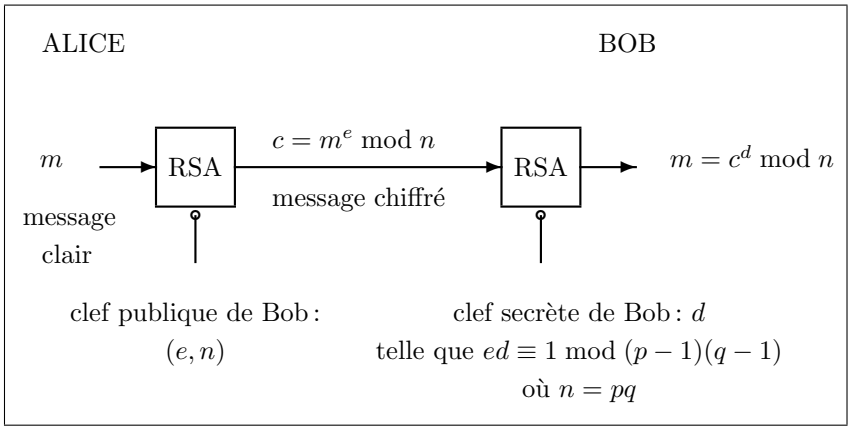


FIG. 1.2 –: *Algorithme de chiffrement RSA*

Le principe est donc qu'il est très difficile, vue la taille des nombres employés,

de factoriser une clef publique pour retrouver la clef secrète correspondant.

Un inconvénient majeur des systèmes à clef publique est leur excessive lenteur : dans le meilleur des cas, le RSA est mille fois moins rapide que le DES. Comme il est très contraignant de chiffrer un long message avec le RSA, on combine généralement les algorithmes à clef publique et à clef secrète : le message est chiffré au moyen d'un algorithme symétrique très rapide dont la clef est transmise avec le RSA.

2 L'authentification

Dans certains cas, la confidentialité des données importe peu mais il est nécessaire de s'assurer de leur intégrité, c'est-à-dire de vérifier qu'elles n'ont pas été modifiées lors de la transmission, ou simplement de leur provenance.

2.1 La signature numérique

La *signature numérique* consiste à adjoindre au texte clair un petit nombre de bits qui dépendent simultanément du message et de son auteur. Pour obtenir les mêmes fonctionnalités que la signature que l'on appose au bas d'un texte à support papier, il faut que chacun puisse vérifier une signature mais que personne ne puisse l'imiter. Elle garantit alors à la fois l'intégrité du message reçu et l'identité de son auteur.

Certains algorithmes de chiffrement à clef publique fournissent immédiatement une solution : les fonctions de chiffrement et de déchiffrement du RSA étant identiques, il suffit à l'auteur de chiffrer un condensé du message avec sa propre clef secrète pour produire une signature ; chacun pourra alors la vérifier en utilisant la clef publique correspondante.

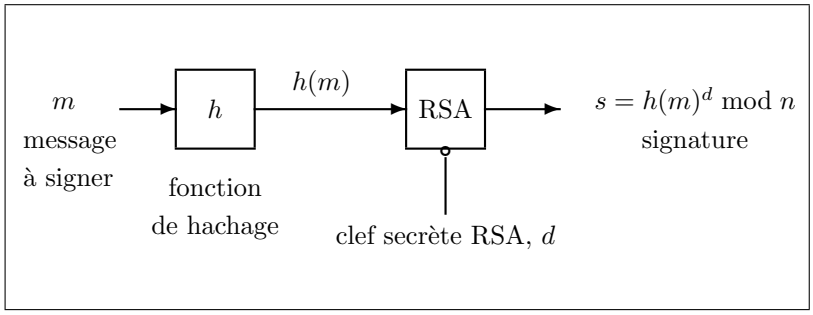


FIG. 1.3 –: *Signature numérique utilisant l'algorithme RSA*

L'emploi d'une *fonction de hachage* associant à tout message un condensé de longueur fixée est imposé par la lenteur du chiffrement RSA et la volonté que la signature soit courte et de taille constante. Une telle fonction doit à la fois résister aux *collisions*, c'est-à-dire qu'il doit être très difficile de trouver deux

messages ayant même condensé — ce qui impose au condensé une longueur d'au-moins 128 bits — et être très rapide. On utilise donc en général des fonctions de même type que les fonctions de chiffrement par blocs décrites précédemment.

2.2 L'identification

On peut aussi désirer simplement s'assurer de l'identité de son interlocuteur, indépendamment de l'émission d'un message. On utilise pour cela des *protocoles d'identification* : un utilisateur prouve son identité en montrant qu'il connaît un secret qui lui est propre. L'emploi d'un mot de passe est une solution très rudimentaire puisque l'utilisateur est contraint de dévoiler son secret. Un protocole interactif peut en revanche lui permettre de convaincre son interlocuteur qu'il connaît le secret sans jamais l'exhiber. Un tel protocole est dit *à apport nul de connaissance* (*zero-knowledge*) si la transaction ne fournit aucune autre information sur le secret de l'utilisateur que les données publiques [GMR85]. L'un des schémas d'identification les plus employés est celui de Fiat-Shamir [FS86] qui, comme le RSA, repose sur un problème de la factorisation.

Partie I

Attaque des cryptosystèmes à mots de poids faible

Introduction

Au cours des quinze dernières années, les systèmes de chiffrement à clef publique solides se sont désespérément raréfiés au point de se réduire à la famille des systèmes fondés sur les problèmes de théorie des nombres. Cette entière dépendance de la cryptographie à clef publique de deux uniques problèmes — qui sont du reste très proches —, celui de la factorisation et celui du logarithme discret, suscite une grande inquiétude car nul ne peut exclure d'importants progrès dans les techniques de factorisation. L'hypothèse de nouvelles attaques efficaces du RSA semble encore plus menaçante car il n'a jamais été prouvé qu'il était nécessaire de savoir factoriser une clef publique pour décrypter les messages chiffrés par le RSA. Il n'est donc pas inimaginable qu'apparaisse subitement un algorithme remettant en cause la sécurité de la plupart des systèmes utilisés actuellement pour protéger l'information. Les travaux récents de Coppersmith [Cop96] ont déjà révélé de façon relativement spectaculaire que l'emploi de l'exposant public 3 — choix effectué dans bon nombre de mises en œuvre du RSA afin d'augmenter le débit de chiffrement — était en général extrêmement dangereux.

Pour prévenir une telle situation, il est donc indispensable de disposer d'une alternative solide au RSA même si elle n'en présente pas tous les avantages. Il me paraît de toute manière peu probable dans l'immédiat de concevoir un système de chiffrement à clef publique possédant autant d'avantages que le RSA — un taux de transmission égal à 1, des fonctions de chiffrement et de déchiffrement identiques, des clefs publiques de petite taille...

Ceci a donc motivé mon intérêt pour la seule classe de systèmes à clef publique fondés sur des problèmes autres que ceux de la théorie des nombres et qui résiste encore à la cryptanalyse : les *cryptosystèmes à mots de poids faible*. Je regroupe sous cette appellation tous les systèmes reposant sur la difficulté de trouver un mot de poids faible dans un code linéaire, c'est-à-dire les systèmes de chiffrement de McEliece [McE78] et de Niederreiter [Nie86] ainsi que les schémas d'identification proposés successivement par Marc Girault [Gir90], Jacques Stern [Ste93] et Pascal Véron [Vér95b]. La taille importante des clefs utilisées et leur taux de transmission relativement faible ont conduit certains à négliger ces systèmes de chiffrement, sans se rendre compte qu'ils présentaient un avantage décisif sur le RSA : ils sont considérablement plus rapides ! Pour toutes ces raisons, leur sécurité mérite donc d'être étudiée de très près.

Cette première partie débute donc par une présentation générale des sys-

tèmes de chiffrement de McEliece et de Niederreiter et des schémas d'identification de Girault, de Stern et de Véron, suivie d'une comparaison de leurs performances avec les systèmes à clef publique employés habituellement. Je montre au chapitre 3 qu'en l'absence d'attaque structurelle induite par l'utilisation des codes de Goppa irréductibles comme clefs publiques, la cryptanalyse de ces systèmes se ramène au problème général du décodage d'un code linéaire jusqu'à une distance fixée ou à la recherche de mots de poids minimal. Je détaille et optimise alors les différentes variantes de l'algorithme de décodage par ensembles d'information. Le chapitre suivant est consacré à l'étude de deux nouveaux algorithmes de décodage et de recherche de mots de poids minimal que j'ai conçus en collaboration avec Hervé Chabanne et Florent Chabaud [CC94, CC95a]. Une analyse très précise de leur complexité à l'aide d'une modélisation par un processus markovien me permet d'améliorer les attaques précédemment connues des systèmes à mots de poids faible [Can95] et de détecter certaines faiblesses qui compromettent la sécurité du système de McEliece. L'efficacité de ces nouveaux algorithmes apparaît pleinement au chapitre 5 puisqu'ils m'ont permis de déterminer la distance minimale de certains codes BCH de longueur 511, problème qui restait ouvert depuis près de 30 ans [CC95b].

Chapitre 2

Principaux cryptosystèmes à mots de poids faible

A l'instar de la théorie des nombres, la théorie des codes correcteurs d'erreurs offre aux cryptographes quelques problèmes NP-complets, tels celui de la détermination des poids d'un code linéaire et celui du décodage complet [BMvT78]. De plus, les instances de ces problèmes sont d'un maniement relativement commode puisqu'il s'agit le plus souvent d'objets d'algèbre linéaire. Aussi toute une classe de cryptosystèmes fondés sur la théorie des codes correcteurs a-t-elle émergé dès l'apparition des premiers systèmes à clef publique.

L'idée fondatrice de McEliece [McE78] est d'utiliser comme clef secrète un code linéaire très structuré, pour lequel on dispose d'un algorithme de décodage rapide, et de fournir comme clef publique un code équivalent, construit de façon à dissimuler la structure algébrique initiale et donc pour qu'il se comporte comme un code aléatoire. Ainsi, le système de chiffrement à clef publique de McEliece repose sur la difficulté de décoder un code de Goppa dont, seule, une matrice génératrice permutée est connue. En 1986, H. Niederreiter proposa un autre système de chiffrement exploitant la difficulté de trouver un mot de poids minimal de syndrome donné [Nie86], dont la sécurité est en fait équivalente à celle du système proposé par McEliece. Une autre voie consiste à utiliser la complexité de la recherche d'un mot de poids faible dans un code aléatoire pour construire des schémas d'identification à apport nul de connaissance. Initiés par Jacques Stern [Ste89a] et Marc Girault [Gir90], ces travaux ont abouti à des protocoles plus faciles à mettre en œuvre, tels ceux proposés par Jacques Stern à CRYPTO'93 [Ste93] et par Pascal Véron [Vér95b]. Tous ces systèmes fondés sur les codes présentent certes des inconvénients qui rendent leur utilisation moins agréable que celle du RSA : la taille des clefs est importante, et leur taux de transmission est relativement faible. Ils sont cependant beaucoup plus rapides, notamment en chiffrement, que les systèmes communément employés ; de plus ils constituent actuellement la seule alternative crédible aux cryptosystèmes à clef publique fondés sur la factorisation et le logarithme discret.

Je montrerai que tous ces systèmes — systèmes de chiffrement et schémas

d'identification — reposent en fait sur la difficulté de trouver un mot de poids donné faible dans un code linéaire ; c'est pourquoi je les regrouperai sous la terminologie *cryptosystèmes à mots de poids faible*. Ce premier chapitre est consacré à une présentation détaillée des plus connus d'entre eux.

1 Systèmes de chiffrement à mots de poids faible

Je décris ici les systèmes de chiffrement à clef publique présentés par McEliece [McE78] et Niederreiter [Nie86], qui sont tous deux équivalents d'un point de vue cryptographique. Je commencerai par décrire le principe de leur fonctionnement et reviendrai ensuite sur la famille de codes qu'ils utilisent.

1.1 Principe général du système de McEliece

Le système de McEliece utilise comme clef secrète un code linéaire binaire $\mathcal{C}$, de longueur n, dimension k et distance minimale d, pour lequel on connaît un algorithme rapide de décodage. Je noterai $t = \lfloor \frac{d-1}{2} \rfloor$ sa capacité de correction. Dans son article, McEliece choisit ce code parmi la famille des codes de Goppa irréductibles, mais je montrerai par la suite que ce choix peut être moins restrictif.

Le fonctionnement de ce système est résumé à la table 2.1.

Clef secrète :
- $\mathcal{C}$: code linéaire binaire $[n, k, d]$ choisi parmi une famille de codes Γ, représenté par exemple par une matrice génératrice.
- S : matrice aléatoire binaire inversible de taille $k \times k$.
- P : matrice aléatoire binaire de permutation de taille $n \times n$.

Clef publique :
G' : matrice $k \times n$ obtenue par le produit $G' = SGP$,
où G est une matrice génératrice du code $\mathcal{C}$.

Algorithme de chiffrement :
Texte chiffré correspondant à un message m de k bits : $c = mG' + e$,
où e est un vecteur binaire de longueur n et de poids t.

Algorithme de déchiffrement :
Multiplier le message chiffré par la matrice P^{-1} — pour obtenir $cP^{-1} = mSG + eP^{-1}$ —, puis utiliser l'algorithme de décodage rapide du code $\mathcal{C}$ pour retrouver mS, et finalement $m = (mS)S^{-1}$.

TAB. 2.1 –: *Système de chiffrement à clef publique de McEliece*

En réalité, la clef publique du système n'est autre qu'une matrice génératrice d'un autre code linéaire, $\mathcal{C}'$, *équivalent* au code $\mathcal{C}$ — 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 $\mathcal{C}'$ par le terme *code public*, par opposition au *code secret*, $\mathcal{C}$. 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 :

Définition 2.1 (Problème du décodage jusqu'à la distance w)
Entrée :

– *G : matrice binaire $k \times n$, de rang k.*

– *x : vecteur binaire de longueur n*

Problème : *Trouver, s'il existe, un vecteur binaire m de longueur k tel que*

$$d(x, mG) \leq w$$

1.1.1 Paramètres

J'ai repris ici la présentation originale de McEliece mais ce système peut plus généralement être mis en œuvre avec des codes secrets et publics définis sur un corps à q éléments.

Les paramètres originaux proposés par McEliece sont les suivants :

$$n = 1024,\ k = 524,\ d = 101$$

Un paramètre important pour un tel système est la taille des clefs publiques puisque celles-ci doivent être stockées dans un annuaire. Ici, il s'agit de matrices binaires à k lignes et n colonnes, qui nécessitent donc 65,5 Koctets. Ce coût élevé en mémoire est un des inconvénients majeurs du système.

Les paramètres suggérés par McEliece induisent un taux de transmission k/n de l'ordre de 51,2 %. C'est ce taux de transmission relativement faible qui est généralement rédhibitoire dans l'utilisation du système de McEliece, face à l'absence de redondance du RSA. Il est toutefois possible de l'augmenter de façon naturelle. En effet, le vecteur d'erreurs, e, peut jouer le rôle d'un "canal subliminal"dans la mesure où il peut contenir de l'information. Cette modification nécessite alors l'emploi d'un algorithme mettant en bijection les mots de longueur n de poids t et les entiers compris entre 1 et $\binom{n}{t}$. Malheureusement l'algorithme classique explicitant cette bijection [LCF90, Gui] ralentit considérablement les procédures de chiffrement et de déchiffrement. Il permet à ce prix d'inclure 284 bits d'information dans le vecteur d'erreurs — $\binom{1024}{50} > 2^{284}$ — et donc d'atteindre un taux de transmission de l'ordre de 80 %. Un compromis

entre le temps de calcul et le taux de transmission est néanmoins fourni par un algorithme approché proposé par Nicolas Sendrier, qui utilise un code de Huffman [Sen95a] ; il n'explicite pas entièrement la bijection entre les vecteurs d'erreurs et les messages de 284 bits mais il possède une complexité linéaire — il effectue en moyenne de l'ordre de $n + \log_2 \binom{n}{t}$ opérations binaires. Par ce biais, le vecteur d'erreurs peut contenir 276 bits d'information, ce qui porte le taux de transmission à 78 %. Toutefois, cette variante du système de McEliece semble dangereuse d'un point de vue cryptographique : la connaissance de quelques bits du texte clair peut en effet conduire à la détermination de certains bits du vecteur d'erreurs. Une diminution même minime du poids de e permet alors de décoder c, comme je le montrerai au chapitre 4.

1.1.2 Complexité des procédures de chiffrement et de déchiffrement

La complexité du chiffrement est celle de la procédure d'encodage d'un code $[n, k]$, c'est-à-dire en moyenne $\frac{nk}{2}$ opérations binaires, auxquelles il faut toutefois ajouter la complexité de la génération du vecteur d'erreurs. Dans toute la suite, nous supposerons que celle-ci a été effectuée à l'aide de l'algorithme proposé par Sendrier utilisant un code de Huffman. Le nombre d'opérations binaires effectuées en moyenne lors du chiffrement est donc

$$W_1^C = \frac{nk}{2} + \left(n + 3 \log_2 \binom{n}{t} \right)$$

Le déchiffrement dépend, lui, de l'algorithme de décodage utilisé pour les codes de la famille Γ. Dans le cas où Γ correspond aux codes de Goppa de longueur 2^m, l'algorithme utilisé est celui l'algorithme d'Euclide étendu ou celui de Berlekamp-Massey. La procédure complète nécessite successivement n opérations binaires pour la permutation de c, nt additions dans $\mathbb{F}_{2^m}$ pour le calcul du syndrome, $4t^2+2t$ multiplications dans $\mathbb{F}_{2^m}$ pour l'algorithme d'Euclide [Vér92, chapitre III], de l'ordre de $n(2t+1)$ multiplications dans $\mathbb{F}_{2^m}$ pour le calcul des racines du polynôme localisateur — que l'on peut ramener à des multiplications par α si l'on utilise la recherche de Chien [CCO69] —, et enfin $\frac{k^2}{2}$ opérations binaires pour la multiplication par S^{-1}. En résumé le nombre d'opérations binaires effectuées lors du déchiffrement est donné par

$$W_1^D = n + mnt + 4m^2t^2 + 2m^2t + mn(2t+1) + \frac{k^2}{2}$$

Les expressions du nombre d'opérations binaires par bit d'information transmis effectuées lors de ces deux procédures est donné à la table 2.4 (page 27), ainsi que leurs valeurs pour les paramètres proposés par McEliece.

1.1.3 Linéarité du système

Il est dès à présent possible de mettre en relief une première faiblesse dans la sécurité du système de McEliece, qui émane de la linéarité des codes utilisés. Elle concerne le cas où l'on chiffre à deux reprises le même message.

L'addition de deux chiffrés permet tout d'abord de déterminer avec une grande probabilité s'ils correspondent au même texte clair. En effet, l'addition bit-à-bit de deux textes chiffrés c_1 et c_2 quelconques produit un vecteur composé d'un mot xG' du code public et d'un vecteur-erreur $e_1 + e_2$ de poids au plus $2t$. Dans le cas où c_1 et c_2 correspondent au même texte clair, le poids de leur somme est donc inférieur ou égal à $2t$. Or, la probabilité que $w(c_1 + c_2) \leq 2t$ dans le cas général est majorée par

$$Pr[w(c_1 + c_2) \leq 2t] \leq Pr[w(xG') \leq 4t]$$

Cette probabilité dépend bien entendu de la distribution des poids du code $\mathcal{C}$ qui est le plus souvent inconnue. Néanmoins, si on l'approxime par celle d'un code aléatoire, c'est-à-dire

$$Pr[w(xG') = i] = \frac{\binom{n}{i}}{2^n}$$

on en déduit

$$Pr[w(c_1 + c_2) \leq 2t] \leq \frac{\sum_{i=2t+1}^{4t} \binom{n}{i}}{2^n}$$

Ainsi, si la somme de deux textes chiffrés par le système de McEliece, avec ses paramètres originaux, est de poids inférieur ou égal à $2t$, alors la probabilité pour que ces deux messages correspondent au même texte clair est supérieure à $1 - 10^{-89}$.

Or, après avoir constaté que $c_1 = mG' + e_1$ et $c_2 = mG' + e_2$ correspondaient au même texte clair, il est aisé de retrouver m. Il suffit pour cela de déterminer, à l'aide de la donnée de $c_1 + c_2$, un ensemble d'information ne contenant aucune position du support de e_1. Je propose alors l'algorithme suivant :

1. Choisir k positions d'information I du code public $\mathcal{C}'$ parmi les bits nuls de $c_1 + c_2$.

2. Soient $\bar{G}'$ et $\bar{c}_1$ les parties respectivement de la matrice G' et du vecteur c_1 correspondant aux positions de I.
   Si $w(c_1 + \bar{c}_1\bar{G}'G') = t$, alors $m = \bar{c}_1\bar{G}'$.
   Sinon, retourner à l'étape 1.

TAB. 2.2 –: *Algorithme permettant de retrouver un texte clair à partir de deux de ses chiffrés par le système de McEliece*

Proposition 2.2 *La connaissance de deux textes chiffrés par le système de McEliece correspondant au même texte clair m permet de déterminer m en* $\mathcal{O}\left(k^3 \frac{\binom{n-2t+2\frac{t^2}{n}}{k}}{\binom{n-2t+\frac{t^2}{n}}{k}}\right)$ *opérations binaires en moyenne.*

preuve : L'algorithme permet de trouver m si et seulement si l'ensemble d'information I ne contient aucune position d'erreurs, *i.e.* aucune position du support de e_1.

Le nombre de zéros du vecteur $c_1 + c_2$ est en moyenne $n - 2t + 2\frac{t^2}{n}$. En effet $w(c_1 + c_2) = w(e_1 + e_2) = 2t - 2w(e_1 \wedge e_2)$, et la probabilité pour qu'une position fasse à la fois partie du support de e_1 et de e_2 est égale à $\frac{t^2}{n^2}$. Parmi ces $n - 2t + 2\frac{t^2}{n}$ positions, seules $\frac{t^2}{n^2}$ correspondent donc à une position du support de e_1.

Ainsi la probabilité pour qu'un ensemble de positions choisies parmi les zéros de $c_1 + c_2$ ne contienne aucune position de $supp(e_1)$ est

$$P = \frac{\binom{n-2t+\frac{t^2}{n}}{k}}{\binom{n-2t+2\frac{t^2}{n}}{k}}$$

L'algorithme doit donc répéter en moyenne P^{-1} fois $\mathcal{O}(k^3)$ opérations, ce qui correspond bien à la complexité annoncée. □

Pour les paramètres proposés par McEliece, cela signifie donc que l'algorithme décrit à la table 2.2 doit être itéré en moyenne moins de 8 fois, ce qui est évidemment très faible.

1.2 Principe général du système de Niederreiter

Le système proposé par Niederreiter [Nie86] correspond à une approche duale du système de McEliece. Son principe est résumé à la table 2.3

Cette fois-ci, la clef publique est une matrice de parité du code public $\mathcal{C}'$. Le texte clair correspond à un vecteur de longueur n et de poids t, et le texte chiffré à son syndrome relativement au code $\mathcal{C}'$. 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 :

Définition 2.3 (Problème de la recherche des mots de poids inférieur à w d'un coset)
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*

$$eH = s$$

Clef secrète :
- $\mathcal{C}$: code linéaire binaire $[n, k, d]$ choisi parmi une famille de codes Γ, représenté par exemple par une matrice de parité.
- S : matrice aléatoire binaire inversible de taille $(n-k) \times (n-k)$.
- P : matrice aléatoire binaire de permutation de taille $n \times n$.

Clef publique :
H' : matrice $(n-k) \times n$ obtenue par le produit $H' = SHP$,
où H est une matrice de parité du code $\mathcal{C}$.

Algorithme de chiffrement :
Texte chiffré correspondant à un message m de longueur n et de poids t :
$c = m\,{}^tH'$.

Algorithme de déchiffrement :
Multiplier le message chiffré par la matrice ${}^tS^{-1}$ — pour obtenir $c\,{}^tS^{-1} = m\,{}^tP\,{}^tH$ —, puis utiliser l'algorithme de décodage rapide du code $\mathcal{C}$ pour retrouver $m\,{}^tP$, et finalement $m = (m\,{}^tP)\,{}^tP^{-1}$.

TAB. 2.3 –: *Système de chiffrement à clef publique de Niederreiter*

Il est alors aisé de montrer que les systèmes de McEliece et de Niederreiter sont équivalents du point de vue de leur sécurité [LDW94], puisque les problèmes du décodage jusqu'à la distance w et de la recherche de mots de poids inférieur à w dans un coset sont équivalents.

Proposition 2.4 *Les problèmes du décodage d'un code linéaire jusqu'à la distance w, décrit à la définition 2.1, et de la recherche de mots de poids inférieur ou égal à w dans un coset de ce code, décrit à la définition 2.3, sont équivalents.*

preuve :

– Le problème du décodage jusqu'à la distance w consiste, à partir d'un vecteur de la forme $x = mG + e$, à retrouver m . En multipliant les deux membres de cette équation par la transposée d'une matrice de parité H du code $\mathcal{C}$ (obtenue par une procédure d'élimination de Gauss), on a

$$s = x\,{}^tH = mG\,{}^tH + e\,{}^tH = e\,{}^tH$$

Soit Φ un algorithme permettant de résoudre le problème de la recherche de mots de poids inférieur ou égal à w dans un coset. Il permet alors de calculer le vecteur-erreur e à partir de la donnée de s, et par conséquent de retrouver m.

– Inversement, le problème de la recherche de mots de poids inférieur à w dans un coset consiste, à partir de $s = e\,{}^tH$, à trouver e où e est de poids

inférieur ou égal à w. Soit I un ensemble de $n-k$ coordonnées tel que la matrice carrée $H_{|I}$ soit inversible. Le vecteur x dont la restriction à I vaut $s\ ({}^tH_{|I})^{-1}$ et dont tous les autres bits sont nuls a pour syndrome s, relativement au code $\mathcal{C}$. Or, l'ensemble des mots de syndrome s est un translaté de $\mathcal{C}$ dont un mot de poids inférieur ou égal à w est e. Le vecteur x s'exprime donc comme la somme de e et d'un mot du code $\mathcal{C}$. Un algorithme Ψ capable de décoder jusqu'à la distance w permet donc de retrouver e à partir de la donnée de x.

□

Corollaire 2.5 (Equivalence entre les systèmes de McEliece et de Niederreiter) *Trouver un algorithme permettant de déchiffrer les messages obtenus avec le système de Niederreiter est équivalent à trouver un algorithme permettant de déchiffrer les messages obtenus avec le système de McEliece, lorsque ces deux systèmes utilisent le même code public $\mathcal{C}'$.*

De la même façon, retrouver la clef secrète du système de Niederreiter à partir de la clef publique est équivalent à retrouver la clef secrète du système de McEliece à partir de sa clef publique.

1.2.1 Paramètres

Dans son article original, 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 $\mathbb{F}_{16}$, soit avec un code de Reed-Solomon [30,12,19] sur $\mathbb{F}_{31}$. Les paramètres de ces codes sont évidemment trop petits pour que le système résiste à la cryptanalyse, comme l'ont montré Brickell et Odlyzko [BO92] 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.

Proposition 2.6 *Soit H' la clef publique d'un système de chiffrement suivant le schéma de Niederreiter et $\bar{H}' = UH'$ une forme systématique de H'. Déchiffrer le système de Niederreiter de clef publique $\bar{H}'$ est équivalent à déchiffrer le système de Niederreiter de clef publique H'.*

preuve : Soit $c = m\,{}^tH'$ un message chiffré par le système de Niederreiter avec la clef publique H'. En multipliant ce message par la matrice ${}^tU$, on obtient

$$x = c\,{}^tU = m\,{}^t\bar{H}'$$

Tout oracle capable de déchiffrer le système de Niederreiter de clef publique $\bar{H}'$ permet alors de déterminer le texte clair m. □

Si l'on utilise des codes [1024,524,101], le taux de transmission du système de Niederreiter est d'autre part supérieur à celui du système de McEliece puisqu'il est donné par

$$\tau = \frac{\log_2 \binom{n}{t}}{n-k}$$

Il est donc de l'ordre de 56,8 %. La figure 2.1 donne les taux de transmission respectifs des systèmes de McEliece et de Niederreiter en fonction de la dimension des codes de Goppa irréductibles de longueur 1024 utilisés.

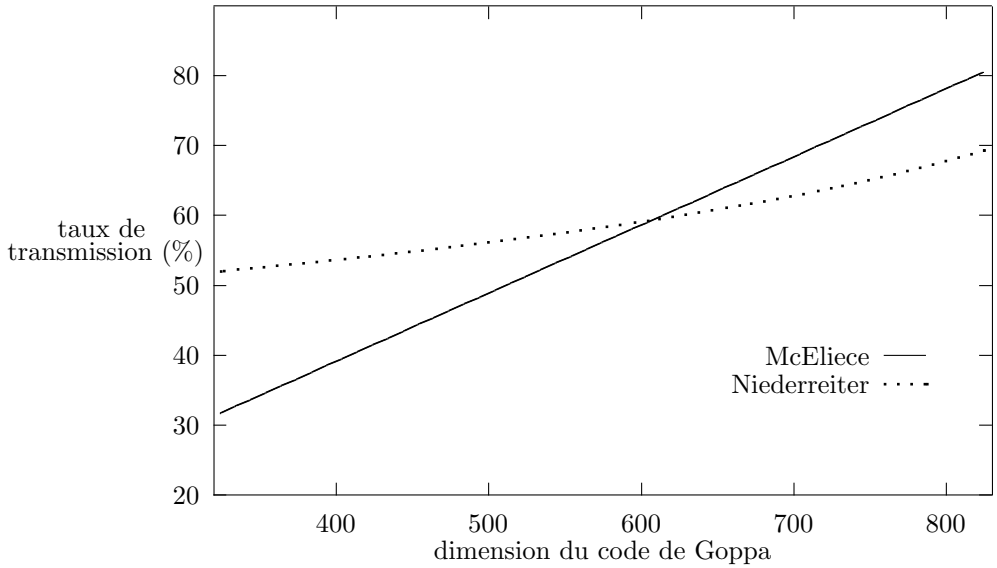


FIG. 2.1 –: *Evolution du taux de transmission des systèmes de McEliece et de Niederreiter en fonction de la dimension des codes de Goppa irréductibles utilisés en longueur 1024*

Un autre avantage du système de Niederreiter est qu'il n'est pas sujet à l'attaque présentée à la table 2.2. Contrairement au système de McEliece, il produit un chiffrement complètement 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.

1.2.2 Complexité des procédures de chiffrement et déchiffrement

Une autre différence entre les systèmes de McEliece et de Niederreiter réside dans la complexité de leurs procédures respectives de chiffrement et de déchiffrement. La représentation des messages clairs par des vecteurs de longueur n et de poids t nécessite en effet dans le système de Niederreiter, à la fois lors du chiffrement et du déchiffrement, l'utilisation d'un algorithme explicitant la bijection entre les vecteurs de poids t et les mots de $\log_2 \binom{n}{t}$ bits, tels ceux cités précédemment. Je supposerai que l'algorithme choisi est celui de Nicolas Sendrier.

Par contre, la mise sous forme systématique de la matrice publique H' et le fait que le vecteur m à multiplier ait un poids faible réduisent considérablement le coût de la multiplication matrice-vecteur, qui ne nécessite en moyenne que $\frac{(n-k)kt}{n}$ opérations binaires, si l'on estime en moyenne que $\frac{kt}{n}$ bits de m correspondent à la partie redondante de H'.

$$W_2^C = \frac{(n-k)kt}{n} + n + 3\log_2 \binom{n}{t}$$

Le coût du déchiffrement est pratiquement identique à celui du système de McEliece, à ceci près que le syndrome est obtenu directement par le message chiffré et que la multiplication par la matrice S^{-1} requiert cette fois $\frac{(n-k)^2}{2}$ opérations. De plus, il faut y ajouter le calcul du texte clair à partir du message de poids t qui n'apparaissait pas dans la procédure de déchiffrement du système de McEliece.

$$W_2^D = n + 4m^2t^2 + 2m^2t + mn(2t+1) + \frac{(n-k)^2}{2} + n + 3\log_2 \binom{n}{t}$$

La table 2.4 compare donc les systèmes de McEliece et de Niederreiter en termes de taille de clefs publiques, de taux de transmission et de complexité du chiffrement et du déchiffrement par bit d'information transmis. Je donne à titre indicatif les valeurs correspondant au système RSA utilisé avec un modulo de 1024 bits et l'exposant public 17 (qui est l'exposant employé dans PGP). Je suppose ici que la méthode utilisée lors du chiffrement et du déchiffrement RSA pour multiplier deux entiers de 2^m bits est celle de Karatsuba (cf. [Knu69, page 279]) dont le coût est de l'ordre de 3^{m+1} opérations binaires.

Ces résultats montrent que, dans la plupart des cas, il est préférable d'utiliser le système de Niederreiter au système de McEliece. On constate également que les systèmes à mots de poids faible sont beaucoup plus rapides que le RSA (et les autres systèmes reposant sur la théorie des nombres), surtout en chiffrement.

Une particularité de ces cryptosystèmes est qu'ils sont complètement asymétriques. Cette asymétrie peut être avantageuse dans certaines applications car le chiffrement nécessite très peu d'opérations et peut donc être effectué par une carte, mais elle constitue un obstacle majeur à la construction d'une signature digitale selon le modèle développé par Diffie et Hellman [DH76] et utilisé par la suite avec le chiffrement RSA.

	McEliece code $[n, k, 2t+1]$	Niederreiter code $[n, k, 2t+1]$	RSA modulo de n bits
taille de la clef publique	kn	$k(n-k)$	$2n$
	67 072 octets	32 750 octets	256 octets
nb de bits d'information transmis par chiffrement	k	$\alpha \simeq \log_2(\binom{n}{t})$	n
	512	276	1024
taux de transmission	$\frac{k}{n}$	$\frac{\log_2 \binom{n}{t}}{n-k}$	1
	51,17 %	56,81 %	100 %
nb d'opérations binaires du chiffrement par bit d'information	$\frac{n}{2} + \frac{n}{k}$	$\frac{(n-k)kt}{\alpha n} + \frac{n}{\alpha}$	$125\frac{3^{m-1}}{2^m}$
	513,9	50,1	2402,7
nb d'opérations binaires du déchiffrement par bit d'information	$\frac{W_1^D}{k}$	$\frac{W_2^D}{\alpha}$	$\frac{25}{2}3^{m-1}$
	5140,3	7863,3	738 112,5

avec $W_1^D = n + mnt + 4m^2t^2 + 2m^2t + mn(2t+1) + \frac{k^2}{2}$
et $W_2^D = 2n + 4m^2t^2 + 2m^2t + mn(2t+1) + \frac{(n-k)^2}{2}$.

TAB. 2.4 –: *Comparaison des performances des systèmes de McEliece et de Niederreiter : les valeurs numériques données pour les systèmes de McEliece et de Niederreiter correspondent à l'utilisation d'un code [1024,524,101], et pour le RSA à un modulo pq de 1024 bits*

1.3 Conditions sur la famille de codes secrets

Comme tout système de chiffrement à clef publique, le système de McEliece (ou celui de Niederreiter) est sujet à deux grandes catégories d'attaques :

- *les algorithmes recherchant la clef secrète* : le problème est alors de retrouver la structure initiale du code secret à partir d'une matrice génératrice (ou de parité) d'un code équivalent.

- *les algorithmes de déchiffrement* : il s'agit ici de retrouver le texte clair à partir du texte chiffré, ce qui revient à décoder le code public.

Le premier type de cryptanalyse conditionne donc la nature de la famille de codes secrets Γ que l'on peut employer. Elle doit en effet satisfaire les trois propriétés suivantes :

1. Pour une longueur, une dimension et une distance minimale données, la famille Γ doit être suffisamment grande pour que toute attaque énumérative soit hors de portée.

2. Tout code de Γ doit pouvoir être décodé par un algorithme rapide.

3. Aucune information sur la structure d'un code de Γ ne doit pouvoir être déduite de la connaissance d'une matrice génératrice d'un code équivalent.

La première condition, sur la taille de la famille Γ, vise à se prémunir d'une attaque qui consisterait à énumérer tous les codes de Γ jusqu'à trouver celui qui est équivalent au code public. Ceci peut être mis en œuvre par un algorithme conçu par Nicolas Sendrier qui permet, à partir de matrices génératrices de deux codes, de déterminer s'ils sont équivalents et, dans le cas positif, d'exhiber la permutation [Sen96]. Il utilise pour cela comme invariant du code la distribution des poids de son hull — le hull d'un code est l'intersection du code et de son dual. Pour les paramètres proposés par McEliece, il nécessite entre 7 et 80 secondes sur une station DEC 3000/900 pour calculer la permutation entre deux codes, et encore moins de temps pour constater que les codes ne sont pas équivalents. Ces performances imposent donc une valeur élevée du cardinal de Γ. L'algorithme de Sendrier ne s'applique cependant pas à la variante du système de McEliece imaginée par Sidelnikov [Sid94] qui utilise comme unique code secret le code de Reed-Muller d'ordre 3 (de longueur 1024 ou 2048) et comme clef publique une matrice génératrice d'un code équivalent. En effet, l'algorithme permettant de retrouver la permutation entre deux codes ne fonctionne pas si le groupe d'automorphismes du code est trop gros, ce qui est le cas pour les codes de Reed-Muller. Malgré tout, l'utilisation d'un seul code secret, qui, de surcroît, est très structuré, laisse planer un doute quant à la sécurité du système de Sidelnikov.

La troisième condition est la plus restrictive. Ainsi quantité de familles de codes bien connues sont à proscrire : les codes concaténés initialement proposés par Niederreiter en raison de la rapidité de leur algorithme de décodage ne vérifient pas cette propriété, comme l'a montré Nicolas Sendrier [Sen94, Sen95b]. Les codes de Reed-Solomon généralisés ne conviennent pas non plus puisqu'il est possible de retrouver leur structure à partir de la clef publique en utilisant l'algorithme de Sidelnikov-Shestakov [SS92].

La famille des codes de Goppa irréductibles a, elle, résisté jusqu'à ce jour à ce type d'attaques. La classe plus générale des codes alternants peut également être utilisée.

1.4 Une famille de codes secrets particulière : les codes de Goppa irréductibles

Un code de Goppa sur $\mathbb{F}_q$ est déterminé par un polynôme g de $\mathbb{F}_{q^m}[X]$, appelé son *polynôme de Goppa*, et un ensemble d'éléments de $\mathbb{F}_{q^m}$ ne contenant pas de racines de g, son *support*.

Définition 2.7 (Codes de Goppa) *Soit m un entier positif, g un polynôme de $\mathbb{F}_{q^m}[X]$ de degré t et $L = (\alpha_1, \ldots, \alpha_n)$ un ensemble d'éléments de $\mathbb{F}_{q^m}$ tel que, pour tout α_i dans L, $g(\alpha_i) \neq 0$. Le* code de Goppa *de polynôme g et de support L, noté $\Gamma(L, g)$, est l'ensemble des vecteurs $(c_1, \ldots, c_n)$ de $\mathbb{F}_q^n$ vérifiant*

$$\sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} \equiv 0 \bmod g(x)$$

Un code de Goppa est donc linéaire. Une manière simple de le représenter est d'utiliser une forme particulière de sa matrice de parité obtenue en remplaçant chaque élément de $\mathbb{F}_{q^m}$ par le vecteur-colonne de $\mathbb{F}_q^m$ correspondant dans la matrice

$$\hat{H} = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_n^2 \\ \vdots & \vdots & & \vdots \\ \alpha_1^t & \alpha_2^t & \ldots & \alpha_n^t \end{pmatrix} \begin{pmatrix} g(\alpha_1)^{-1} & & & & 0 \\ & g(\alpha_2)^{-1} & & & \\ & & g(\alpha_3)^{-1} & & \\ & & & \ddots & \\ 0 & & & & g(\alpha_n)^{-1} \end{pmatrix} \tag{2.1}$$

La dimension de ce code de Goppa est donc minorée par

$$k \geq n - mt$$

Par cette représentation il apparaît immédiatement que tout code de Goppa est un *code alternant*. Il est en particulier la restriction à $\mathbb{F}_q$ d'un code de Reed-Solomon généralisé [MS77].

Les codes de Goppa binaires constituent un cas important car ils peuvent être décrit à l'aide de polynômes à coefficients dans $\mathbb{F}_{2^m}$.

Proposition 2.8 (Polynôme localisateur d'un mot d'un code de Goppa binaire) *Soit g un polynôme de degré t de $\mathbb{F}_{2^m}[X]$ et $L = (\alpha_1, \ldots, \alpha_n)$ un ensemble d'éléments de $\mathbb{F}_{2^m}$ ne contenant pas de racines de g. Soit $c = (c_1, \ldots, c_n)$ un vecteur de $\mathbb{F}_2^n$, S_c son support et $\sigma_c(x) = \prod_{i \in S}(x - \alpha_i)$ son polynôme localisateur. Alors c est un mot du code $\Gamma(L, g)$ si et seulement g divise la dérivée de son polynôme localisateur σ_c.*

preuve : Par définition, c est un mot du code $\Gamma(L, g)$ si et seulement si

$$\mathcal{R}_c(x) = \sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} \equiv 0 \bmod g(x)$$

Or, cette fraction rationnelle s'exprime en fonction du polynôme localisateur de c :

$$\mathcal{R}_c(x)\sigma_c(x) = \sum_{i \in S_c} c_i \prod_{\substack{j \in S_c \\ j \neq i}} (x - \alpha_j) = \sigma_c'(x)$$

Comme par définition le polynôme de Goppa g n'a pas de racines parmi les éléments du support, il est premier avec σ_c. On en déduit donc que c est un mot du code de Goppa si et seulement $\sigma_c'(x) \equiv 0 \bmod g(x)$. □

En caractéristique 2, la dérivée d'un polynôme ne contient que des termes pairs — les termes impairs proviennent de la dérivation d'un terme pair et sont donc précédés d'un coefficient pair. Elle peut donc s'écrire comme un carré parfait. Aussi la condition g divise σ_c' équivaut-elle à dire que le carré parfait de plus bas degré multiple de g, noté $\bar{g}$, divise σ_c'. Lorsque g est sans facteurs multiples, $\bar{g}$ est alors égal à g^2. Les mots de $\Gamma(L, g)$ sont alors exactement ceux dont le polynôme localisateur σ s'écrit

$$\sigma(x) = [p(x)]^2 + x[m(x)g(x)]^2$$

avec m et p dans $\mathbb{F}_{2^m}[X]$.

Le degré de σ_c étant égal au poids de c, on en déduit immédiatement une borne sur la distance minimale de $\Gamma(L, g)$.

Proposition 2.9 (Distance minimale d'un code de Goppa dont le polynôme est sans facteurs multiples) *La distance minimale d d'un code de Goppa dont le polynôme g est sans facteurs multiples vérifie*

$$d \geq 2\, deg(g) + 1$$

McEliece préconise pour son système l'utilisation des codes de Goppa binaires irréductibles de support L égal à $\mathbb{F}_{2^{10}}$ tout entier dont le polynôme de Goppa est de degré 50 sur $\mathbb{F}_{2^{10}}$. Mais on peut, au vu de la proposition précédente, utiliser plus généralement les codes de Goppa dont les polynômes sont de degré 50, sans racines dans $\mathbb{F}_{2^{10}}$ et sans facteurs multiples.

Dénombrement des codes de Goppa dont le polynôme est sans facteurs multiples et sans racines dans $\mathbb{F}_{q^m}$

En remarquant que deux polynômes g et γg avec $\gamma \in \mathbb{F}_{q^m}^\star$ génèrent le même code de Goppa, on peut dénombrer les clefs secrètes du système de McEliece en comptant les polynômes unitaires de degré 50 de $\mathbb{F}_{2^{10}}[X]$, sans racines dans $\mathbb{F}_{2^{10}}$ et sans facteurs multiples.

Proposition 2.10 (Nombre de polynômes unitaires sans facteurs multiples et sans racines dans $\mathbb{F}_{q^m}$) *Le nombre de polynômes de $\mathbb{F}_{q^m}[X]$ unitaires de degré t, sans facteurs multiples et sans racines dans $\mathbb{F}_{q^m}$ est donné par*

$$\sum_{i=0}^{t}(-1)^i\binom{q^m-1+i}{i}q^{m(t-i)} \; - q^m\sum_{i=0}^{t-2}(-1)^i\binom{q^m-1+i}{i}q^{m(t-i-2)}$$

preuve : Le nombre de polynômes unitaires de degré t de $\mathbb{F}_{q^m}[X]$, sans facteurs multiples et sans racines dans $\mathbb{F}_{q^m}$ correspond au nombre de produits distincts de polynômes irréductibles unitaires de $\mathbb{F}_{q^m}[X]$ de degré au moins 2, dont la somme des degrés vaut t.

Le nombre de polynômes unitaires irréductibles sur $\mathbb{F}_{q^m}$ de degré i est donné par

$$N_{q^m}(i) = \frac{1}{i}\sum_{d/i}\mu(\frac{i}{d})q^{mi}$$

où μ est la fonction de Möbius [Ber68, théorème 3.43]. Par conséquent, la série génératrice des polynômes unitaires de $\mathbb{F}_{q^m}[X]$ sans racines dans $\mathbb{F}_{q^m}$ et sans facteurs multiples est

$$s(x) = \prod_{i>1}(1+x^i)^{N_{q^m}(i)}$$

D'après [Ber68, équation 3.33], $\prod_{i\geq 1}(1-x^i)^{N_{q^m}(i)} = 1-q^mx$. On en déduit donc pour la série s

$$s(x) = \prod_{i>1}\frac{(1-x^{2i})^{N_{q^m}(i)}}{(1-x^i)^{N_{q^m}(i)}} = \frac{1-q^mx^2}{(1-x^2)^{q^m}}\;\frac{(1-x)^{q^m}}{1-q^mx} = \frac{1-q^mx^2}{(1-q^mx)(1+x)^{q^m}}$$

Il faut alors calculer le coefficient du terme de degré t de la série, noté s_t. Il suffit pour cela d'écrire

$$\frac{1}{(1+x)^{q^m}} = \sum_{n\geq 0}(-1)^n\binom{q^m-1+n}{n}x^n$$

On en déduit

$$\frac{1}{(1-q^mx)(1+x)^{q^m}} = \sum_{n\geq 0}x^n\sum_{i=0}^{n}(-1)^i\binom{q^m-1+i}{i}q^{m(n-i)}$$

Et par conséquent

$$s_t = \sum_{i=0}^{t}(-1)^i \binom{q^m - 1 + i}{i} q^{m(t-i)} \; - q^m \sum_{i=0}^{t-2}(-1)^i \binom{q^m - 1 + i}{i} q^{m(t-i-2)}$$

□

Le nombre de tels polynômes de degré 50 sur $\mathbb{F}_{2^{10}}$ est donc égal à $2^{498,55}$, c'est-à-dire 36,8 % des polynômes unitaires de $\mathbb{F}_{2^{10}}$. On multiplie ainsi par un facteur de l'ordre de 18 le cardinal de la famille de codes secrets proposée par McEliece. Ce résultat montre d'une part que cette famille met le système à l'abri de toute attaque du type de celle décrite au paragraphe précédent, nécessitant une énumération des codes secrets, et également qu'il est très rapide de trouver, par tirage aléatoire parmi les polynômes unitaires de degré 50 de $\mathbb{F}_{2^{10}}[X]$, un polynôme permettant de construire un code secret.

1.5 Attaques induites par l'utilisation des codes de Goppa

Il convient maintenant de se demander si l'utilisation des codes de Goppa dans le système de McEliece n'introduit pas certaines failles dans sa sécurité. Cette question revient à s'interroger sur l'existence de certaines attaques qui exploiteraient certaines propriétés structurelles des codes de Goppa. Comme à l'habitude, ces attaques peuvent avoir deux finalités :

- déduire la structure d'un code de Goppa de la donnée d'une matrice génératrice (ou de parité) permutée.
- trouver un algorithme rapide permettant de décoder un code de Goppa permuté.

Une solution directe du premier problème semble hors de portée. En effet, toute la structure algébrique d'un code de Goppa est héritée de celle du code de Reed-Solomon généralisé correspondant ; elle résulte donc de propriétés sur le corps $\mathbb{F}_{2^m}$. Or, le changement de base introduit dans le système de McEliece par la multiplication de la matrice génératrice (ou de parité) par une matrice inversible S fait complètement disparaître la structure qui existait sur $\mathbb{F}_{2^m}$. Contrairement à une idée communément admise [Hei87, page 29], cette matrice S a donc un rôle cryptographique fondamental. Supposons en effet que l'on utilise le système de Niederreiter avec pour clef publique une matrice $H' = HP$ où P est une matrice de permutation et H une matrice de parité du code secret $\Gamma(\mathbb{F}_{2^m}, g)$ obtenue par la méthode classique, c'est-à-dire par l'expansion des éléments de $\mathbb{F}_{2^m}$ en vecteurs-colonne dans la matrice $\hat{H}$ de l'équation (2.1). En retrouvant l'isomorphisme utilisé entre $\mathbb{F}_{2^m}$ et $\mathbb{F}_2^m$, on pourrait ramener facilement H' à une matrice $\hat{H}'$ à coefficients dans $\mathbb{F}_{2^m}$, déduite de $\hat{H}$ par permutation des colonnes. Pour déterminer cette permutation, il suffit de diviser la seconde ligne de $\hat{H}'$ par la première puisque l'on obtient ainsi l'image du support $\mathbb{F}_{2^m}$ par la permutation.

L'existence de la matrice inversible S rend donc impossible toute identification "directe"du code de Goppa à partir d'une matrice génératrice d'un code équivalent. La seule attaque structurelle qui soit envisageable pour retrouver la structure du code secret consisterait alors à définir un invariant permettant de classifier les codes de Goppa — *i.e.* une propriété qui soit conservée par permutation des coordonnées. De cette façon, on pourrait, à partir du code public, déterminer une classe de codes beaucoup plus restreinte que Γ à laquelle appartiendrait nécessairement le code secret. Le problème essentiel est qu'une telle attaque n'est réaliste que si l'invariant peut être déterminé extrêmement rapidement — ce qui n'est pas le cas des invariants usuels, telle la distribution des poids du code ou même celle de son hull —, ou si la famille de codes Γ peut être partitionnée par certaines relations d'équivalence.

Le second problème évoqué est celui du décodage rapide d'un code de Goppa permuté. La question posée est alors : un code équivalent à un code de Goppa, tels ceux définis par McEliece, a-t-il une structure bien connue qui permette de le décoder facilement ? Une solution au problème du décodage apparaît immédiatement dans le cas où le code public est, lui aussi, un code de Goppa. Cette possibilité fut évoquée par Adams et Meijer qui, après une rapide estimation probabiliste, conclurent qu'elle ne se réalisait jamais [AM87]. Cette affirmation est manifestement erronée puisque le groupe d'automorphismes, non trivial, des codes de Reed-Solomon généralisés induit l'existence de permutations qui transforment un code de Goppa en un autre.

Proposition 2.11 *Soit g un polynôme de $\mathbb{F}_{2^m}[X]$ et $L = (\alpha_1, \ldots, \alpha_n)$ un sous-ensemble de $\mathbb{F}_{2^m}$. La permutation du support $(\alpha_1, \ldots, \alpha_n) \mapsto (\pi(\alpha_1), \ldots, \pi(\alpha_n))$ avec*

$$\pi(\alpha_i) = a\alpha_i^{2^j} + b \quad \text{où } a, b \in \mathbb{F}_{2^m},\ a \neq 0,\ j \geq 0$$

transforme le code de Goppa binaire $\Gamma(L, g)$ en un code de Goppa $\Gamma(\pi(L), h)$ où h est le polynôme obtenu en remplaçant chaque racine u du polynôme g dans son corps de décomposition par $\pi(u)$.

Il convient de remarquer que le polynôme de Goppa h du code permuté a le même degré que celui de g et que la propriété d'irréductibilité est également conservée. On en déduit par exemple que tous les codes de Goppa irréductibles 2-correcteurs de longueur 2^m sont équivalents [Mor79].

Ainsi, comme l'a remarqué Gibson [Gib91b], parmi les $2^m!$ permutations possibles des coordonnées d'un code de Goppa de longueur 2^m, au moins $m2^m(2^m - 1)$ produisent un code de Goppa. Pour les paramètres de McEliece, ces clefs faibles ne représentent toutefois qu'une proportion de 2.10^{-2629} %.

Une autre conséquence de la proposition 2.11 est la détermination de classes d'équivalence de codes de Goppa : en effet, l'existence de permutations qui, à un code de Goppa, associent un autre code Goppa de mêmes paramètres conduit à définir des classes de polynômes générant des codes équivalents. L'intérêt

principal de ce travail entamé par Gibson [Gib91a] est de classifier les codes de Goppa ce qui pourrait entre autres faciliter le calcul d'invariants.

Soit $n = 2^m$. Notons $\mathcal{P}_t$ l'ensemble des polynômes unitaires de degré t de $\mathbb{F}_{2^m}[X]$ et $\mathcal{L}$ l'ensemble des permutations définies à la proposition 2.11. L'ensemble $\mathcal{L}$ peut s'écrire comme le produit de deux sous-groupes, $\mathcal{L} = \mathcal{SA}$ où $\mathcal{S}$ est le groupe engendré par le Frobenius $\sigma : x \mapsto x^2$ et $\mathcal{A}$ est le groupe des transformations affines $x \mapsto ax + b,\ a, b \in \mathbb{F}_{2^m},\ a \neq 0$. On associe à $\mathcal{L}$ la relation d'équivalence suivante sur $\mathcal{P}_t$: deux polynômes g et h sont dits équivalents s'il existe une permutation π de $\mathcal{L}$ telle que les racines de h sont images par π des racines de g. Je noterai alors abusivement $h = \pi(g)$. Deux polynômes équivalents génèrent donc deux codes de Goppa équivalents au sens usuel. La réciproque n'est cependant pas démontrée puisque l'on ne sait pas si seules les permutations de $\mathcal{L}$ produisent des codes de Goppa équivalents.

Pour faciliter le dénombrement des classes d'équivalence sur $\mathcal{P}_t$, Gibson est amené à définir un ensemble particulier de polynômes, qu'il nomme *polynômes réguliers d'ordre 1*. Pour tout polynôme p de $\mathcal{P}_t$, je note p_i le coefficient du terme x^i de $p(x)$.

Proposition 2.12 (Polynômes réguliers d'ordre 1) [Gib91a] *Soit P le sous-ensemble de $\mathcal{P}_t$ suivant*

- *Si t est impair*

$$P = \{p \in \mathcal{P}_t,\ p_{t-1} = 0 \text{ et } p_{t-2} = 1\}$$

- *Si t est pair et $t/2$ est pair*

$$P = \{p \in \mathcal{P}_t,\ p_{t-1} = 1 \text{ et } p_{t-2} = 0\}$$

- *Si t est pair et $t/2$ est impair*

$$\begin{aligned} P \quad = \quad & \{p \in \mathcal{P}_t,\ p_{t-1} = 1, p_{t-3} \neq k \text{ et } p_{t-4} = k(p_{t-2} + p_{t-2}^2)\} \\ & \cup \{p \in \mathcal{P}_t,\ p_{t-1} = 0, p_{t-2} = 0 \text{ et } p_{t-4} = 1\} \end{aligned}$$

  où $k = \frac{t-2}{4} \bmod 2$.

Dans chacun de ces cas, P vérifie $|P| = n^{t-2}$ et

$$\forall p \in P,\ \forall \pi \in \mathcal{L},\ \ \pi(p) \in P \text{ ssi } \pi \in \mathcal{S}$$

Cette propriété induit immédiatement que le cardinal d'une classe d'équivalence qui contient un élément p de P est $cn(n-1)$ où c est la taille de l'orbite $\mathcal{S}p$. L'ensemble des polynômes Q ainsi classifiés, *i.e.* des polynômes équivalents à un polynôme de P, est donc de taille $n^t - n^{t-1}$. Il est également possible de compter le nombre de classes parmi ces $n^t - n^{t-1}$ polynômes :

Proposition 2.13 [Gib91a] *Les polynômes de Q se répartissent en exactement $N_t(m)$ classes d'équivalence où*

$$N_t(m) = \frac{1}{m} \sum_{d/m} \phi(\frac{m}{d}) 2^{d(t-2)}$$

Le nombre de classes d'équivalence de cardinal $cn(n-1)$ est

$$\begin{array}{rcll} V_t(c) & = & \frac{1}{c}\sum_{d/c}\mu(\frac{c}{d})2^{d(t-2)} & \textit{si } c \textit{ divise } m \\ & = & 0 & \textit{sinon} \end{array}$$

où μ est la fonction de Möbius et ϕ l'indicatrice d'Euler.

preuve : Le cardinal de l'orbite d'un polynôme sous l'action de $\mathcal{S}$ étant nécessairement un diviseur de m, le nombre de polynômes de P se décompose en

$$\sum_{c/m} cV_t(c) = 2^{m(t-2)}$$

En utilisant la formule d'inversion de Möbius [HW38, théorème 267], on obtient

$$V_t(c) = \frac{1}{c}\sum_{d/c}\mu(\frac{c}{d})2^{d(t-2)}$$

Le nombre de classes d'équivalence, $N_t(m)$, correspond donc à la somme des $V_t(c)$, c'est-à-dire

$$\begin{array}{rcl} N_t(m) & = & \sum_{c/m} V_t(c) = \sum_{c/m}\frac{1}{c}\sum_{d/c}\mu(\frac{c}{d})2^{d(t-2)} \\ & = & \sum_{d/m} 2^{d(t-2)}\sum_{b/a}\frac{1}{bd}\mu(b) \end{array}$$

où $m = ad$ et $c = bd$. Or, $\sum_{b/a}\frac{a}{b}\mu(b) = \phi(a)$ d'après [HW38, équation 16.3.1]. D'où

$$N_t(m) = \sum_{d/m}\frac{2^{d(t-2)}}{d}\,\frac{d}{m}\phi(\frac{m}{d})$$

□

On en déduit donc que pour les paramètres proposés par McEliece, les polynômes de Goppa de degré 50 sur $\mathbb{F}_{2^{10}}$ se répartissent en plus de $2^{476,68}$ classes d'équivalence, ce qui signifie que, dans une tentative de recherche du code secret en calculant un invariant, le nombre de polynômes à examiner serait divisé par 2^{23}. Ce calcul ne permet certes pas de dénombrer exactement les classes de codes secrets équivalents, puisque je n'ai pas distingué quelles étaient les classes correspondant à des polynômes irréductibles, et qu'il peut exister d'autres permutations que celles de $\mathcal{L}$ produisant des codes équivalents. Néanmoins, à la lumière du cas particulier des Goppa 3-correcteurs que je vais étudier maintenant, ce chiffre laisse penser que le nombre de classes d'équivalence de codes secrets du système de McEliece reste de toute façon très élevé.

Exemple 2.14: Classification des codes de Goppa dont le polynôme est de degré 3

L'ensemble des polynômes P défini à la proposition 2.12 pour le degré 3 est réduit à

$$P = \{x^3 + x + a,\ a \in \mathbb{F}_{2^m}\}$$

Le cardinal de la classe d'équivalence d'un polynôme de P est donc égal à $cn(n-1)$ où c est l'ordre cyclotomique de a. L'ensemble Q des polynômes qui sont équivalents à un polynôme de P et sont ainsi classifiés est donc

$$Q = \{x^3 + ax^2 + bx + c,\ b \neq a^2\}$$

Les polynômes restant à classifier sont donc ceux de la forme $p(x) = x^3 + ax^2 + a^2x + b$. Parmi eux, on distingue :

- la classe d'équivalence de $x^3 = \{(x + a)^3,\ a \in \mathbb{F}_{2^m}\}$, contenant n éléments.
- la classe d'équivalence de $x^3 + 1 = \{x^3 + ax^2 + a^2x + a^3 + b^3,\ a, b \in \mathbb{F}_{2^m}, b \neq 0\}$.
  Si m est impair, elle contient $n(n-1)$ éléments puisque $x \mapsto x^3$ est une application bijective. Par contre, si m est pair, elle ne contient que $\frac{n(n-1)}{3}$ éléments.
- si m est pair, la classe d'équivalence de $x^3 + \alpha$, où α est un élément primitif de $\mathbb{F}_{2^m}$, est $\{x^3 + ax^2 + a^2x + a^3 + \alpha b^3,\ a, b \in \mathbb{F}_{2^m}, b \neq 0\} \cup \{x^3 + ax^2 + a^2x + a^3 + \alpha^2 b^3,\ a, b \in \mathbb{F}_{2^m}, b \neq 0\}$ et contient $\frac{2n(n-1)}{3}$ éléments.

J'en déduis donc une classification complète des codes de Goppa $\Gamma(\mathbb{F}_{2^m}, g)$ où g est un polynôme de degré 3, résumée à la table 2.5.

Parmi ces classes, seules celle de $x^3 + \alpha$ quand m est pair, et certaines de celles des $x^3 + x + \alpha^i$ correspondent à des polynômes irréductibles. Une condition nécessaire pour que $x^3 + x + a$ soit irréductible est que $Tr(a) = Tr(1)$. On peut en déduire que le nombre de polynômes irréductibles parmi les $x^3 + x + a$ est $\frac{n+1}{3}$ si m est pair, et $\frac{n-1}{3}$ si m est impair.

Ainsi, les tables 2.6 et 2.7 donnent la classification de tous les codes de Goppa irréductibles 3-correcteurs de longueur respectivement 128 et 256.

Le calcul de la distribution des poids des codes obtenus pour un représentant de chaque classe d'équivalence irréductible, pour les valeurs de m comprises entre 4 et 9, m'a permis de constater qu'il n'y a pas d'autres équivalences entre deux codes de Goppa irréductibles 3-correcteurs pour ces longueurs que celles définies à la proposition 2.11. Cette dernière remarque ne laisse pas présager beaucoup de chances de réduire de façon significative le nombre de classes d'équivalence des codes de Goppa irréductibles.

représentant	cardinal de la classe	nombre de classes	forme des polynômes de la classe
x^3	n	1	$(x+a)^3$ où $a \in \mathbb{F}_{2^m}$
x^3+1	$n(n-1)$ si m impair	1	$x^3+ax^2+a^2x+b$ où $a,b \in \mathbb{F}_{2^m}, b \neq 0$
	$\frac{n(n-1)}{3}$ si m pair	1	$x^3+ax^2+a^2x+a^3+b^3$ où $a,b \in \mathbb{F}_{2^m}, b \neq 0$
x^3+x	$n(n-1)$	1	$x^3+ax^2+bx+ab$ où $a,b \in \mathbb{F}_{2^m}, b \neq a^2$
$x^3+\alpha$ si m pair	$\frac{2n(n-1)}{3}$	1	$x^3+ax^2+a^2x+a^3+\alpha b^3$ $x^3+ax^2+a^2x+a^3+\alpha^2 b^3$ où $a,b \in \mathbb{F}_{2^m}, b \neq 0$
$x^3+x+\alpha^i$ i représentant d'une classe cyclotomique mod 2^m-1	$cn(n-1)$	$\frac{1}{m}\sum_{d/m}\phi(\frac{m}{d})2^d$	x^3+ax^2+bx+c où $a,b,c \in \mathbb{F}_{2^m}, b \neq a^2$

TAB. 2.5 –: *Classification des codes de Goppa $\Gamma(\mathbf{F}_{2^m}, g)$, où g est un polynôme de degré 3*

représentant	cardinal de la classe
x^3+x+1	$n(n-1)$
$x^3+x+\alpha^3$	$7n(n-1)$
$x^3+x+\alpha^5$	$7n(n-1)$
$x^3+x+\alpha^{19}$	$7n(n-1)$
$x^3+x+\alpha^{21}$	$7n(n-1)$
$x^3+x+\alpha^{23}$	$7n(n-1)$
$x^3+x+\alpha^{43}$	$7n(n-1)$

où α est un élément primitif de $\mathbb{F}_{2^7}$.

TAB. 2.6 –: *Classes d'équivalence des codes de Goppa irréductibles 3-correcteurs en longueur* $n = 128$

représentant	cardinal de la classe
$x^3+\alpha$	$2n(n-1)/3$
x^3+x+1	$n(n-1)$
$x^3+x+\alpha^{17}$	$4n(n-1)$
$x^3+x+\alpha^3$	$8n(n-1)$
$x^3+x+\alpha^{11}$	$8n(n-1)$
$x^3+x+\alpha^{21}$	$8n(n-1)$
$x^3+x+\alpha^{43}$	$8n(n-1)$
$x^3+x+\alpha^{45}$	$8n(n-1)$
$x^3+x+\alpha^{53}$	$8n(n-1)$
$x^3+x+\alpha^{55}$	$8n(n-1)$
$x^3+x+\alpha^{59}$	$8n(n-1)$
$x^3+x+\alpha^{61}$	$8n(n-1)$
$x^3+x+\alpha^{91}$	$8n(n-1)$

où α est un élément primitif de $\mathbb{F}_{2^8}$.

TAB. 2.7 –: *Classes d'équivalence des codes de Goppa irréductibles 3-correcteurs en longueur* $n = 256$

La conclusion de cette étude est donc qu'actuellement, l'utilisation des codes de Goppa irréductibles comme codes secrets dans le système de McEliece n'induit aucune attaque réaliste permettant soit de trouver la clef secrète à partir de la clef publique, soit de décoder rapidement le code public.

2 Schémas d'identification à mots de poids faible

Tout comme les systèmes de chiffrement à clef publique, la plupart des schémas d'identification zero-knowledge utilisés actuellement, dont le plus connu est celui de Fiat-Shamir [FS86], repose sur des problèmes de théorie des nombres. L'idée d'utiliser des codes correcteurs pour construire de nouveaux schémas d'identification fut introduite par Sami Harari [Har88] à travers un protocole cryptanalysé de plusieurs manières par Pascal Véron dans [Vér95b, chapitre 5] et [Vér95a]. Elle fut néanmoins reprise successivement par Jacques Stern [Ste89a] et Marc Girault [Gir90], qui présentèrent des protocoles très simples mais difficilement utilisables du fait du nombre astronomique de bits à échanger lors d'une procédure d'identification. Je montre ici qu'en plus, le schéma de Girault peut être aisément cryptanalysé.

Ces travaux ont toutefois abouti à un protocole praticable proposé par Jacques Stern à CRYPTO'93 et qui résiste toujours à la cryptanalyse. Les performances de ce schéma furent par la suite améliorées par Pascal Véron qui en proposa plusieurs variantes [Vér95b].

2.0.1 Schéma d'identification de Girault

Même s'il est impraticable, de l'avis-même de son auteur, le schéma proposé par Marc Girault [Gir90] a l'avantage de simplifier considérablement ceux présentés par S. Harari et J. Stern quelques années auparavant. Son fonctionnement est décrit à la table 2.8.

Ce protocole est zero-knowledge et le fait de l'itérer τ fois assure une sécurité de $1 - \frac{1}{2^\tau}$.

Comme dans le schéma proposé par Stern à EUROCRYPT'89, on peut identifier la matrice publique H à la transposée d'une matrice de parité d'un code binaire aléatoire $\mathcal{C}$ de longueur n et de dimension $k = n - r$. La clef secrète d'un utilisateur est un mot de poids w de $\mathbb{F}_2^n$ et sa clef publique son syndrome relativement au code $\mathcal{C}$.

Au vu de la proposition 2.6, la matrice H peut être exprimée sous forme systématique. Chaque clef publique comporte donc $r(n-r)+n$ bits. Chacune des τ étapes du protocole d'identification requiert $r(2r-1)(n-r+1)$ opérations binaires et l'échange de $r(n+1) + \frac{1}{2}(n + r^2 + \ell_\sigma)$ bits, où ℓ_σ est le nombre de bits nécessaires pour transmettre une permutation de n positions — on suppose qu'une telle permutation est produite par un générateur de permutations pseudo-aléatoire initialisé par un germe de ℓ_σ bits [Vér95b, page 93].

Les paramètres proposés par Marc Girault

$$n = 512,\ k = 256,\ w = 55,\ \tau = 20$$

Informations publiques partagées par tous les utilisateurs :
H : matrice binaire $n \times r$ aléatoire de rang r.
w : entier positif

Clef secrète d'un utilisateur :
s : vecteur binaire de longueur n et de poids w.

Clef publique d'un utilisateur :
$i = sH$: vecteur binaire de longueur r.

Protocole permettant à Alice de s'identifier auprès de Bob (en τ passes) :

1. Alice choisit une permutation σ de $\{1, \ldots, n\}$ et une matrice aléatoire $r \times r$ inversible, S. Elle calcule $H' = \sigma HS$ et $i' = Si$ qu'elle envoie à Bob.

2. Bob choisit aléatoirement un bit $b \in \{0, 1\}$ qu'il transmet à Alice.

3. - Si $b = 0$, Alice révèle S et σ à Bob, qui vérifie que $\sigma HS = H'$ et $Si = i'$.
   - Si $b = 1$, Alice révèle $s' = \sigma s$ à Bob, qui vérifie que s' est de poids w et que $s'H' = i'$.

TAB. 2.8 –: *Schéma d'identification de Girault*

rendent prohibitive la quantité de bits transmis lors de la transaction — dans ce cas $\ell_\sigma = 120$.

De plus, ce schéma peut être cassé par l'algorithme de Sendrier [Sen96] qui permet de retrouver la permutation entre deux codes équivalents. En effet, les codes utilisés ici sont des codes aléatoires ; leur groupe d'automorphismes est par conséquent trivial avec une grande probabilité. Aussi, de la connaissance de H' et de H , l'algorithme de Sendrier déduit-il la permutation σ et donc la clef secrète s de l'utilisateur dès que le bit b vaut 1.

2.1 Schéma d'identification de Stern

Le schéma d'identification présenté par J. Stern à CRYPTO'93 [Ste93] est, contrairement au précédent, praticable. De plus, il résiste à la cryptanalyse que je viens de décrire. Ce nouveau schéma est décrit à la table 2.9.

L'avantage de ce protocole est qu'il ne nécessite plus la transmission de matrices. Cela réduit en conséquence considérablement le nombre de bits transmis au cours de la transaction.

Comme dans le protocole précédent, la clef publique peut être stockée en $r(n-r)+n$ bits. Chacune des τ passes de l'identification requiert $r(2(n-r)+$

Informations publiques partagées par tous les utilisateurs :
H : matrice binaire $n \times r$ aléatoire de rang r.
h : fonction de hachage.
w : entier positif.

Clef secrète d'un utilisateur :
s : vecteur binaire de longueur n et de poids w.

Clef publique d'un utilisateur :
$i = sH$: vecteur binaire de longueur r.

Protocole permettant à Alice de s'identifier auprès de Bob (en τ passes) :

1. Alice choisit un mot binaire y de longueur n aléatoire et une permutation σ de $\{1, \ldots, n\}$ Elle calcule et envoie à Bob les trois valeurs :
$$c_1 = h(\sigma | yH),\ c_2 = h(\sigma y),\ c_3 = h(\sigma(y + s))$$
2. Bob choisit un aléa $b \in \{0, 1, 2\}$ qu'il transmet à Alice.
3. - Si $b = 0$, Alice révèle y et σ à Bob, qui vérifie que $c_1 = h(\sigma | yH)$ et $c_2 = h(\sigma y)$.
- Si $b = 1$, Alice révèle $y + s$ et σ à Bob, qui vérifie que $c_1 = h(\sigma | (y + s)H + i)$ et que $c_3 = h(\sigma(y + s))$.
- Si $b = 2$, Alice révèle $\sigma(y)$ et $\sigma(s)$ à Bob, qui vérifie que $c_2 = h(\sigma y)$, que $c_3 = h(\sigma(y) + \sigma(s))$ et que le poids de $\sigma(s)$ est bien w.

TAB. 2.9 –: *Schéma d'identification de Stern*

$1) + n + \frac{n}{3}$ opérations binaires et la transmission de $3h + \frac{2}{3}(2n + \ell_\sigma)$ bits, où h est la longueur du haché ([Vér95b, page 94]).

Les paramètres suggérés par l'auteur sont

$$n = 512,\ k = 256,\ w = 56,\ \tau = 35$$

ainsi que l'utilisation d'une fonction h produisant des hachés de 128 bits.

2.2 Schéma d'identification de Véron

L'idée de Pascal Véron est de reprendre la présentation originale de McEliece et d'exprimer le protocole d'identification sous sa forme duale, *i.e.* en utilisant la matrice génératrice d'un code et non sa matrice de parité. Ce nouveau schéma est décrit à la table 2.10.

Informations publiques partagées par tous les utilisateurs :
G : matrice binaire $k \times n$ aléatoire de rang k.
h : fonction de hachage.
w : entier positif.

Clef secrète d'un utilisateur :
m : vecteur binaire de longueur k.
e : vecteur binaire de longueur n et de poids w.

Clef publique d'un utilisateur :
$i = mG + e$: vecteur binaire de longueur n.

Protocole permettant à Alice de s'identifier auprès de Bob (en τ passes) :

1. Alice choisit un mot binaire u de longueur k aléatoire et une permutation σ de $\{1, \ldots, n\}$ Elle calcule et envoie à Bob les trois valeurs :

$$c_1 = h(\sigma),\ c_2 = h(\sigma((u+m)G)),\ c_3 = h(\sigma(uG+i))$$

2. Bob choisit un aléa $b \in \{0, 1, 2\}$ qu'il transmet à Alice.

3. - Si $b = 0$, Alice révèle $u + m$ et σ à Bob, qui vérifie que $c_1 = h(\sigma)$ et $c_2 = h(\sigma((u+m)G))$.
   - Si $b = 1$, Alice révèle $\sigma((u+m)G)$ et σe à Bob, qui vérifie que $c_2 = h(\sigma((u+m)G))$, que $c_3 = h(\sigma((u+m)G) + \sigma e)$, et que le poids de σe est bien w.
   - Si $b = 2$, Alice révèle σ et u à Bob, qui vérifie que $c_1 = h(\sigma)$ et $c_3 = h(\sigma(uG+i))$.

TAB. 2.10 –: *Schéma d'identification de Véron*

La clef publique de l'utilisateur peut être stockée sur $nk + n$ bits. Chacune des τ passes du protocole nécessite en moyenne $\frac{8}{3}k(n-k) + n + \frac{5}{3}k$ opérations binaires et la transmission de $3h + \frac{2}{3}(k + \ell_\sigma + n)$ bits ([Vér95b, page 80]).

Pascal Véron propose pour ce protocole un nouveau jeu de paramètres qui minimise le nombre de bits transmis

$$n = 512,\ k = 120,\ w = 114,\ \tau = 35$$

Je renvoie aux travaux de Pascal Véron [Vér95b] pour une étude plus complète de tous ces schémas. Je compare ici leurs performances à la table 2.11.

	Girault	Stern	Véron
code utilisé	[512,256]	[512,256]	[512,120]
w	55	56	114
taille des clefs publiques	8256 octets	8256 octets	7744 octets
nb d'opérations binaires	$2^{29,3}$	$2^{22,1}$	$2^{22,1}$
nb de bits échangés	3 288 240	40 133	30 987

TAB. 2.11 –: *Comparaison des performances des différents schémas d'identification à mots de poids faible*

2.3 Attaque des schémas d'identification à mots de poids faible

Une attaque générale de tous les schémas d'identification que je viens de présenter consiste à rechercher la clef secrète d'un utilisateur à partir de sa clef publique et du code public $\mathcal{C}$. Elle revient donc à rechercher un mot de poids w dans un coset particulier du code $\mathcal{C}$, ou de façon équivalente (cf. proposition 2.4) à décoder $\mathcal{C}$ jusqu'à la distance w.

Cette fois-ci, le code $\mathcal{C}$ est un code aléatoire. Sa distance minimale d est donc estimée par la borne de Gilbert-Varshamov

$$\frac{k}{n} = 1 - H_2(\frac{d_{GV}}{n})$$

où H_2 est la fonction entropie binaire définie par

$$H_2(x) = -x\log_2(x) - (1-x)\log_2(1-x)$$

L'idée est donc de choisir pour w une valeur légèrement inférieure à celle de la borne de Gilbert-Varshamov. Dans ce cas, la probabilité que le coset considéré contienne un autre mot de poids w — qui permettrait de se faire passer pour Alice — est très faible.

Proposition 2.15 *Soit $\mathcal{C}$ un code linéaire binaire aléatoire de longueur n, de dimension k, de distance minimale d_{GV} et e un mot de longueur n et de poids w. Alors, le nombre de mots de poids inférieur ou égal à w, autres que e, dans le coset $\mathcal{C}+e$ est en moyenne*

$$\frac{1}{2^{n-k}} \sum_{i=d_{GV}}^{2w} \sum_{j=i/2}^{w} \binom{w}{j}\binom{n-w}{i-j}$$

preuve : Soit $x+e$, où $x \in \mathcal{C}$, un mot du coset $\mathcal{C}+e$. Son poids est donné par $w(x+e) = w(x) + w - 2w(x \wedge e)$. Donc $w(x+e) \leq w$ si et seulement si $w(x \wedge e) \geq \frac{w(x)}{2}$. En remarquant que $w(x \wedge e) \leq w$, on obtient donc

$$\begin{aligned} Pr[w(x+e) \leq w] &= \sum_{i=d_{GV}}^{2w} Pr[w(x \wedge e) \geq \frac{i}{2} \,/ w(x) = i] \frac{\binom{n}{i}}{2^n} \\ &= \sum_{i=d_{GV}}^{2w} \frac{\binom{n}{i}}{2^n} \sum_{j=i/2} w \frac{\binom{w}{j}\binom{n-w}{i-j}}{\binom{n}{i}} \\ &= \frac{1}{2^n} \sum_{i=d_{GV}}^{2w} \sum_{j=i/2}^{w} \binom{w}{j}\binom{n-w}{i-j} \end{aligned}$$

□

Un code binaire aléatoire de longueur 512 et de dimension 256 est en moyenne de distance minimale 57. Dans ce cas, le coset $\mathcal{C}+e$ possède donc en moyenne 0,057 mots de poids inférieur ou égal à 56 autres que e. Pour les paramètres choisis par Pascal Véron, le coset $\mathcal{C}+e$ d'un code binaire aléatoire [512,120,115] avec $w(e) = 114$ possède en moyenne 0,073 autres mots de poids inférieur ou égal à w.

Aussi peut-on dans ce cas faire l'hypothèse que le coset $\mathcal{C}+e$ est de poids minimal w.

Conclusion

Les attaques structurelles des systèmes de McEliece et de Niederreiter, visant à retrouver la clef secrète à partir de la clef publique, ont à ce jour toutes échoué dès que la famille de codes secrets est bien choisie. Je vais donc maintenant m'intéresser aux possibilités de trouver un algorithme de déchiffrement pour ces systèmes, *i.e.* un algorithme permettant de déduire de tout texte chiffré le texte clair correspondant. Dans le cas de l'algorithme de McEliece, il s'agit donc d'un algorithme capable de décoder jusqu'à sa capacité de correction un code de Goppa permuté, dont nous avons vu qu'il se comportait comme un code aléatoire. Attaquer le système de Niederreiter et le schéma d'identification de Stern consiste à retrouver un mot de poids w ayant un syndrome donné. La proposition 2.4 montre qu'une telle attaque est équivalente au décodage du code public jusqu'à la distance w.

Aussi la suite de cette étude est-elle consacrée aux algorithmes de décodage jusqu'à la distance w. Dans le cas des systèmes de chiffrement, w est égal à la capacité de correction du code ; dans celui des schémas d'identification, w est légèrement inférieur à sa distance minimale.

Chapitre 3

Le problème général du décodage d'un code linéaire

L'étude menée au chapitre précédent montre donc qu'en l'absence de cryptanalyse exploitant l'utilisation des codes de Goppa irréductibles, l'attaque des systèmes de chiffrement de McEliece et de Niederreiter, ainsi que des schémas d'identification proposés par Stern et Véron, revient à trouver un algorithme général de décodage efficace des codes linéaires jusqu'à une distance fixée.

Contrairement au problème du décodage complet d'un code linéaire, il n'a jamais été démontré que ce problème était NP-dur. Mais l'existence d'un algorithme polynômial pour le résoudre paraît assez peu probable. Cela ne signifie pas pour autant qu'il n'existe pas d'algorithmes de décodage — même de décodage complet — ayant une meilleure complexité que la recherche exhaustive. Toute amélioration des algorithmes existant est donc digne d'intérêt même si elle aboutit à un algorithme exponentiel ; elle conditionne en effet la sécurité des cryptosystèmes à mots de poids faible.

Après avoir rappelé les résultats de Berlekamp, McEliece et van Tilborg sur la complexité du décodage, je décrirai successivement les différents algorithmes de décodage existant : les algorithmes exhaustifs, le décodage par ensembles d'information et ces diverses variantes proposées par Lee et Brickell [LB88], Leon [Leo88] et Stern [Ste89b]. Je m'attacherai tout particulièrement à comparer leurs complexités respectives et évaluer le nombre d'opérations qu'ils requièrent pour attaquer les principaux cryptosystèmes à mots de poids faible. Enfin, je montrerai que tous ces algorithmes peuvent être optimisés de diverses manières.

1 La NP-complétude du décodage complet

Dans leur article fondateur de 1978 [BMvT78], Berlekamp, McEliece et van Tilborg ont montré que le problème général de la détermination des poids des cosets d'un code linéaire est un problème NP-complet.

Définition 3.1 (Problème de la détermination des poids des cosets)
Entrée :
- H : matrice binaire $n \times r$.
- s : vecteur binaire de longueur r.
- w : entier positif
Problème : *Existe-t-il un vecteur binaire e de longueur n et de poids inférieur ou égal à w tel que*

$$eH = s$$

Le problème NP-complet ainsi défini est toutefois plus général que le problème de la détermination des poids des cosets. Ce dernier implique en effet que la matrice H, identifiée à la transposée d'une matrice de parité, est de rang r. Cette restriction ne modifie cependant pas la complexité du problème [BMvT78, section IV]. Du reste, la connaissance au préalable de la matrice H dans les problèmes de théorie des codes ne réduit pas non plus sa difficulté puisque Bruck et Naor ont montré qu'il reste NP-complet si l'on autorise une infinité de précalculs sur la matrice H [BN90].

Par une méthode similaire à celle que j'ai employée dans la preuve de la proposition 2.4, il est aisé de montrer que tout algorithme de décodage complet d'un code linéaire permet de résoudre le problème de la détermination des poids des cosets.

Définition 3.2 (Problème du décodage complet d'un code linéaire)
Entrée :
- G : matrice binaire $k \times n$ de rang k.
- x : vecteur binaire de longueur n.
Problème : *Trouver un vecteur binaire m de longueur k tel que $d(x, mG)$ est minimale.*

Aussi le problème du décodage complet d'un code linéaire est-il un problème NP-dur.

Toutefois, le problème invoqué pour l'attaque des cryptosystèmes à mots de poids faible est celui du décodage jusqu'à une distance fixée ; il est donc en général plus restrictif que celui du décodage complet. En effet, ce dernier permet de décoder tous les mots de l'espace c'est-à-dire jusqu'à la distance ρ, où ρ est le rayon de recouvrement du code — puisque tout vecteur de longueur n est, par définition, à distance au plus ρ d'un mot de code —, alors que je n'ai besoin que de décoder les mots à distance w d'un mot de code pour des valeurs de w de l'ordre, soit de la capacité de correction du code, soit de sa distance minimale.

Le décodage borné n'est donc pas nécessairement un problème NP-dur ; toutefois les résultats actuels laissent envisager peu de chances de le résoudre en un temps polynômial.

2 Notations

Je vais maintenant étudier différents algorithmes de décodage borné. Dans toute la suite, il s'agira donc de décoder un code linéaire binaire $\mathcal{C}$ de longueur n et de dimension $k = Rn$ jusqu'à la distance w. Par le vecteur x je désignerai le mot à décoder, formé de la somme d'un mot de code et d'un vecteur-erreur e de poids w ; G et H seront respectivement une matrice génératrice et une matrice de parité du code $\mathcal{C}$.

Je rappelle que dans les cas qui m'intéressent — w égal à la capacité de correction du code ou proche de sa distance minimale — le coset $\mathcal{C}+x$ ne contient en moyenne qu'un seul mot de poids inférieur ou égal à w (cf. proposition 2.15). Le problème du décodage borné a donc une solution unique. Pour cette raison, tout algorithme de recherche de mot de poids minimal dans un code linéaire permet également de décoder le vecteur x. En effet, le code $\mathcal{C} \cup (\mathcal{C} + x)$ est un code linéaire de longueur n, de dimension $k+1$ et de distance minimale w dont l'unique mot de poids minimal est e.

Pour étudier la complexité des algorithmes de décodage, j'utiliserai les mesures suivantes :

Définition 3.3 (Complexité d'un algorithme)

- *Le* facteur de travail *d'un algorithme de décodage d'un code $[n, nR]$ jusqu'à la distance w, noté $W(n, R, w)$, est le nombre d'opérations binaires qu'il requiert.*

- *Le* coefficient de complexité *correspondant, noté $\alpha(n, R, w)$, est défini par*

$$\alpha(n, R, w) = \lim_{n \to +\infty} \frac{1}{n} \log_2 W$$

Lorsque je parlerai de facteur de travail sans autre précision, il s'agira toujours du nombre d'opérations effectuées *en moyenne*. La complexité en moyenne d'un algorithme de décodage est en effet celle qui est la plus pertinente pour mesurer la résistance des cryptosystèmes à mots de poids faible à cette attaque. D'un point de vue de théorie des codes, la complexité dans le pire des cas est évidemment importante ; son intérêt cryptographique est par contre relativement limité pour le problème qui m'intéresse.

De la même façon, les analyses asymptotiques de complexité n'ont ici qu'une valeur théorique. Dans la pratique, c'est le facteur de travail qui permet de mesurer si une attaque est réaliste ; même si le décompte des opérations binaires effectuées par un algorithme paraît souvent contestable dans la mesure où il dépend souvent de l'implémentation, lui seul permet d'estimer la faisabilité d'une cryptanalyse.

3 Algorithmes exhaustifs

Les premiers algorithmes de décodage que j'évoque sont les algorithmes dits *exhaustifs* puisqu'ils consistent à énumérer soit tous les mots du code, soit tous les vecteurs d'erreurs.

3.1 Recherche exhaustive parmi les mots du code (REC)

Un algorithme immédiat consiste à énumérer les 2^k mots du code $\mathcal{C}$ et à les comparer à x afin de trouver le plus proche de x au sens de la distance de Hamming (cf. table 3.1).

Pour tous les vecteurs m de longueur k,
1. calculer le mot de code mG.
2. calculer le poids de Hamming de $x + mG$.
La solution est obtenue pour le vecteur m qui minimise $w(x+mG)$.

TAB. 3.1 –: *Décodage par recherche exhaustive parmi tous les mots du code*

Il s'agit donc d'un algorithme de décodage complet qui nécessite 2^k itérations. Chacune d'elles effectue la multiplication mG en $\frac{kn}{2}$ opérations binaires en moyenne, l'addition $x + mG$ en $\frac{n}{2}$ opérations et le calcul du poids en n opérations.

Même si on l'utilise comme algorithme de décodage jusqu'à la distance w, son facteur de travail reste identique.

Proposition 3.4 (Complexité du décodage par recherche exhaustive parmi tous les mots du code) *Le facteur de travail de l'algorithme de décodage jusqu'à la distance w par recherche exhaustive parmi tous les mots du code est*

$$W_{REC}(n, R, w) = 2^{nR} n(\frac{Rn}{2} + \frac{3}{2})$$

Son coefficient de complexité est donc

$$\alpha_{REC}(n, R, w) = R$$

3.2 Recherche exhaustive parmi les vecteurs d'erreurs (REE)

Cet algorithme, parfois appelé algorithme de décodage exhaustif par syndrome, consiste à examiner tous les vecteurs d'erreurs possibles et à comparer leur syndrome avec celui de x (cf. table 3.2).

A chaque itération, il effectue donc de l'ordre de $w(n-k)$ opérations pour le calcul du syndrome et $(n-k)$ opérations pour la comparaison. S'il s'agit d'un algorithme de décodage complet, son nombre d'itérations est égal au nombre

Soit $s = x\,{}^tH$.
Tant que $s + s' \neq 0$

1. choisir un vecteur y de poids inférieur ou égal à w.
2. calculer son syndrome $s' = y\,{}^tH$.

TAB. 3.2 –: *Décodage par recherche exhaustive parmi tous les vecteurs d'erreurs*

de cosets du code, c'est-à-dire 2^{n-k}. Dans ce cas, son coefficient de complexité est $1-R$. Par contre, si l'on ne considère que le décodage jusqu'à la distance w, le nombre d'itérations est égal au nombre de vecteurs d'erreurs de poids au plus w, c'est-à-dire $\sum_{i=1}^{w} \binom{n}{i}$.

Proposition 3.5 (Complexité du décodage par recherche exhaustive parmi tous les vecteurs d'erreurs) *Le facteur de travail de l'algorithme de décodage jusqu'à la distance w par recherche exhaustive parmi tous les vecteurs d'erreurs est*

$$W_{REE}(n, R, w) = n(1-R)(w+1)\sum_{i=1}^{w}\binom{n}{i}$$

Son coefficient de complexité est donc

$$\alpha_{REE}(n, R, w) = H_2(\frac{w}{n})$$

preuve : Le coefficient de complexité est obtenu par l'approximation

$$\lim_{n\to+\infty}\frac{1}{n}\sum_{i=0}^{w}\binom{n}{i} = H_2(\frac{w}{n})$$

□

Notons que si le code $\mathcal{C}$ est un code aléatoire, sa distance minimale est estimée par la borne de Gilbert-Varshamov. Le coefficient de complexité de l'algorithme s'il est utilisé pour décoder jusqu'à la capacité de correction du code vaut donc $H_2\left(\frac{H_2^{-1}(1-R)}{2}\right)$, et $H_2(H_2^{-1}(1-R))$ s'il est utilisé pour décoder jusqu'à la distance minimale.

4 Décodage par ensembles d'information (IS)

Le décodage par ensembles d'information est une technique classique qui fut introduite par Prange [Pra62] dans les années soixante pour les codes cycliques, puis généralisée par différents auteurs. Contrairement aux algorithmes exhaustifs, le décodage par ensembles d'information exploite la redondance du code : il repose sur la constatation qu'il suffit pour décoder un vecteur x de trouver

parmi ses coordonnées un ensemble de k positions d'information ne contenant aucune erreur.

Définition 3.6 (Ensemble d'information) *Soit $\mathcal{C}$ un code linéaire de longueur n et de dimension k, et G une matrice génératrice de $\mathcal{C}$. Un sous-ensemble I de $\{1, \ldots, n\}$ de taille k est un* ensemble d'information *pour le code $\mathcal{C}$ si et seulement si la restriction du code à ces k positions est un espace vectoriel de dimension k. Ceci équivaut à dire que la matrice carré G_I correspondant à la restriction de G aux positions de I est inversible.*

De manière similaire, un ensemble de redondance *pour le code est le complémentaire d'un ensemble d'information.*

Pour toute matrice $n \times k$, j'adopterai la notation $G = (U, V)_I$ pour préciser que U (resp. V) désigne la restriction de G à l'ensemble I (resp. à son complémentaire).

Dès lors que l'on trouve un ensemble d'information I ne contenant aucune position erronée, il suffit de décomposer $x = (x_I, x_J)_I$ et $G = (U, V)_I$; le vecteur-erreur est alors égal à $x + x_I U^{-1} G = (0, x_J + x_I U^{-1} V)_I$.

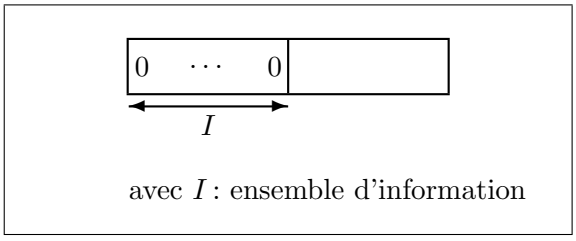


FIG. 3.1 –: *Motifs d'erreurs corrigés à chaque itération par l'algorithme de décodage par ensembles d'information : on corrige tous les vecteurs ne possédant aucune position erronée dans l'ensemble d'information I.*

Cette propriété justifie donc l'algorithme de décodage décrit à la table 3.3. Son utilisation pour attaquer les cryptosystèmes à mots de poids faible était

Tant qu'un vecteur d'erreurs de de poids inférieur ou égal à w n'a pas été trouvé

1. Choisir aléatoirement un ensemble d'information I.
2. Mettre la matrice G sous forme systématique $U^{-1}G = (Id, Z)_I$ et décomposer le vecteur x sous la forme $x = (x_I, x_J)_I$.
3. Calculer $w(x_J + x_I Z)$.
   Si ce poids est inférieur ou égal à w, alors $e = (0, x_J + x_I Z)_I$.

TAB. 3.3 –: *Décodage par ensembles d'information*

déjà mentionnée dans l'article de McEliece [McE78].

Un des problèmes essentiels qui apparaît dans cette présentation réside dans le fait que k positions choisies aléatoirement ne forment pas toujours un ensemble d'information, sauf si le code est un code MDS. La probabilité que k colonnes de la matrice génératrice G soient linéairement indépendantes varie évidemment suivant le code étudié. Dans le cas où il s'agit d'un code aléatoire, on peut malgré tout l'estimer par la proportion de matrices inversibles parmi les matrices binaires $k \times k$. En simplifiant la formule donnée dans [LN83, page 455], on obtient que le nombre de matrices binaires $k \times (k+m)$ de rang k est égal à

$$2^{\frac{k(k-1)}{2}} \prod_{i=1}^{k} (2^{i+m} - 1)$$

Aussi la probabilité qu'une matrice binaire $k \times (k+m)$ soit inversible vaut-elle

$$\begin{aligned} q_{k,m} &= 2^{\frac{-k(k+1)}{2}} \prod_{i=1}^{k} 2^i \prod_{i=m+1}^{m+k} \left(1 - \frac{1}{2^i}\right) \\ &= \prod_{i=m+1}^{m+k} \left(1 - \frac{1}{2^i}\right) \end{aligned}$$

Cela implique que, dès que la valeur de k est suffisamment élevée, la probabilité pour que k coordonnées d'un code binaire aléatoire forment un ensemble d'information est de l'ordre de 0,289. Cette probabilité croît malgré tout très rapidement avec m, ce qui signifie que dans la pratique tout ensemble de $k+5$ positions contient un ensemble d'information (en effet $q_{k,5} = 0,97$).

Dans son calcul de la complexité du décodage par ensembles d'information [vT94, pages 47-48], van Tilburg considère donc qu'il est nécessaire de choisir en moyenne 3,46 ensembles de k positions pour obtenir un ensemble d'information, et par conséquent d'effectuer 3,46 éliminations de Gauss sur la matrice G lors de chaque itération. Cette approche n'est évidemment pas réaliste puisque, en général, après avoir examiné k colonnes de la matrice G on a obtenu $k-1$ ou $k-2$ positions d'information. Dans la pratique, on choisit alors aléatoirement une ou deux autres colonnes pour achever la procédure de pivot au lieu de la recommencer entièrement. Le fait que moins du tiers des choix de k colonnes fournisse un ensemble d'information n'augmente donc pas le coût de l'élimination de Gauss ; elle nécessite tout au plus une ou deux permutations de colonnes sur la matrice G, ce qui est complètement négligeable.

Proposition 3.7 (Facteur de travail du décodage par ensembles d'information) *Le facteur de travail de l'algorithme de décodage par ensembles d'information jusqu'à la distance w est*

$$W_{IS}(n,R,w) = \frac{\binom{n}{nR}}{\binom{n-w}{nR}} \left(\frac{n^3R^2}{2} + \frac{n^2R(1-R)}{2} + n(1-R) \right)$$

Le coefficient de complexité correspondant est

$$\alpha_{IS}(n,R,w) = H_2\left(\frac{w}{n}\right) - (1-R)H_2\left(\frac{w}{(1-R)n}\right)$$

preuve : Le nombre d'itérations effectuées en moyenne par cet algorithme se déduit de la probabilité qu'un ensemble I de k positions ne contienne aucune des w positions erronées

$$Pr[I \cap supp(e) = \emptyset] = \frac{\binom{n-w}{k}}{\binom{n}{k}}$$

Le nombre d'opérations binaires par itération se décompose comme suit :

- $\frac{nk^2}{2}$ opérations pour l'élimination de Gauss sur la matrice G.
- $\frac{k(n-k)}{2}$ opérations binaires pour le calcul de $x_J + x_I Z$.
- $n-k$ opérations pour le calcul du poids.

L'expression du coefficient de complexité est obtenu à partir de

$$\frac{\binom{n}{k}}{\binom{n-w}{k}} = \frac{\binom{n}{w}}{\binom{n-k}{w}}$$

et de l'estimation

$$\lim_{n\to+\infty} \frac{1}{n} \log_2 \binom{n}{t} = H_2\left(\frac{n}{t}\right)$$

□

Il serait néanmoins intéressant d'estimer le nombre maximal d'itérations que doit effectuer l'algorithme pour corriger w erreurs. Il s'agit malheureusement d'un problème combinatoire très ardu, connu sous le nom de *problème du recouvrement*-(n,ℓ,t), qui consiste à estimer le nombre minimal $b(n,\ell,t)$ de sous-ensembles de taille ℓ d'un ensemble de n objets tel que tout ensemble de t objets est contenu dans au moins un des sous-ensembles de taille ℓ. Calculer le nombre minimal d'ensembles d'information qu'il faut examiner afin que, pour tout erreur de poids w, au moins l'un d'eux ne contienne aucune position erronée — ou, de façon équivalente, afin que toute erreur de poids w soit contenue dans un des ensembles de redondance — revient donc à résoudre le problème du recouvrement pour les paramètres $(n, n-k, w)$.

La valeur exacte de $b(n, n-k, w)$ n'est actuellement pas connue. Toutefois, Coffey et Goodman en ont donné un équivalent asymptotique :

Proposition 3.8 [CG90] *Soit R et w deux constantes telles que $0 < w < (1-R)n < n$. Alors*

$$\lim_{n\to+\infty} \frac{1}{n} \log_2 b(n, \lfloor n(1-R)\rfloor, w) = H_2\left(\frac{w}{n}\right) - (1-R)H_2\left(\frac{w}{n(1-R)}\right)$$

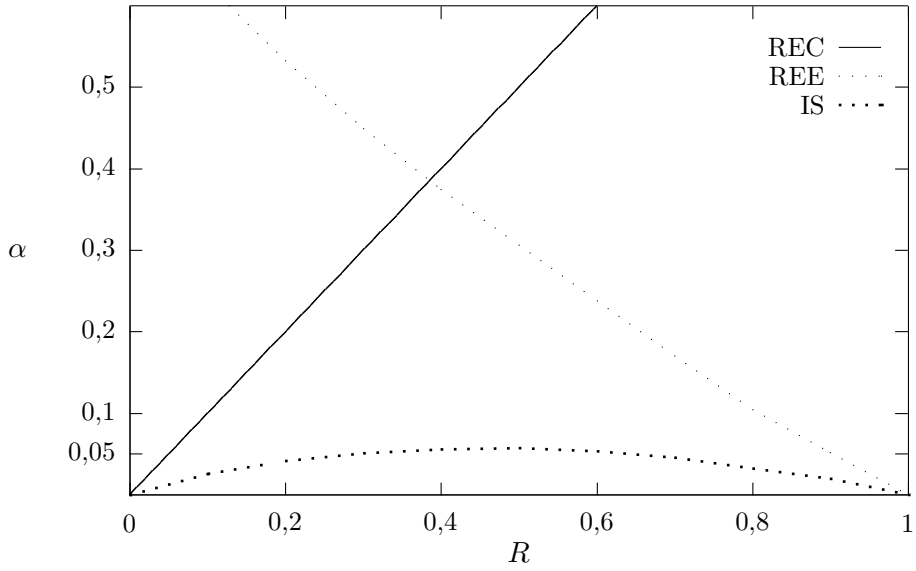


FIG. 3.2 –: *Coefficient de complexité des algorithmes classiques permettant de décoder un code binaire aléatoire* $[n, Rn]$ *jusqu'à sa capacité de correction*

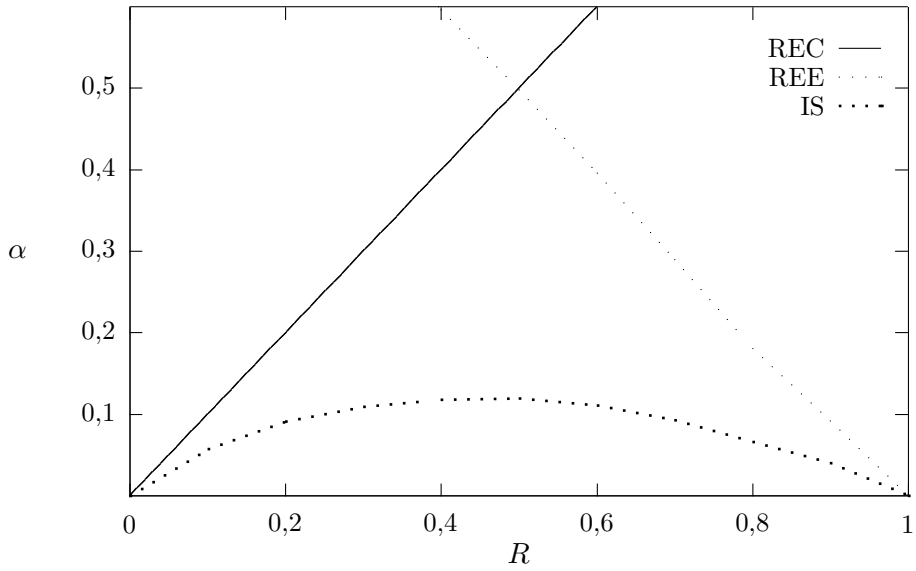


FIG. 3.3 –: *Coefficient de complexité des algorithmes classiques permettant de décoder un code binaire aléatoire* $[n, Rn]$ *jusqu'à sa distance minimale*

Ce résultat implique donc que, asymptotiquement, la complexité dans le pire des cas du décodage par ensembles d'information jusqu'à la distance w est la même que sa complexité en moyenne.

Les figures 3.2 et 3.3 permettent de comparer la complexité asymptotique moyenne du décodage par ensembles d'information avec celles des algorithmes exhaustifs. Elles représentent en effet le coefficient de complexité de ces algorithmes s'ils sont utilisés pour décoder un code binaire aléatoire jusqu'à la distance w, pour w égal à la capacité de correction du code, et à sa distance minimale.

5 Généralisations du décodage par ensembles d'information

De nombreuses variantes du décodage par ensembles d'information permettent d'en réduire le facteur de travail. Ces divers algorithmes peuvent paraître à première vue hétéroclites ; en fait, ils exploitent principalement deux idées. La première est d'introduire des effacements dans l'algorithme précédent, c'est-à-dire d'autoriser certains motifs d'erreurs dans l'ensemble d'information. C'est cette technique, introduite par Kasami pour les codes cycliques [Kas64], qui est reprise dans l'algorithme de Lee et Brickell. La deuxième idée consiste, dans un premier temps, à décoder jusqu'à une certaine distance un code poinçonné, obtenu en enlevant certaines positions de redondance du code $\mathcal{C}$, puis à vérifier si ce décodage partiel s'applique au code tout entier. Cette méthode est, elle, sous-jacente dans l'algorithme proposé par Leon.

Ce point de vue m'a permis, en collaboration avec Florent Chabaud, de concevoir un algorithme plus général que ceux proposés auparavant, qui unifie l'approche de Lee et Brickell et celle de Leon. Je montrerai également que l'algorithme conçu par Jacques Stern peut être interprété de manière identique.

5.1 Algorithme de Lee-Brickell (LB)

L'algorithme présenté par Lee et Brickell [LB88] pour cryptanalyser le système de McEliece est similaire au Q-décodage de Evseev [Evs83].

Il consiste simplement à rechercher un ensemble d'information contenant au plus p positions erronées — p est un paramètre de l'algorithme. Il peut donc être considéré comme un algorithme de décodage par ensembles d'information *avec p effacements*.

Cette approche est justifiée par le fait que, dans le décodage par ensembles d'information, l'élimination de Gauss permettant de mettre la matrice génératrice sous forme systématique est de loin la procédure la plus coûteuse. Afin d'accéder à un meilleur compromis entre le nombre d'itérations et le coût de chacune d'elles, il est donc préférable d'augmenter légèrement le facteur de travail de chaque itération en vérifiant en plus si l'ensemble d'information considéré ne contient pas qu'un tout petit nombre de positions erronées, car cela réduit considérablement le nombre total d'itérations.

A l'origine, Lee et Brickell proposèrent d'utiliser cet algorithme pour $p = 2$. En réalité, je montrerai par la suite que la valeur optimale de p dépend des paramètres du code considéré.

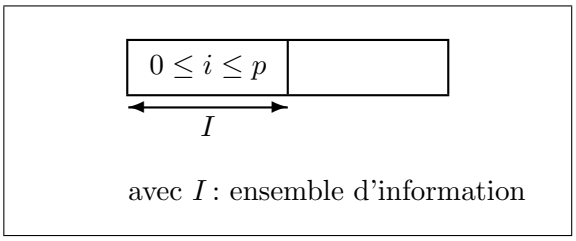


FIG. 3.4 –: *Motifs d'erreurs corrigés à chaque itération par l'algorithme de Lee-Brickell : on corrige tous les vecteurs d'erreurs dont la restriction à l'ensemble d'information I est de poids au plus p.*

Tant qu'un vecteur d'erreurs de poids inférieur ou égal à w n'a pas été trouvé

1. Choisir aléatoirement un ensemble d'information I.
2. Mettre la matrice G sous forme systématique $U^{-1}G = (Id, Z)_I$ et décomposer le vecteur x sous la forme $x = (x_I, x_J)_I$.
3. Pour tous les vecteurs ϵ_i de longueur k et de poids $i \leq p$, calculer $w(x_J + (x_I + \epsilon_i)Z)$.
   Si ce poids est inférieur ou égal à $w - i$, alors $e = (\epsilon_i, y_{\epsilon_i})_I$.

TAB. 3.4 –: *Algorithme de Lee-Brickell*

Proposition 3.9 (Facteur de travail de l'algorithme de Lee-Brickell) *Le facteur de travail de l'algorithme de Lee-Brickell utilisé pour décoder jusqu'à la distance w est*

$$W_{LB(p)}(n, R, w) = \frac{1}{\pi_{LB(p)}} \left(\frac{n^3R^2}{2} + \frac{n^2R(1-R)}{2} + n(1-R) \sum_{i=0}^{p} (i+1) \binom{nR}{i} \right)$$

$$\text{avec } \pi_{LB(p)} = \frac{1}{\binom{n}{nR}} \sum_{i=0}^{p} \binom{n-w}{nR-i} \binom{w}{i}$$

preuve : La probabilité pour qu'un ensemble d'information contienne au plus p positions erronées est

$$\frac{1}{\binom{n}{k}} \sum_{i=0}^{p} \binom{n-w}{k-i} \binom{w}{i}$$

Le nombre d'opérations par itération se décompose en :

– $\frac{nk^2}{2}$ opérations pour l'élimination de Gauss sur la matrice G.

– $\frac{k(n-k)}{2}$ opérations binaires pour calculer $x_J + x_I Z$.

– pour les $\binom{k}{i}$ valeurs de ϵ_i :

  – $(n-k)(i-1)$ opérations binaires pour calculer $\epsilon_i Z$.

  – $(n-k)$ opérations pour l'addition $x_J + (x_I + \epsilon_i)Z$.

  – $(n-k)$ opérations pour le calcul du poids.

□

5.2 Algorithme de Leon (LEON)

L'algorithme proposé par J.S. Leon [Leo88] était initialement conçu pour la recherche de mots de poids minimal dans un code linéaire mais il peut également être appréhendé comme un algorithme de décodage. La seule différence qu'il introduit par rapport au décodage par ensembles d'information avec effacements est qu'il considère à chaque itération un ensemble de positions S de taille supérieure à k qui, de plus, ne contient pas nécessairement un ensemble d'information. Mais, tout comme l'algorithme de Lee-Brickell, il décode tous les motifs d'erreurs dont la restriction à S est de poids inférieur ou égal à p. La

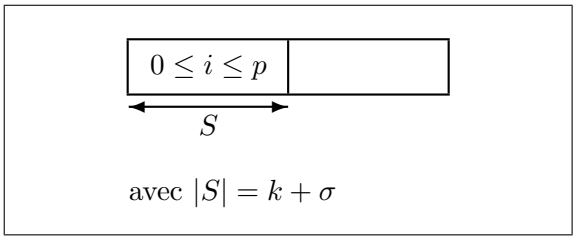


FIG. 3.5 –: *Motifs d'erreurs corrigés à chaque itération par l'algorithme de Leon (version originale) : on corrige tous les vecteurs d'erreurs dont la restriction à la sélection S est de poids au plus p.*

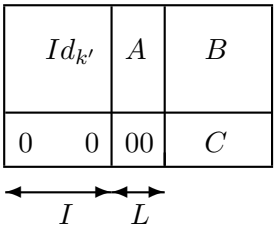
présentation originale de l'algorithme de Leon est détaillée à la table 3.5.

Les paramètres proposés heuristiquement par Leon sont $p = 2$ et $\sigma = 2$.

Le fait d'augmenter la taille de la sélection de sorte qu'elle ne soit pas réduite à k positions peut être assimilé à une méthode de filtrage des erreurs. En effet, au lieu, comme dans l'algorithme de Lee-Brickell, de calculer directement le poids de chaque mot de $(n-k)$ bits de la forme $(x_L|x_J) + (x_I + \epsilon_i)(A|B)$, on s'assure au préalable que sa restriction à l'ensemble L est de poids très faible. Si cette condition n'est pas satisfaite, il est très improbable que le poids total du mot soit faible ; poursuivre le calcul est alors, sinon inutile, au moins très peu rentable. Cela revient donc, dans un premier temps, à décoder jusqu'à la distance p le code de longueur $k + \sigma$ et de dimension e, obtenu à partir du code $\mathcal{C}$ en

Tant qu'un vecteur d'erreurs de poids inférieur ou égal à w n'a pas été trouvé

1. Choisir aléatoirement un ensemble S de $k+\sigma$ positions.
2. A l'aide d'une élimination de Gauss, mettre la matrice G sous la forme suivante



   et décomposer $x = (x_I|x_L|x_{J\setminus L})$, où I correspond aux indices des colonnes sélectionnées qui sont linéairement indépendantes, c'est-à-dire $S = I \cup L$ avec $|I| = rg(G_{|S}) = k'$.
3. Pour tous les vecteurs ϵ_i de longueur k' et de poids $i \leq p$,
   - calculer $w' = w(x_L + (x_I + \epsilon_i)A)$.
   - Si $w' \leq p - i$ alors, pour tout vecteur η de longueur $k - k'$, calculer $w(x_{J\setminus L} + (x_I + \epsilon_i)B + \eta C)$.
     Si ce poids est inférieur ou égal à $w - w' - i$, alors $e = (\epsilon_i \mid x_L + (x_I + \epsilon_i)A \mid x_{J\setminus L} + (x_I + \epsilon_i)B + \eta C)$.

TAB. 3.5 –: *Algorithme de Leon (version originale)*

poinçonnant les positions $\{1, \ldots, n\} \setminus S$. On accroît ainsi légèrement le nombre d'itérations effectuées par l'algorithme mais on diminue considérablement la nombre d'opérations requises par chacune d'elles.

La probabilité pour que l'ensemble S contienne au plus p positions erronées est

$$\frac{1}{\binom{n}{k+\sigma}} \sum_{i=0}^{p} \binom{n-w}{k+\sigma-i} \binom{w}{i}$$

La probabilité pour que les $k+\sigma$ colonnes de $G_{|S}$ soient de rang k' est obtenue en calculant la proportion de matrices de rang k' parmi les matrices binaires de taille $k \times (k+\sigma)$.

$$q_{k,\sigma}(k') = 2^{k'(k'-1)/2-(k+\sigma)k} \prod_{i=0}^{k'-1} \frac{(2^{k+\sigma-i}-1)(2^{k-i}-1)}{2^{i+1}-1}$$

A chaque itération, le nombre d'opérations effectuées par l'algorithme se dé-

compose donc en :

- $\frac{k^2 n}{2}$ opérations pour l'élimination de Gauss.
- $\frac{k'(n-k')}{2}$ opérations pour le calcul de $(x_L + x_I A | x_{J \setminus L} + x_I B)$.
- Pour chacune des $\binom{k'}{i}$ valeurs de ϵ_i, $(i+1)(k+\sigma-k')$ opérations pour calculer t.
- La probabilité que w' soit inférieur ou égal à $p-i$ est alors

$$\frac{1}{2^{k+\sigma-k'}} \sum_{j=0}^{p-i} \binom{k+\sigma-k'}{j}$$

  Dans le cas où cette condition est satisfaite, le calcul de $x_{J \setminus L} + x_I B + \epsilon_i B$ nécessite $i(n-k-\sigma)$ opérations. Il faut ensuite, pour chacune des $2^{k-k'}$ valeurs de η, $(\frac{k-k'}{2} + 1$ opérations sur $(n-k-\sigma)$ bits pour achever la procédure.

Toutefois, il apparaît clairement que lorsque le rang des $(k+\sigma)$ colonnes sélectionnées est inférieur à k,le nombre d'opérations est considérablement plus élevé puisque que l'algorithme doit passer en revue les $2^{k-k'}$ valeurs de η. La table 3.6 évalue le coût supplémentaire induit par le fait que S ne contienne pas d'ensemble d'information, lors du décodage d'un code binaire aléatoire [512,256] jusqu'à sa capacité de correction.

rang k' de la sélection S	k	$k-1$	$k-2$	$k-3$
fréquence	77,01 %	22,00 %	0,98 %	0,01 %
nb d'opérations binaires supplémentaires par rapport au cas $k'=k$	0	$2^{16,6}$	$2^{17,6}$	$2^{18,1}$

TAB. 3.6 –: *Variation, en fonction du rang des $k+\sigma$ colonnes sélectionnées, du nombre d'opérations binaires effectuées lors d'une itération de l'algorithme de Leon pour décoder un code binaire aléatoire [512,256] jusqu'à sa capacité de correction ($w = 28$), avec les paramètres proposés par Leon, $p = 2$ et $\sigma = 2$. Lorsque la sélection S contint un ensemble d'information, l'algorithme effectue 2^{24} opérations binaires.*

Comme je l'ai déjà noté au sujet de l'algorithme de décodage par ensembles d'information, imposer que les colonnes sélectionnées soient linéairement indépendantes ne nécessite aucune opération supplémentaire lors de l'élimination de Gauss. Il est donc préférable de modifier l'algorithme original de Leon et de ne choisir pour S que des ensembles contenant un ensemble d'information. Je décris à la table 3.7 ce nouvel algorithme que j'appelerai désormais algorithme de Leon.

Tant qu'un vecteur d'erreurs de poids inférieur ou égal à w n'a pas été trouvé

1. Choisir aléatoirement un ensemble d'information I et un ensemble L de σ positions de redondance.
2. A l'aide d'une élimination de Gauss, mettre la matrice G sous la forme suivante
$$U^{-1}G = \left(Id_k|Z_L|Z_{J\setminus L}\right)$$
et décomposer $x = (x_I|x_L|x_{J\setminus L})$.
3. Pour tous les vecteurs ϵ_i de longueur k et de poids $i \leq p$,
   - calculer $w' = w(x_L + (x_I + \epsilon_i)Z_L)$.
   - Si $w' \leq p - i$, calculer $w(x_{J\setminus L} + (x_I + \epsilon_i)Z_{J\setminus L})$. Si ce poids est inférieur ou égal à $w - w' - i$, alors $m = x_I + \epsilon_i$.

TAB. 3.7 –: *Algorithme de Leon (modifié)*

Proposition 3.10 (Facteur de travail de l'algorithme de Leon) *Le facteur de travail de l'algorithme de Leon (modifié) utilisé pour décoder jusqu'à la distance w est*

$$\begin{aligned} W_{LEON(p,\sigma)}(n,R,w) &= \frac{1}{\pi_{LEON(p,\sigma)}}\left[\frac{n^3R^2}{2} + \frac{n^2R(1-R)}{2}\right. \\ &+ \left.\sum_{i=0}^{p}(i+1)\binom{k}{i}\left(\sigma + (n(1-R)-\sigma)\frac{1}{2^\sigma}\sum_{j=0}^{p-i}\binom{\sigma}{j}\right)\right] \end{aligned}$$

$$\text{avec } \pi_{LEON(p,\sigma)} = \frac{1}{\binom{n}{nR+\sigma}}\sum_{i=0}^{p}\binom{n-w}{nR+\sigma-i}\binom{w}{i}$$

5.3 Algorithme de décodage utilisant un code poinçonné (DCP)

Cette nouvelle description conduit de façon naturelle à examiner un algorithme plus général. L'algorithme précédent commençait en effet par décoder jusqu'à la distance p un code poinçonné de longueur $(k + \sigma)$ et de dimension k — formé par la restriction de $\mathcal{C}$ à l'ensemble $I \cup L$ — en utilisant le décodage par ensembles d'information avec p effacements. Toutefois, rien ne justifie que la distance de décodage de ce code poinçonné soit limitée par le nombre d'effacements autorisés.

Une généralisation immédiate consiste donc à utiliser pour le code poinçonné l'algorithme de décodage par ensembles d'information avec p effacements jusqu'à la distance s, avec $p \leq s$. Les motifs d'erreurs corrigés par cet algorithme sont donc de la forme :

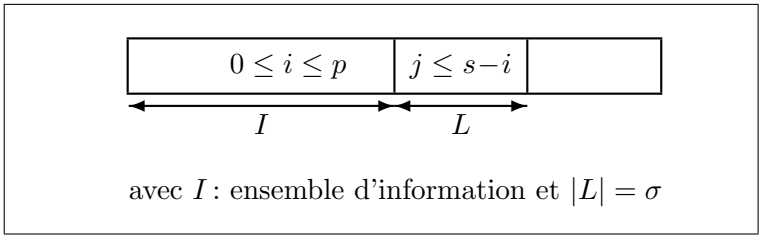


FIG. 3.6 –: *Motifs d'erreurs corrigés à chaque itération par l'algorithme DCP : on corrige tous les vecteurs d'erreurs dont la restriction à l'ensemble $I \cup L$ est de poids au plus s et tels que l'ensemble d'information I contient au plus p positions erronées.*

Ce nouvel algorithme est décrit à la table 3.8. Il correspond à l'algorithme de Lee-Brickell lorsque $\sigma = 0$ et $s = p$, et à celui de Leon pour $s = p$.

Tant qu'un vecteur d'erreurs de poids inférieur ou égal à w n'a pas été trouvé

1. Choisir aléatoirement un ensemble d'information I et un ensemble L de σ positions de redondance.
2. A l'aide d'une élimination de Gauss, mettre la matrice G sous la forme suivante
$$U^{-1}G = \left(Id_k | Z_L | Z_{J\setminus L}\right)$$
et décomposer $x = (x_I | x_L | x_{J\setminus L})$.
3. Pour tous les vecteurs ϵ_i de longueur k et de poids $i \leq p$,
   - calculer $w' = w(x_L + (x_I + \epsilon_i)Z_L)$.
   - Si $w' \leq s - i$, calculer $w(x_{J\setminus L} + (x_I + \epsilon_i)Z_{J\setminus L})$.
     Si ce poids est inférieur ou égal à $w - w' - i$, alors $m = x_I + \epsilon_i$.

TAB. 3.8 –: *Algorithme DCP*

Proposition 3.11 (Facteur de travail de l'algorithme DCP) *Le facteur de travail de l'algorithme DCP utilisé pour décoder jusqu'à la distance w est*

$$W_{DCP(p,\sigma,s)}(n,R,w) = \frac{1}{\pi_{DCP(p,\sigma,s)}}\left[\frac{n^3R^2}{2}+\frac{n^2R(1-R)}{2}\right.$$
$$\left. + \sum_{i=0}^{p}(i+1)\binom{k}{i}\left(\sigma+(n(1-R)-\sigma)\frac{1}{2^\sigma}\sum_{j=0}^{s-i}\binom{\sigma}{j}\right)\right]$$

$$avec\ \pi_{DCP(p,\sigma,s)} = \frac{\sum_{i=0}^{p}\binom{n-w}{nR-i}\binom{w}{i}}{\binom{n}{nR}}\ \frac{\sum_{j=0}^{s-i}\binom{n(1-R)-w+i}{\sigma-j}\binom{w-i}{j}}{\binom{n(1-R)}{\sigma}}$$

preuve : La probabilité que l'ensemble d'information contienne i positions erronées, avec $0 \leq i \leq p$, et que la sélection L en contienne au plus $s-i$ est

$$\pi_{DCP(p,\sigma,s)} = \frac{\sum_{i=0}^{p}\binom{n-w}{nR-i}\binom{w}{i}}{\binom{n}{nR}}\ \frac{\sum_{j=0}^{s-i}\binom{n(1-R)-w+i}{\sigma-j}\binom{w-i}{j}}{\binom{n(1-R)}{\sigma}}$$

Le nombre d'opérations effectuées à chaque itération est pratiquement identiquement à celui de l'algorithme de Leon modifié. Seule change la probabilité que l'on poursuive la procédure après avoir calculé le poids de $x_L + (x_I + \epsilon_i)Z_L$; elle vaut cette fois-ci

$$\frac{1}{2^\sigma}\sum_{j=0}^{s-i}\binom{\sigma}{j}$$

□

5.4 Algorithme de Stern (S)

De légères modifications dans sa présentation originale [Ste89b] rendent l'algorithme proposé par Jacques Stern très proche du précédent. En effet, il consiste également à décoder un code poinçonné en utilisant l'algorithme de décodage par ensembles d'information avec effacements. La seule différence est qu'il n'autorise que certains motifs d'effacements : il divise à chaque itération l'ensemble d'information I en deux parties égales, I_1 et I_2, et ne corrige que les vecteurs d'erreurs dont les restrictions à I_1 et à I_2 sont de poids exactement p, et qui s'annulent sur L.

Le choix de ces motifs d'erreurs particuliers est justifié par le fait qu'ils peuvent être détectés très facilement : le partage de l'ensemble d'information en deux parties égales induit un partage similaire des lignes de la matrice génératrice systématique $(Id_k, Z)_I$; les lignes de Z se décomposent donc en deux matrices Z_1 et Z_2. Introduire p effacements dans chacune des parties I_1 et I_2 consiste alors à ajouter au vecteur $x_J + x_I Z$ toutes les combinaisons linéaires Λ_1 de p lignes de la matrice Z_1 et tous les combinaisons linéaires Λ_2 de p lignes de la matrice Z_2. Imposer que la sélection L ne contienne aucune position erronée revient en fait à n'examiner que les combinaisons Λ_1 et Λ_2 dont la somme sur L vaut x_L. Cette procédure de décodage est détaillée à la table 3.9.

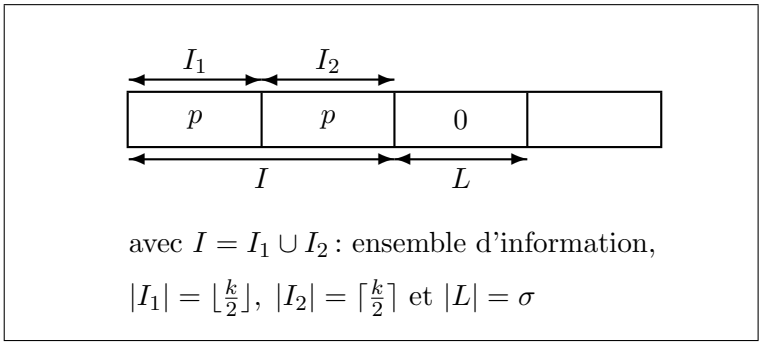


FIG. 3.7 –: *Motifs d'erreurs corrigés à chaque itération par l'algorithme de Stern : on corrige tous les vecteurs d'erreurs dont les restrictions à chacune des moitiés de l'ensemble d'information I sont de poids exactement p et qui s'annulent sur la sélection L.*

Une étude asymptotique de la complexité de l'algorithme de Stern a conduit Florent Chabaud à proposer les paramètres $p = 2$ où 3, selon la mémoire disponible, et $\sigma = \log_2 k$ [Cha92].

Proposition 3.12 (Facteur de travail de l'algorithme de Stern) *Le facteur de travail de l'algorithme de Stern utilisé pour décoder jusqu'à la distance w est*

$$\begin{aligned} W_{S(p,\sigma)}(n,R,w) &= \frac{1}{\pi_{S(p,\sigma)}} \left[\frac{n^3R^2}{2} + \frac{n^2R(1-R)}{2} + (2p+1)\sigma \binom{nR/2}{p} \right. \\ &+ \left. (2p+1)(n(1-R)-\sigma)\frac{1}{2^\sigma}\binom{nR/2}{p}^2 + K\left(\binom{nR/2}{p} + 2^\sigma\right)\right] \end{aligned}$$

$$\text{avec } \pi_{S(p,\sigma)} = \frac{\binom{n-w}{nR-2p}\binom{w}{2p}}{\binom{n}{nR}} \frac{\binom{2p}{p}}{4^p} \frac{\binom{n(1-R)-w+2p}{\sigma}}{\binom{n(1-R)}{\sigma}}$$

et K correspondant à la taille d'un mot machine ($K = 32$ ou 64).

preuve : La probabilité que le vecteur d'erreurs corresponde au motif de la figure 3.7 est égale à la probabilité que I contienne $2p$ positions erronées multipliée par la probabilité que I_1 en contienne p parmi ces $2p$, multipliée enfin par la probabilité que la sélection L soit exempte de toute erreur. On obtient donc

$$\pi_{S(p,\sigma)} = \frac{\binom{n-w}{k-2p}\binom{w}{2p}}{\binom{n}{k}} \frac{\binom{2p}{p}}{4^p} \frac{\binom{n-k-w+2p}{\sigma}}{\binom{n-k}{\sigma}}$$

Le nombre d'opérations binaires effectuées à chaque itération se décompose de la manière suivante :

– $\frac{nk^2}{2}$ opérations pour l'élimination de Gauss.

Tant qu'un vecteur d'erreurs de poids inférieur ou égal à w n'a pas été trouvé

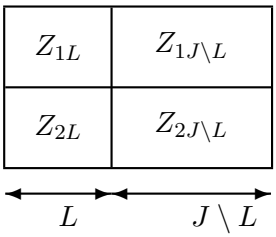
1. Choisir aléatoirement un ensemble d'information I et un ensemble L de σ positions de redondance.

2. A l'aide d'une élimination de Gauss, mettre la matrice G sous la forme suivante

$$U^{-1}G = \left(Id_k|Z_L|Z_{J\setminus L}\right)$$

et décomposer $x = (x_I|x_L|x_{J\setminus L})$.

3. Diviser aléatoirement les lignes de la matrice $(Z_L|Z_{J\setminus L})$ en deux parties de même taille



4. Pour tous les vecteurs ϵ_1 de longueur $k/2$ et de poids p, calculer $\epsilon_1 Z_{1L}$

5. Pour tous les vecteurs ϵ_2 de longueur $k/2$ et de poids p, calculer $x_L + x_I Z_L + \epsilon_2 Z_{2L}$.

6. Pour tout couple (ϵ_1, ϵ_2) tel que $\epsilon_1 Z_{1L} = x_L + x_I Z_L + \epsilon_2 Z_{2L}$, calculer $w(x_{J\setminus L} + x_I Z_{J\setminus L} + \epsilon_1 Z_{1J\setminus L} + \epsilon_2 Z_{2J\setminus L})$.
Si ce poids est inférieur ou égal à $w-2p$, alors $m = x_I + (\epsilon_1|\epsilon_2)$.

TAB. 3.9 –: *Algorithme de Stern*

– $\frac{k(n-k)}{2}$ opérations pour le calcul de $(x_L|x_J) + x_I Z$.

– Pour chacun des $\binom{k/2}{p}$ vecteur ϵ_1, le calcul de $\epsilon_1 Z_{1L}$ nécessite $p\sigma$ opérations binaires. De même, pour chacun des $\binom{k/2}{p}$ vecteur ϵ_2, le calcul de $x_L + x_I Z_L + \epsilon_2 Z_{2L}$ nécessite $(p+1)\sigma$ opérations binaires.

– Le nombre de collisions, c'est-à-dire de couples (ϵ_1, ϵ_2) tels que $\epsilon_1 Z_{1L} = x_L + x_I Z_L + \epsilon_2 Z_{2L}$, est alors

$$\frac{1}{2^\sigma}\binom{k/2}{p}^2$$

et pour chacune de ces collisions, l'algorithme effectue $(2p)$ additions sur des mots de $(n-k-\sigma)$ bits et un calcul de poids, c'est-à-dire au total $(n-k-\sigma)(2p+1)$ opérations.

– Il faut enfin tenir compte du coût de l'allocation dynamique de la mémoire requise pour le stockage des valeurs des vecteurs de σ bits $\epsilon_1 Z_{1L}$ et $x_L + x_I Z_L + \epsilon_2 Z_{2L}$. Ce coût est estimé à $K\left(p\binom{k/2}{p} + 2^\sigma\right)$ opérations binaires, où K est la taille d'un mot-machine — en général $K = 32$ ou 64.

□

5.5 Applications des algorithmes classiques de décodage à la cryptanalyse des systèmes à mots de poids faibles

Le tableau 3.10 donne le facteur de travail requis par les algorithmes classiques pour attaquer les cryptosystèmes à mots de poids faible de McEliece, Stern et Véron. Pour chacun de ces algorithmes de décodage, j'ai utilisé les paramètres proposés par leurs auteurs.

cryptosystèmes	McEliece	Stern	Véron
code utilisé	[1024,524]	[524,256]	[512,120]
distance de décodage w	50	56	114
décodage par ensembles d'information	$2^{80,7}$	$2^{84,9}$	$2^{72,5}$
algorithme de Lee-Brickell	$p = 2$ $2^{71,2}$	$p = 2$ $2^{74,9}$	$p = 2$ $2^{63,9}$
algorithme de Leon	$p = 2$ $\sigma = 2$ $2^{70,7}$	$p = 2$ $\sigma = 2$ $2^{74,9}$	$p = 2$ $\sigma = 2$ $2^{63,8}$
algorithme de Stern	$p = 2$ $\sigma = 9$ $2^{69,9}$	$p = 2$ $\sigma = 8$ $2^{73,5}$	$p = 2$ $\sigma = 7$ $2^{62,0}$

TAB. 3.10 –: *Facteur de travail des algorithmes de décodage classiques pour cryptanalyser les systèmes à mots de poids faibles*

6 Optimisation des algorithmes de décodage classiques

J'ai montré que les diverses généralisations du décodage par ensembles d'information pouvaient en fait être synthétisées en deux algorithmes : celui du décodage utilisant un code poinçonné et celui de Jacques Stern. Toutefois, ces deux méthodes peuvent être implémentées sous différentes formes : on peut par exemple les utiliser, soit comme des algorithmes de décodage, soit comme des algorithmes de recherche de mots de poids faible. De même, on peut préférer les mettre en œuvre sous leur forme duale, qui manipule une matrice de parité du code $\mathcal{C}$ au lieu d'une matrice génératrice. Ces modifications, certes minimes, induisent cependant quelques variations du facteur de travail. Aussi un calcul précis du nombre d'opérations effectuées permet-il à la fois, en fonction de la taille du code étudié et de la distance de décodage, d'optimiser les paramètres des ces algorithmes et d'en choisir la version la plus efficace.

6.1 Décodage vs. recherche d'un mot de poids minimal

Comme je l'ai déjà noté, les paramètres des cryptosystèmes à mots de poids faible impliquent que le code linéaire $\mathcal{C}'$ de longueur n et de dimension $k+1$, formé par la réunion du code $\mathcal{C}$ et du coset $\mathcal{C}+x$, a pour unique mot de poids minimal le vecteur recherché e. Plutôt que d'utiliser un algorithme de décodage jusqu'à la distance w, on peut donc se servir d'un algorithme de recherche de mots de poids minimal pour cryptanalyser les systèmes à mots de poids faible.

Il apparaît alors que les algorithmes de décodage que je viens de présenter peuvent également être appréhendés de ce point de vue. C'était d'ailleurs l'approche originale des algorithmes de Leon et de Stern. En effet, il suffit alors de reprendre les algorithmes précédents en les appliquant au code $\mathcal{C}'$ et en considérant que le vecteur x est nul. Les tables 3.11 et 3.12 présentent respectivement les variantes de l'algorithme DCP et de celui de Stern pour la recherche de mots de poids minimal.

Les facteurs de travail correspondant sont alors légèrement modifiés puisque la dimension du code étudié est dans ce cas égale à $k+1$. Par contre, la procédure qui consistait à calculer le vecteur $x_J + x_I Z$ disparaît. Le nombre d'itérations effectuées par l'algorithme DCP devient

$$\frac{1}{\pi_{DCP(p,\sigma,s)}} = \frac{\binom{n}{nR}}{\sum_{i=1}^{p}\binom{n-w}{nR-i}\binom{w}{i}} \frac{\binom{n(1-R)}{\sigma}}{\sum_{j=0}^{s-i}\binom{n(1-R)-w+i}{\sigma-j}\binom{w-i}{j}}$$

puisque les mots de poids w trouvés à chaque itération contienne au moins un bit égal à 1 dans l'ensemble d'information.

6.2 Approche duale

Tels que je les ai décris, les algorithmes de décodage classiques manipulent tous une matrice génératrice du code $\mathcal{C}$. Toutefois, on peut de façon équivalente

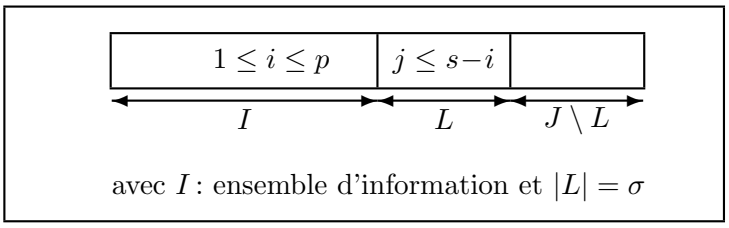


FIG. 3.8 –: *Forme des mots de poids de poids minimal détectés à chaque itération par l'algorithme DCP*

Tant qu'un mot de poids inférieur ou égal à w n'a pas été trouvé

1. Choisir aléatoirement un ensemble d'information I et un ensemble L de σ positions de redondance.
2. A l'aide d'une élimination de Gauss, mettre la matrice G sous la forme suivante
$$U^{-1}G = (Id_k|Z)$$
3. Pour toutes les combinaisons linéaires Λ_i de i lignes de la matrice Z, pour $1 \leq i \leq p$,
   - calculer le poids w' de la restriction à L de Λ_i.
   - Si $w' \leq s - i$, alors calculer $w(\Lambda_{i|J\setminus L})$.
     Si ce poids est inférieur ou égal à $w - w' - i$, la somme des lignes de la matrice G correspondant à Λ_i est un mot de poids minimal.

TAB. 3.11 –: *Algorithme DCP utilisé pour la recherche de mots de poids minimal*

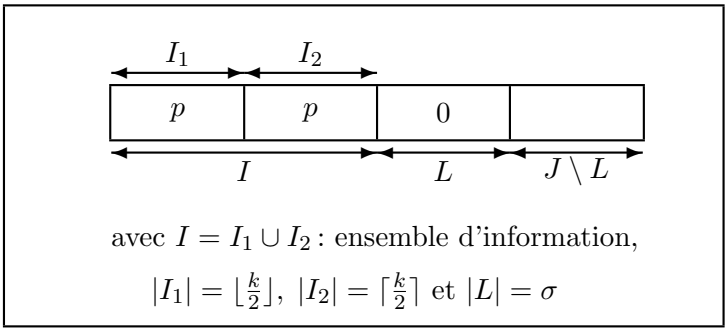


FIG. 3.9 –: *Forme des mots de poids de poids minimal détectés à chaque itération par l'algorithme de Stern*

Tant qu'un mot de poids inférieur ou égal à w n'a pas été trouvé

1. Choisir aléatoirement un ensemble d'information I et un ensemble L de σ positions de redondance.
2. A l'aide d'une élimination de Gauss, mettre la matrice G sous la forme suivante
$$U^{-1}G = (Id_k|Z)$$
3. Diviser aléatoirement les lignes de la matrice Z en deux parties de même taille, Z_1 et Z_2.
4. Calculer la restriction à L de toutes les combinaisons linéaires Λ_1 de p lignes de Z_1.
5. Calculer la restriction à L de toutes les combinaisons linéaires Λ_2 de p lignes de Z_2.
6. Pour tout couple (Λ_1, Λ_2) tel que $\Lambda_{1|L} = \Lambda_{2|L}$,
calculer $w((\Lambda_1 + \Lambda_2)_{J\setminus L})$.
Si ce poids est inférieur ou égal à $w - 2p$, la somme des lignes de G correspondant à $\Lambda_1 + \Lambda_2$ est un mot de poids minimal.

TAB. 3.12 –: *Algorithme de Stern utilisé pour la recherche de mots de poids minimal*

travailler avec une matrice de parité. La table 3.13 détaille par exemple la version duale de l'algorithme de Stern — qui est d'ailleurs sa version originale.

La seule différence introduite dans le calcul du facteur de travail porte sur le coût de l'élimination de Gauss qui est évalué à $\frac{n(n-k)^2}{2}$ opérations au lieu de $\frac{nk^2}{2}$. Cependant, comme l'élimination de Gauss opère sur les lignes de la matrice H tandis que les autres opérations opèrent sur ses colonnes, il est nécessaire de modifier la représentation en mémoire de sa partie redondante Z après la mise sous forme systématique. Cette conversion requiert par conséquent $n(n-k)$ opérations binaires supplémentaires.

Tant qu'un vecteur d'erreurs de poids inférieur ou égal à w n'a pas été trouvé

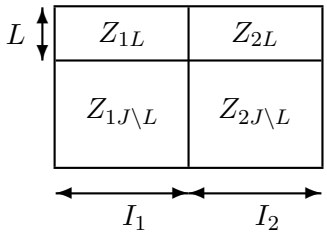
1. Choisir aléatoirement un ensemble de redondance J et un sous-ensemble L de σ positions de J.

2. A l'aide d'une élimination de Gauss, mettre la matrice H sous la forme suivante

$$U^{-1}H = (Z|Id_{n-k})$$

et décomposer $x = (x_I|x_L|x_{J\setminus L})$.

3. Diviser aléatoirement les colonnes de la matrice Z en deux parties de même taille et partitionner ses lignes suivant la décomposition $J = L \cup (J \setminus L)$.



4. Pour tous les vecteurs ϵ_1 de longueur $k/2$ et de poids p, calculer $\epsilon_1 \, {}^tZ_{1L}$

5. Pour tous les vecteurs ϵ_2 de longueur $k/2$ et de poids p, calculer $x_L + x_I \, {}^tZ_L + \epsilon_2 \, {}^tZ_{2L}$.

6. Pour tout couple (ϵ_1, ϵ_2) tel que $\epsilon_1 \, {}^tZ_{1L} = x_L + x_I \, {}^tZ_L + \epsilon_2 \, {}^tZ_{2L}$,
calculer $w(x_{J\setminus L} + x_I \, {}^tZ_{J\setminus L} + \epsilon_1 \, {}^tZ_{1J\setminus L} + \epsilon_2 Z_{2J\setminus L})$.
Si ce poids est inférieur ou égal à $w-2p$, alors $m = x_I + (\epsilon_1|\epsilon_2)$.

TAB. 3.13 –: *Algorithme de Stern, version duale*

Le choix de manipuler une matrice génératrice ou une matrice de parité du code est donc dicté par la valeur de son taux d'expansion R.

6.3 Optimisation des paramètres

Ces deux types de modifications conduisent, pour l'algorithme DCP et celui de Stern, à quatre versions différentes. Je les désignerai par les lettres D ou M selon qu'il s'agit d'un algorithme de Décodage ou de recherche de mots de poids Minimal, et par les lettres G et P selon qu'elles manipulent une matrice Génératrice ou une matrice de Parité.

Les légères modifications du facteur de travail de ces algorithmes sont résumées dans les deux propositions suivantes.

Proposition 3.13 (Facteur de travail des différentes versions de l'algorithme DCP) *Le facteur de travail de l'algorithme DCP est :*

– *lorsqu'il est utilisé pour décoder un code $[n, nR]$ jusqu'à la distance w*

$$\begin{aligned} W_{DCP_{GD}(p,\sigma,s)} &= \frac{1}{\pi_{DCP_D(p,\sigma,s)}}\left[\frac{n^3R^2}{2} + \frac{n^2R(1-R)}{2} + J_{DCP_D(p,\sigma,s)}\right] \\ W_{DCP_{PD}(p,\sigma,s)} &= \frac{1}{\pi_{DCP_D(p,\sigma,s)}}\left[\frac{n^3(1-R)^2}{2} + \frac{3n^2R(1-R)}{2} + J_{DCP_D(p,\sigma,s)}\right] \end{aligned}$$

$$\begin{aligned} \text{avec } \pi_{DCP_D(p,\sigma,s)} &= \frac{\binom{n}{nR}}{\sum_{i=0}^{p}\binom{n-w}{nR-i}\binom{w}{i}} \frac{\binom{n(1-R)}{\sigma}}{\sum_{j=0}^{s-i}\binom{n(1-R)-w+i}{\sigma-j}\binom{w-i}{j}} \\ \text{et } J_{DCP_D(p,\sigma,s)} &= \sum_{i=0}^{p}(i+1)\binom{k}{i}\left(\sigma + (n(1-R)-\sigma)\frac{1}{2^\sigma}\sum_{j=0}^{s-i}\binom{\sigma}{j}\right) \end{aligned}$$

– *lorsqu'il est utilisé pour rechercher un mot de poids minimal w dans un code $[n, nR]$*

$$\begin{aligned} W_{DCP_{GM}(p,\sigma,s)} &= \frac{1}{\pi_{DCP_M(p,\sigma,s)}}\left[\frac{n^3R^2}{2} + J_{DCP_M(p,\sigma,s)}\right] \\ W_{DCP_{PM}(p,\sigma,s)} &= \frac{1}{\pi_{DCP_M(p,\sigma,s)}}\left[\frac{n^3(1-R)^2}{2} + n^2R(1-R) + J_{DCP_M(p,\sigma,s)}\right] \end{aligned}$$

$$\begin{aligned} \text{avec } \pi_{DCP_M(p,\sigma,s)} &= \frac{\binom{n}{nR}}{\sum_{i=1}^{p}\binom{n-w}{nR-i}\binom{w}{i}} \frac{\binom{n(1-R)}{\sigma}}{\sum_{j=0}^{s-i}\binom{n(1-R)-w+i}{\sigma-j}\binom{w-i}{j}} \\ \text{et } J_{DCP_M(p,\sigma,s)} &= \sum_{i=1}^{p} i\binom{k}{i}\left(\sigma + (n(1-R)-\sigma)\frac{1}{2^\sigma}\sum_{j=0}^{s-i}\binom{\sigma}{j}\right) \end{aligned}$$

Proposition 3.14 (Facteur de travail des différentes versions de l'algorithme de Stern) *Le facteur de travail de l'algorithme de Stern est :*

– *lorsqu'il est utilisé pour décoder un code $[n, nR]$ jusqu'à la distance w*

$$\begin{aligned} W_{S_{GD}(p,\sigma,s)} &= \frac{1}{\pi_{S(p,\sigma,s)}}\left[\frac{n^3R^2}{2}+\frac{n^2R(1-R)}{2}+J_{S_D(p,\sigma,s)}\right] \\ W_{S_{PD}(p,\sigma,s)} &= \frac{1}{\pi_{S(p,\sigma,s)}}\left[\frac{n^3(1-R)^2}{2}+\frac{3n^2R(1-R)}{2}+J_{S_D(p,\sigma,s)}\right] \end{aligned}$$

$$\begin{aligned} \text{avec } \pi_{S(p,\sigma,s)} &= \frac{\binom{n-w}{nR-2p}\binom{w}{2p}}{\binom{n}{nR}}\,\frac{\binom{2p}{p}}{4^p}\,\frac{\binom{n(1-R)-w+2p}{\sigma}}{\binom{n(1-R)}{\sigma}} \\ \text{et } J_{S_D(p,\sigma,s)} &= (2p+1)\sigma\binom{nR/2}{p}+(2p+1)(n(1-R)-\sigma)\frac{\binom{nR/2}{p}^2}{2^\sigma} \\ &+ K\left(\binom{k/2}{p}+2^\sigma\right) \end{aligned}$$

– *lorsqu'il est utilisé pour rechercher un mot de poids minimal w dans un code $[n, nR]$*

$$\begin{aligned} W_{S_{GM}(p,\sigma,s)} &= \frac{1}{\pi_{S(p,\sigma,s)}}\left[\frac{n^3R^2}{2}+n^2R(1-R)+J_{S_M(p,\sigma,s)}\right] \\ W_{S_{PM}(p,\sigma,s)} &= \frac{1}{\pi_{S(p,\sigma,s)}}\left[\frac{n^3(1-R)^2}{2}+n^2R(1-R)+J_{S_M(p,\sigma,s)}\right] \end{aligned}$$

$$\text{avec } J_{S_M(p,\sigma,s)} = 2p\sigma\binom{nR/2}{p}+2p(n(1-R)-\sigma)\frac{\binom{nR/2}{p}^2}{2^\sigma}+K\left(\binom{k/2}{p}+2^\sigma\right)$$

Ces résultats permettent donc d'optimiser les paramètres des algorithmes pour des tailles de codes données et d'en choisir la version la moins coûteuse. Les tables 3.14 et 3.15 traduisent les fruits de ces optimisations, qui améliorent notablement les facteurs de travail des attaques des systèmes à mots de poids faible calculés à la table 3.10.

cryptosystèmes	McEliece	Stern	Véron
code utilisé	[1024,524]	[512,256]	[512,120]
distance de décodage w	50	56	114
algorithme DCP, GD	$p=3$ $\sigma=11$ $s=4$ $2^{69,6}$	$p=3$ $\sigma=17$ $s=4$ $2^{73,5}$	$p=3$ $\sigma=19$ $s=8$ $2^{62,5}$
algorithme DCP, PD	$p=3$ $\sigma=11$ $s=4$ $2^{69,5}$	$p=3$ $\sigma=17$ $s=4$ $2^{73,5}$	$p=3$ $\sigma=20$ $s=9$ $2^{63,3}$
algorithme DCP, GM	$p=3$ $\sigma=11$ $s=4$ $2^{69,3}$	$p=3$ $\sigma=17$ $s=4$ $2^{73,5}$	$p=3$ $\sigma=19$ $s=8$ $2^{62,6}$
algorithme DCP, PM	$p=3$ $\sigma=11$ $s=4$ $2^{69,3}$	$p=3$ $\sigma=17$ $s=4$ $2^{73,5}$	$p=3$ $\sigma=20$ $s=9$ $2^{63,6}$
Meilleure version	GM ou PM $p=3$ $\sigma=11$ $s=4$ $2^{69,3}$	identiques $p=3$ $\sigma=17$ $s=4$ $2^{73,5}$	GD $p=3$ $\sigma=19$ $s=8$ $2^{62,5}$

TAB. 3.14 –: *Facteur de travail des différentes versions optimisées de l'algorithme de décodage utilisant un code poinçonné (DCP) pour cryptanalyser les systèmes à mots de poids faibles*

cryptosystèmes	McEliece	Stern	Véron
code utilisé	[1024,524]	[512,256]	[512,120]
distance de décodage w	50	56	114
algorithme de Stern, GD	$p=2$ $\sigma=17$ $2^{66,2}$	$p=2$ $\sigma=13$ $2^{71,6}$	$p=3$ $\sigma=17$ $2^{61,9}$
algorithme de Stern, PD	$p=2$ $\sigma=17$ $2^{66,0}$	$p=2$ $\sigma=13$ $2^{71,6}$	$p=3$ $\sigma=16$ $2^{62,7}$
algorithme de Stern, GM	$p=2$ $\sigma=16$ $2^{66,2}$	$p=2$ $\sigma=13$ $2^{71,8}$	$p=3$ $\sigma=17$ $2^{62,2}$
algorithme de Stern, PM	$p=2$ $\sigma=16$ $2^{66,1}$	$p=2$ $\sigma=13$ $2^{71,8}$	$p=3$ $\sigma=16$ $2^{63,0}$
Meilleure version	PD $p=2$ $\sigma=17$ $2^{66,0}$	GD ou PD $p=2$ $\sigma=13$ $2^{71,6}$	GD $p=3$ $\sigma=17$ $2^{61,9}$

TAB. 3.15 –: *Facteur de travail des différentes versions optimisées de l'algorithme de Stern pour cryptanalyser les systèmes à mots de poids faibles*

Chapitre 4

Un algorithme de décodage itératif

Les meilleurs algorithmes généraux de décodage et de recherche de mots de poids minimal dans un code linéaire aléatoire sont tous des variantes de l'algorithme de décodage par ensembles d'information. Ils sont par conséquent tous conçus suivant le même schéma : ils explorent une succession d'ensembles d'information choisis aléatoirement, et pour chacun d'eux, ils effectuent une élimination de Gauss sur la matrice génératrice du code puis quelques opérations sur sa forme systématique. Les algorithmes introduits par Lee et Brickell, Leon, et Stern visaient tous à réduire le coût relatif de l'élimination de Gauss dans le décodage par ensembles d'information, puisque, initialement, cette procédure représentait 99,9 % des calculs pour décoder un code aléatoire de longueur 512 et de dimension 256. Bien que le temps de calcul soit mieux réparti entre les différentes procédures dans l'algorithme DCP et dans celui de Stern, la mise sous forme systématique de la matrice génératrice reste relativement coûteuse — surtout quand le nombre d'effacements p est faible.

Cette constatation m'a donc amené, en collaboration avec Hervé Chabanne et Florent Chabaud, à imaginer un nouvel algorithme de décodage dans lequel les formes systématiques successives de la matrice G pourraient se déduire les unes des autres sans que l'on ait recours à une élimination de Gauss. Cela impose alors que les différents ensembles d'information ne soient plus indépendants. Cette technique est en fait similaire à celle qui est employée dans l'algorithme du simplexe. J'ai donc modifié en ce sens les algorithmes probabilistes décrits au chapitre précédent. La disparition de l'élimination de Gauss diminue considérablement le nombre d'opérations effectuées à chaque itération ; toutefois, je montrerai que le nombre moyen d'itérations augmente dès lors que les ensembles d'information examinés ne sont plus indépendants. Malgré tout, cela permet d'améliorer les attaques précédentes des cryptosystèmes à mots de poids faible.

Après avoir présenté le principe général de cet algorithme itératif, je montrerai que son facteur de travail peut être déterminé par l'étude détaillée d'un

processus markovien, et je justifierai cette analyse par quelques résultats expérimentaux. J'appliquerai ensuite ces résultats aux problèmes du décodage et de la recherche de mots de poids minimal dans un code binaire aléatoire, puis à la cryptanalyse des principaux systèmes à mots de poids faible.

1 Principe de l'algorithme itératif

Même s'ils améliorent considérablement de ce point de vue le décodage par ensembles d'information, les diverses versions de l'algorithme DCP et de celui de Stern consacrent une partie importante de leur temps d'exécution à effectuer des éliminations de Gauss sur la matrice génératrice, ou la matrice de parité, du code étudié. Cette constatation générale est explicitée dans le tableau 4.1, qui évalue le coût relatif de l'élimination de Gauss dans les versions optimisées de ces deux algorithmes, lorsqu'ils sont utilisés attaquer les principaux cryptosystèmes à mots de faible. Il apparaît de plus que ce coût croît considérablement lorsque le nombre d'effacements p autorisés par l'algorithme est faible, ce qui est en particulier très contraignant pour l'algorithme de Stern qui est limité à de faibles valeurs de p du fait de la taille de la mémoire utilisée.

cryptosystèmes	McEliece		Stern		Véron	
code utilisé	[1024,524]		[512,256]		[512,120]	
w	50		56		114	
algo. DCP	GM $p = 3$ $\sigma = 11$ $s = 4$	$12,3$ %	GM $p = 3$ $\sigma = 17$ $s = 4$	$10,5$ %	GM $p = 3$ $\sigma = 19$ $s = 8$	$9,4$ %
algo. de Stern	PD $p = 2$ $\sigma = 17$	$80,5$ %	GD $p = 2$ $\sigma = 13$	$60,2$ %	GD $p = 3$ $\sigma = 17$	$9,6$ %

TAB. 4.1 –: *Coût relatif de l'élimination de Gauss dans le facteur de travail des algorithmes DCP et Stern pour la cryptanalyse des systèmes à mots de poids faible*

Il est donc souhaitable de s'affranchir de cette procédure coûteuse en utilisant une technique similaire à celle du simplexe. L'emploi de cette méthode fut auparavant suggérée par Omura [Omu72], qui proposa un algorithme d'optimisation linéaire en nombres entiers par descente pour le décodage, puis reprise par van Tilburg [vT88, vT94] sous le nom de "bit swapping" et H. Chabanne et B. Corteau [CC93]. Elle consiste à ne plus examiner à chaque nouvelle itération un ensemble d'information complètement indépendant du précédent — ce qui nécessite une élimination de Gauss complète pour mettre la matrice génératrice sous forme systématique —, mais à en choisir un qui ne diffère du précédent que par un élément. J'ai formalisé cette idée par la notion d'*ensembles d'information voisins*.

Définition 4.1 (Ensembles d'information voisins) *Deux ensembles d'in-*

formation I et I' sont voisins *si et seulement si*

$$\exists \lambda \in I,\ \exists \mu \in \{1, \ldots, n\} \setminus I,\ \textit{tels que } I' = (I \setminus \{\lambda\}) \cup \{\mu\}$$

Comme deux ensembles d'information quelconques pour un même code peuvent toujours être joints par une suite finie d'ensembles d'information voisins, cette méthode peut être utilisée pour générer les ensembles d'information successifs examinés dans les algorithmes de décodage précédemment étudiés. La condition à imposer aux éléments λ et μ pour que, si I est un ensemble d'information, $(I \setminus \{\lambda\}) \cup \{\mu\}$ le soit également est extrêmement simple.

Proposition 4.2 *Soit I un ensemble d'information pour le code $\mathcal{C}$ et $U^{-1}G = (Id_k, Z)_I$ la forme systématique associée de la matrice génératrice. Soit λ un élément de I et μ un élément de son complémentaire J. Alors, l'ensemble $I' = (I \setminus \{\lambda\}) \cup \{\mu\}$ est un ensemble d'information pour le code $\mathcal{C}$ si et seulement si $z_{\lambda,\mu} = 1$.*

preuve : L'expression de la matrice G sous forme systématique induit les relations de dépendance linéaire suivantes sur les colonnes de G

$$\forall j \in J,\ G_j = \sum_{i \in I} z_{i,j} G_i$$

Les colonnes d'indice λ et μ sont donc en particulier liées par

$$G_\mu = z_{\lambda,\mu} G_\lambda + \sum_{i \in I \setminus \{\lambda\}} Z_{i,\mu} G_i$$

Aussi les colonnes G_μ et $(G_i)_{i \in I \setminus \{\lambda\}}$ sont-elles linéaires indépendantes si et seulement si $z_{\lambda,\mu} = 1$. □

Le choix d'un ensemble d'information voisin du précédent à chaque nouvelle itération permet alors de mettre à jour la forme systématique de la matrice génératrice à l'aide d'une seule procédure de pivot, et non plus d'une élimination complète.

Proposition 4.3 (Remise à jour de la matrice systématique) *Soit I et I' deux ensembles d'information voisins tels que $I' = (I \setminus \{\lambda\}) \cup \{\mu\}$, et $(Id_k, Z)_I$ et $(Id_k, Z')_{I'}$ les matrices génératrices systématiques associées. Alors Z' se déduit de Z par*

- $\forall j \in J',\ \ z'_{\mu,j} = z_{\lambda,j}$
- $\forall i \in I' \setminus \{\mu\}$,
  - $\forall j \in J' \setminus \{\lambda\},\ \ z'_{i,j} = z_{i,j} + z_{i,\mu} z_{\lambda,j}$
  - $z'_{i,\lambda} = z_{i,\mu}$

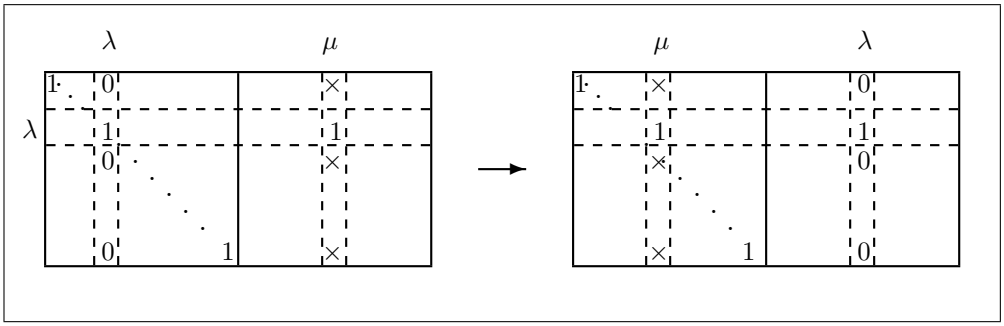


FIG. 4.1 –: *Remise à jour de la forme systématique de la matrice génératrice pour des ensembles d'information voisins*

La matrice systématique associée à I' est en effet obtenue en échangeant les colonnes d'indice λ et μ de la matrice précédente. Comme la colonne d'indice λ appartient à la matrice Identité, il suffit pour cela d'une simple procédure de pivot en position (λ, μ), *i.e.* d'additionner la ligne d'indice λ de la matrice à toutes les autres lignes possédant un 1 sur la colonne μ.

L'emploi de cette procédure considérablement moins coûteuse permet donc d'améliorer les algorithmes classiques de décodage et de recherche de mots de poids minimal. Les tables 4.2 et 4.3 décrivent respectivement l'algorithme DCP pour le décodage (version GD) et l'algorithme de Stern pour la recherche de mots de poids minimal (version GM) ainsi modifiés.

2 Modélisation des algorithmes itératifs par un processus markovien

Contrairement à celles des algorithmes présentés au chapitre 3, les itérations successives des algorithmes itératifs que je viens de décrire ne sont pas indépendantes. La méthode utilisée précédemment pour calculer le nombre moyen d'itérations n'est donc plus valable, et il est nécessaire de modéliser les algorithmes itératifs par un processus stochastique à temps discret.

2.1 Description du modèle

Cette modélisation consiste à associer à chaque itération de l'algorithme de décodage une variable aléatoire qui décrit la distribution des positions erronées suivant la décomposition $(I, L, J \setminus L)$ des coordonnées. Quand il s'agit d'un algorithme de recherche de mots de poids minimal, on considère de manière similaire la distribution des positions du support du mot de poids minimal. L'algorithme atteint alors un état de succès lorsque cette décomposition correspond à un des motifs d'erreurs décodés par cette itération. Dans la plupart des cas, il suffit pour décrire l'itération de considérer le nombre de positions erronées contenues par l'ensemble d'information. Toutefois, quand ce nombre

Initialisation :
Choisir aléatoirement un ensemble d'information I, mettre la matrice G sous forme systématique $U^{-1}G = (Id_k, Z)_I$ à l'aide d'une élimination de Gauss et calculer $y = x_J + x_I Z$ où $x = (x_I, x_J)_I$.

Tant qu'un vecteur d'erreurs de poids inférieur ou égal à w n'a pas été trouvé :

1. Choisir aléatoirement un ensemble L de σ positions de redondance, décomposer la matrice Z sous la forme $(Z_L, Z_{J\setminus L})$ et $y = (y_L, y_{J\setminus L})$.
2. Pour tous les vecteurs ϵ_i de longueur k et de poids $i \leq p$,
   - calculer $w' = w(y_L + \epsilon_i Z_L)$.
   - Si $w' \leq s - i$, alors calculer $w(y_{J\setminus L} + \epsilon_i Z_{J\setminus L})$.
     Si ce poids est inférieur ou égal à $w - w' - i$,
     alors le vecteur d'erreurs est $e = (\epsilon_i, y + \epsilon_i Z)_I$.
3. Choisir aléatoirement λ dans I et μ dans son complémentaire J.
   Remplacer I par $(I \setminus \{\lambda\}) \cup \{\mu\}$ en mettant à jour la matrice Z et le vecteur y par la procédure :
   - $\forall j \in (J \setminus \{\mu\}) \cup \{\lambda\}, \;\; z_{\mu,j} \leftarrow z_{\lambda,j}$
   - $\forall i \in I \setminus \{\lambda\}, :$
     - $\forall j \in J \setminus \{\mu\}, \;\; z_{i,j} \leftarrow z_{i,j} + z_{i,\mu} z_{\lambda,j}$
     - $z_{i,\lambda} \leftarrow z_{i,\mu}$
4. $\forall j \in (J \setminus \{\mu\}), \;\; y_j \leftarrow y_j + y_\mu z_{\lambda,j}$ et $y_\lambda \leftarrow y_\mu$.

TAB. 4.2 –: *Algorithme DCP itératif pour le décodage (version GD)*

Initialisation :
Choisir aléatoirement un ensemble d'information I et mettre la matrice G sous forme systématique $U^{-1}G = (Id_k, Z)_I$ à l'aide d'une élimination de Gauss.

Tant qu'un mot de poids w n'a pas été trouvé :

1. Choisir aléatoirement un ensemble L de σ positions de redondance ; diviser aléatoirement les lignes de la matrice Z en deux parties de même taille, Z_1 et Z_2.

2. Calculer la restriction à L de toutes les combinaisons linéaires Λ_1 de p lignes de Z_1.

3. Calculer la restriction à L de toutes les combinaisons linéaires Λ_2 de p lignes de Z_2.

4. Pour tout couple (Λ_1, Λ_2) tel que $\Lambda_{1|L} = \Lambda_{2|L}$,
   calculer $w((\Lambda_1 + \Lambda_2)_{J\setminus L})$.
   Si ce poids est inférieur ou égal à $w - 2p$, la somme des lignes de G correspondant à $\Lambda_1 + \Lambda_2$ est un mot de poids w.

5. Choisir aléatoirement λ dans I et μ dans son complémentaire J.
   Remplacer I par $(I \setminus \{\lambda\}) \cup \{\mu\}$ en mettant à jour la matrice Z par la procédure :

   – $\forall j \in (J \setminus \{\mu\}) \cup \{\lambda\},\ \ z_{\mu,j} \leftarrow z_{\lambda,j}$
   – $\forall i \in I \setminus \{\lambda\}$, :
   - $\forall j \in J \setminus \{\mu\},\ \ z_{i,j} \leftarrow z_{i,j} + z_{i,\mu} z_{\lambda,j}$
   - $z_{i,\lambda} \leftarrow z_{i,\mu}$

TAB. 4.3 –: *Algorithme de Stern itératif pour la recherche de mots de poids minimal (version GM)*

correspond à un des effacements autorisés par l'algorithme, il est nécessaire de distinguer si l'on se trouve dans un état de succès ou un état d'échec.

Définition 4.4 (Processus stochastique associé à l'algorithme de décodage DCP) *Soit $(I, L, J \setminus L)$ la décomposition de $\{1, \ldots, n\}$ associée à l'itération i de l'algorithme de décodage DCP. A cette itération est associée une variable aléatoire X_i prenant ses valeurs dans l'espace des états*

$$\mathcal{E} = \{0_F, 1_F, \ldots, p_F\} \cup \{0_S, 1_S, \ldots, p_S\} \cup \{p+1, \ldots, w\}$$

de la manière suivante

$$\begin{aligned}
X_i = u_F \quad & ssi \quad |I \cap supp(e)| = u \ et \ |L \cap supp(e)| > s - u, \ \ \forall\, 0 \leq u \leq p \\
X_i = u_S \quad & ssi \quad |I \cap supp(e)| = u \ et \ |L \cap supp(e)| \leq s - u, \ \ \forall\, 0 \leq u \leq p \\
X_i = v \quad & ssi \quad |I \cap supp(e)| = v, \ \ \forall\, p+1 \leq v \leq w
\end{aligned}$$

L'ensemble des états de succès est donc

$$\mathcal{S} = \{0_S, 1_S, \ldots, p_S\}$$

et celui des états d'échec

$$\mathcal{F} = \{0_F, 1_F, \ldots, p_F\} \cup \{p+1, \ldots, w\}$$

Définition 4.5 (Processus stochastique associé à l'algorithme de décodage de Stern) *Soit $(I_1, I_2, L, J \setminus L)$ la décomposition de $\{1, \ldots, n\}$ associée à l'itération i de l'algorithme de décodage de Stern et $I = I_1 \cup I_2$. A cette itération est associée une variable aléatoire X_i prenant ses valeurs dans l'espace des états*

$$\mathcal{E} = \{0, \ldots, 2p-1\} \cup \{(2p)_F, (2p)_S\} \cup \{2p+1, \ldots, w\}$$

de la manière suivante

$$\begin{aligned}
X_i = v \quad & ssi \quad |I \cap supp(e)| = v, \ \ \forall\, v \neq 2p \\
X_i = (2p)_F \quad & ssi \quad |I \cap supp(e)| = 2p \ et \ (|I_1 \cap supp(e)| \neq p \ ou \ |L \cap supp(e)| \neq 0) \\
X_i = (2p)_S \quad & ssi \quad |I \cap supp(e)| = 2p \ et \ |I_1 \cap supp(e)| = p \ et \ |L \cap supp(e)| = 0
\end{aligned}$$

L'ensemble des états de succès est donc

$$\mathcal{S} = \{(2p)_s\}$$

et celui des états d'échec

$$\mathcal{F} = \{0, \ldots, 2p-1\} \cup \{(2p)_F\} \cup \{2p+1, \ldots, w\}$$

Lorsque ces algorithmes sont utilisés pour rechercher un mot de poids minimal w, le processus stochastique reste inchangé pour l'algorithme de Stern. Par contre, pour l'algorithme DCP, il est légèrement modifié puisque, contrairement aux motifs d'erreurs, les mots de poids minimal détecté par l'algorithme possèdent au moins un bit 1 dans l'ensemble d'information I (cf. figure 3.8).

Définition 4.6 (Processus stochastique associé à l'algorithme de recherche de mots de poids minimal DCP) *Soit $(I, L, J \setminus L)$ la décomposition de $\{1, \ldots, n\}$ associée à l'itération i de l'algorithme de recherche de mots de poids minimal DCP. A cette itération est associée une variable aléatoire X_i prenant ses valeurs dans l'espace des états*

$$\mathcal{E} = \{0\} \cup \{1_F, \ldots, p_F\} \cup \{1_S, \ldots, p_S\} \cup \{p+1, \ldots, w\}$$

de la manière suivante

$$\begin{aligned} X_i = u_F \quad & ssi \quad |I \cap supp(e)| = u \ et \ |L \cap supp(e)| > s - u, \ \ \forall\, 1 \leq u \leq p \\ X_i = u_S \quad & ssi \quad |I \cap supp(e)| = u \ et \ |L \cap supp(e)| \leq s - u, \ \ \forall\, 1 \leq u \leq p \\ X_i = v \quad & ssi \quad |I \cap supp(e)| = v, \ \ \forall\, p+1 \leq v \leq w \ ou \ v = 0 \end{aligned}$$

L'ensemble des états de succès est donc

$$\mathcal{S} = \{1_S, \ldots, p_S\}$$

et celui des états d'échec

$$\mathcal{F} = \{0\} \cup \{1_F, \ldots, p_F\} \cup \{p+1, \ldots, w\}$$

Proposition 4.7 *Le processus stochastique $\{X_i\}_{i \in \mathbb{N}}$ associé à chacun des algorithmes itératifs est une chaîne de Markov homogène.*

preuve : Les sélections I, L — et I_1 et I_2 pour l'algorithme de Stern — correspondant à l'itération i ne dépendent que de l'ensemble d'information précédent, puisque L, I_1 et I_2 sont choisis aléatoirement. On a donc, pour toute itération i et pour tout ensemble d'états $(u_0, \ldots, u_i)$

$$Pr[X_i = u_i / X_{i-1} = u_{i-1}, \ X_{i-2} = u_{i-2}, \cdots, X_0 = u_0] = Pr[X_i = u_i / X_{i-1} = u_{i-1}]$$

De plus, cette probabilité ne dépend pas de l'itération. Il existe donc une matrice P telle que

$$\forall i \in \mathbb{N}, \ \forall (u, v) \in \mathcal{E}^2, \ \ Pr[X_i = v / X_{i-1} = u] = P_{u,v}$$

□

Le processus markovien $\{X_i\}_{i \in \mathbb{N}}$ est donc entièrement déterminé par sa matrice de transition P et sa distribution initiale π_0. Ces deux quantités peuvent

être aisément déterminées pour chacun des algorithmes puisque deux ensembles d'information successifs ne varient que d'un élément.

Proposition 4.8 (Matrice de transition du processus markovien associé à l'algorithme de décodage DCP) *La matrice de transition P à $(w+p+1)$ lignes et $(w+p+1)$ colonnes représentant l'algorithme de décodage DCP est donnée par*

$$\begin{aligned}
P_{u,u} &= \frac{k-u}{k} \times \frac{n-k-(w-u)}{n-k} + \frac{u}{k} \times \frac{w-u}{n-k} \ si\ u > p \\
P_{u,u-1} &= \frac{u}{k} \times \frac{n-k-(w-u)}{n-k} \ si\ u > p+1 \\
P_{u,u+1} &= \frac{k-u}{k} \times \frac{w-u}{n-k} \ si\ u > p-1 \\
P_{(u)_F,(u)_F} &= (1-\beta_u)\left[\frac{k-u}{k} \times \frac{n-k-(w-u)}{n-k} + \frac{u}{k} \times \frac{w-u}{n-k}\right] \ si\ 0 \le u \le p \\
P_{(u)_F,(u)_S} &= \beta_u\left[\frac{k-u}{k} \times \frac{n-k-(w-u)}{n-k} + \frac{u}{k} \times \frac{w-u}{n-k}\right] \ si\ 0 \le u \le p \\
P_{(u)_F,(u-1)_F} &= (1-\beta_{u-1})\left[\frac{u}{k} \times \frac{n-k-(w-u)}{n-k}\right] \ si\ 1 \le u \le p+1 \\
P_{(u)_F,(u-1)_S} &= \beta_{u-1}\left[\frac{u}{k} \times \frac{n-k-(w-u)}{n-k}\right] \ si\ 1 \le u \le p+1 \\
P_{(u)_F,(u+1)_F} &= (1-\beta_{u+1})\left[\frac{k-u}{k} \times \frac{w-u}{n-k}\right] \ si\ 0 \le u \le p-1 \\
P_{(u)_F,(u+1)_S} &= \beta_{u+1}\left[\frac{k-u}{k} \times \frac{w-u}{n-k}\right] \ si\ 0 \le u \le p-1 \\
P_{(p)_F,p+1} &= (1-\beta_{p+1})\left[\frac{k-p}{k} \times \frac{w-p}{n-k}\right] \\
P_{(u)_S,(u)_S} &= 1 \ si\ 0 \le u \le p \\
P_{u,v} &= 0 \ sinon
\end{aligned}$$

$$avec\ \beta_u = Pr[X_i = (u)_S \ / \ |I \cap supp(e)| = u] = \sum_{j=0}^{s-u} \frac{\binom{w-u}{j}\binom{n-k-w+u}{\sigma-j}}{\binom{n-k}{\sigma}}$$

La distribution initiale de probabilités π_0 est

$$\begin{aligned}
\pi_0(u_F) &= (1-\beta_u)\frac{\binom{n-w}{k-u}\binom{w}{u}}{\binom{n}{k}} \ si\ 0 \le u \le p \\
\pi_0(u_S) &= \beta_u\frac{\binom{n-w}{k-u}\binom{w}{u}}{\binom{n}{k}} \ si\ 0 \le u \le p \\
\pi_0(u) &= \frac{\binom{n-w}{k-u}\binom{w}{u}}{\binom{n}{k}} \ si\ p+1 \le u \le w
\end{aligned}$$

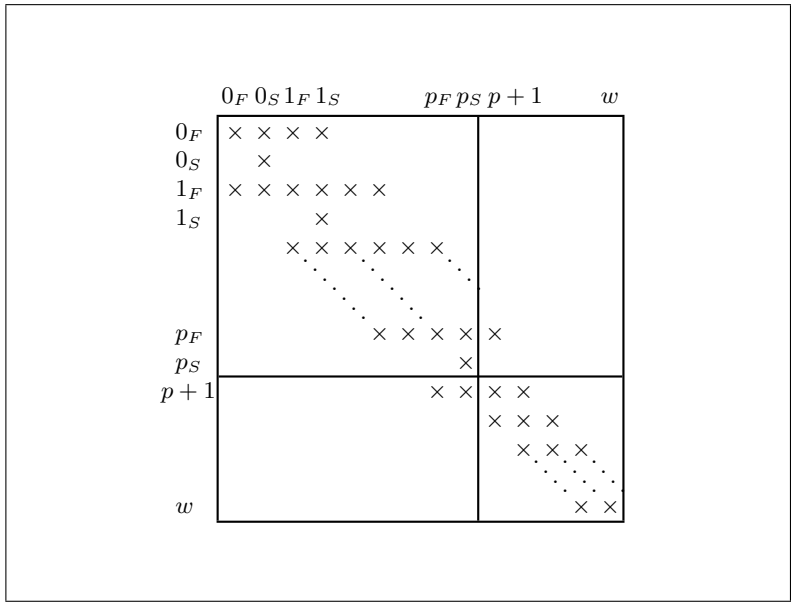


FIG. 4.2 –: *Forme de la matrice de transition du processus markovien associé à l'algorithme de décodage DCP*

Lorsque l'algorithme DCP est utilisé pour la recherche de mots de poids minimal, la matrice de transition est quasiment identique ; seul l'état 0 n'est plus dédoublé.

Proposition 4.9 (Matrice de transition du processus markovien associé à l'algorithme de Stern) *La matrice de transition P à $(w+2)$ lignes et $(w+2)$ colonnes représentant l'algorithme de décodage ou de recherche de mots de poids minimal de Stern est donnée par*

$$
\begin{aligned}
P_{u,u} &= \frac{k-u}{k} \times \frac{n-k-(w-u)}{n-k} + \frac{u}{k} \times \frac{w-u}{n-k} \ si\ u \notin \{(2p)_S, (2p)_F\} \\
P_{u,u-1} &= \frac{u}{k} \times \frac{n-k-(w-u)}{n-k} \ si\ u \neq 2p+1 \\
P_{u,u+1} &= \frac{k-u}{k} \times \frac{w-u}{n-k} \ si\ u \neq 2p-1 \\
P_{2p+1,(2p)_F} &= (1-\beta)\left[\frac{2p+1}{k} \times \frac{n-k-(w-(2p+1))}{n-k}\right] \\
P_{2p+1,(2p)_S} &= \beta\left[\frac{2p+1}{k} \times \frac{n-k-(w-(2p+1))}{n-k}\right] \\
P_{2p-1,(2p)_F} &= (1-\beta)\left[\frac{k-(2p-1)}{k} \times \frac{w-(2p-1)}{n-k}\right] \\
P_{2p-1,(2p)_S} &= \beta\left[\frac{k-(2p-1)}{k} \times \frac{w-(2p-1)}{n-k}\right]
\end{aligned}
$$

$$
\begin{aligned}
P_{(2p)_F,(2p)_F} &= (1-\beta)\left[\frac{k-2p}{k}\times\frac{n-k-(w-2p)}{n-k}+\frac{2p}{k}\times\frac{w-2p}{n-k}\right]\\
P_{(2p)_F,(2p)_S} &= \beta\left[\frac{k-2p}{k}\times\frac{n-k-(w-2p)}{n-k}+\frac{2p}{k}\times\frac{w-2p}{n-k}\right]\\
P_{(2p)_S,(2p)_S} &= 1\\
P_{u,v} &= 0 \textit{ sinon}
\end{aligned}
$$

$$
\textit{avec } \beta = Pr[X_i = (2p)_S \;/\; |I\cap supp(e)| = 2p] = \frac{\binom{2p}{p}\binom{k-2p}{k/2-p}}{\binom{k}{k/2}}\;\frac{\binom{n-k-w+2p}{\sigma}}{\binom{n-k}{\sigma}}
$$

La distribution initiale de probabilités π_0 est

$$
\begin{aligned}
\pi_0(u) &= \frac{\binom{w}{u}\binom{n-w}{k-u}}{\binom{n}{k}} \textit{ si } u\notin\{(2p)_F,(2p)_S\}\\
\pi_0((2p)_F) &= \frac{(1-\beta)\binom{w}{2p}\binom{n-w}{k-2p}}{\binom{n}{k}}\\
\pi_0((2p)_S) &= \frac{\beta\binom{w}{2p}\binom{n-w}{k-2p}}{\binom{n}{k}}
\end{aligned}
$$

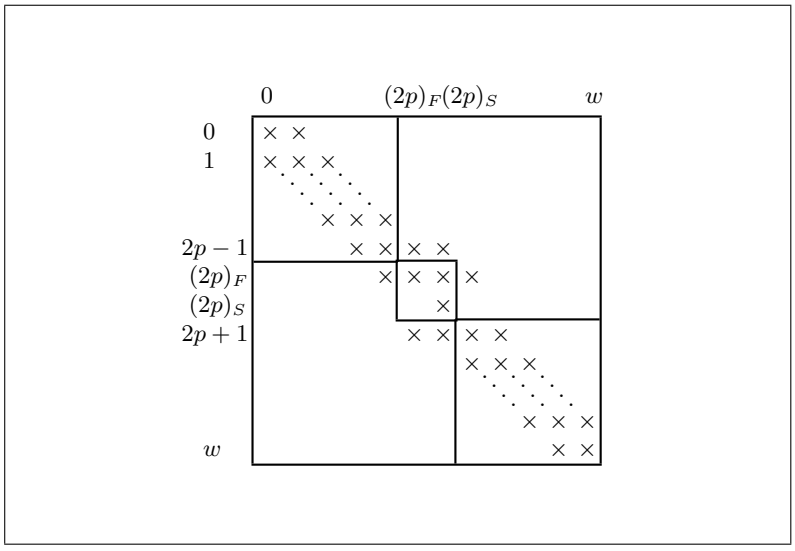


FIG. 4.3 –: *Forme de la matrice de transition du processus markovien associé à l'algorithme de Stern*

La chaîne de Markov associée à chacun des algorithmes itératifs est apériodique. L'ensemble $\mathcal{S}$ de ses états de succès correspond à l'ensemble de ses *états persistants*, c'est-à-dire que l'on ne peut pas quitter l'ensemble $\mathcal{S}$ une fois qu'on y est entré. De plus, chacun de ces états sont *absorbants*. La chaîne $\{X_i\}_{i\in\mathbb{N}}$ est donc une *chaîne de Markov absorbante*.

Une propriété classique des chaînes absorbantes ayant un nombre fini d'états est que, quelque soit l'état initial du processus, la probabilité pour qu'il soit dans un *état transient* — c'est-à-dire non persistant — après n itérations tend vers 0 quand n tend vers l'infini. Appliquée aux algorithmes itératifs, cette propriété induit le théorème suivant :

Théorème 4.10 (Convergence des algorithmes itératifs) *L'algorithme itératif DCP et l'algorithme itératif de Stern, utilisés pour le décodage ou pour la recherche de mots de poids minimal, convergent.*

2.2 Nombre moyen d'itérations des algorithmes itératifs

La propriété de chaîne absorbante permet également d'expliciter le nombre moyen d'itérations effectuées par chacun des algorithmes itératifs. On peut en effet associer à toute chaîne absorbante sa *matrice fondamentale*, définie par Kemeny et Snell [KS60].

Proposition 4.11 (Matrice fondamentale d'une chaîne de Markov absorbante) [KS60] *Soit $\{X_i\}_{i\in\mathbb{N}}$ une chaîne de Markov absorbante finie et P sa matrice de transition. Soit Q la restriction de P aux états transients de la chaîne. Alors la matrice $(Id - Q)$ est inversible et son inverse R, appelé* matrice fondamentale*, est donné par*

$$R = (Id - Q)^{-1} = \sum_{m=0}^{\infty} Q^m$$

De leur matrice fondamentale, on déduit aisément l'espérance du nombre d'itérations effectuées par les algorithmes itératifs.

Théorème 4.12 (Nombre moyen d'itérations des algorithmes itératifs) *Soit $\{X_i\}_{i\in\mathbb{N}}$ la chaîne de Markov associée à un algorithme itératif, π_0 sa distribution initiale de probabilités et R sa matrice fondamentale. L'espérance du nombre d'itérations N effectuée par cet algorithme est*

$$E(N) = \sum_{i\in\mathcal{F}} \pi_0(i) \sum_{j\in\mathcal{F}} R_{i,j}$$

où $\mathcal{F}$ est l'ensemble des états d'échec de la chaîne.

preuve : L'espérance de N est donné par

$$\begin{aligned} E(N) &= \sum_{n=0}^{\infty} nPr[X_n \in \mathcal{S} \text{ et } X_{n-1} \in \mathcal{F}] \\ &= \sum_{n=0}^{\infty} \sum_{m=0}^{n-1} Pr[X_n \in \mathcal{S} \text{ et } X_{n-1} \in \mathcal{F}] \end{aligned}$$

En utilisant le théorème de Fubini, on obtient

$$\begin{aligned} E(N) &= \sum_{m=0}^{\infty} \sum_{n=m+1}^{\infty} Pr[X_n \in \mathcal{S} \text{ et } X_{n-1} \in \mathcal{F}] \\ &= \sum_{m=0}^{\infty} Pr[X_m \in \mathcal{F}] \\ &= \sum_{m=0}^{\infty} \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{F}} Pr[X_m = j \ / \ X_0 = i] \\ &= \sum_{i \in \mathcal{F}} \pi_0(i) \sum_{j \in \mathcal{F}} \sum_{m=0}^{\infty} (Q^m)_{i,j} \\ &= \sum_{i \in \mathcal{F}} \pi_0(i) \sum_{j \in \mathcal{F}} R_{i,j} \end{aligned}$$

□

2.3 Variance du nombre d'itérations des algorithmes itératifs

La matrice fondamentale permet également de calculer la variance du nombre d'itérations, qui estime la dispersion du temps de calcul effectif de chaque algorithme par rapport à son facteur de travail moyen.

Théorème 4.13 (Variance du nombre d'itérations des algorithmes itératifs) *Soit $\{X_i\}_{i \in \mathbb{N}}$ la chaîne de Markov associée à un algorithme itératif, π_0 sa distribution initiale de probabilités et R sa matrice fondamentale. La variance du nombre d'itérations N effectuée par cet algorithme est*

$$V(N) = \sum_{i \in \mathcal{F}} \pi_0(i) \sum_{j \in \mathcal{F}} (2R_{i,j} - \delta_{i,j}) E_j(N) \ - \left(\sum_{i \in \mathcal{F}} \pi_0(i) E_i(N) \right)^2$$

où $\mathcal{F}$ est l'ensemble des états d'échec de la chaîne, $\delta_{i,j}$ le symbole de Kronecker et $E_i(N)$ le nombre moyen d'itérations effectuées par l'algorithme lorsqu'il part de l'état i, c'est-à-dire

$$E_i(N) = \sum_{j \in \mathcal{F}} R_{i,j}$$

preuve : Notons $E_i(N^2)$ l'espérance de la variable aléatoire N^2 lorsque le processus est initialement dans l'état i. On a alors

$$\begin{aligned} E_i(N^2) &= \sum_{n=1}^{\infty} n^2 Pr[X_n \in \mathcal{S} \text{ et } X_{n-1} \in \mathcal{F} \ / \ X_0 = i] \\ &= \sum_{j \in \mathcal{S}} P_{i,j} + \sum_{j \in \mathcal{F}} \sum_{n=1}^{\infty} P_{i,j} (n+1)^2 [X_n \in \mathcal{S} \text{ et } X_{n-1} \in \mathcal{F} \ / \ X_0 = j] \\ &= \sum_{j \in \mathcal{S}} P_{i,j} + \sum_{j \in \mathcal{F}} \sum_{n=1}^{\infty} P_{i,j} E_j((N+1)^2) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{j\in\mathcal{S}} P_{i,j} + \sum_{j\in\mathcal{F}} \sum_{n=1}^{\infty} P_{i,j} \left(E_j(N^2) + 2E_j(N) + 1 \right) \\
&= 1 + \sum_{j\in\mathcal{F}} \sum_{n=1}^{\infty} P_{i,j} \left(E_j(N^2) + 2E_j(N) \right)
\end{aligned}
$$

Notons respectivement $\bar{x}$ et $\bar{n}$ les vecteurs $(E_i(N^2))_{i\in\mathcal{F}}$ et $(E_i(N))_{i\in\mathcal{F}}$. La dernière équation s'écrit sous forme matricielle

$$\bar{x} = \bar{1} + Q\bar{x} + 2Q\bar{n}$$

où Q est la restriction de la matrice de transition P aux états d'échec. On obtient par conséquent, en utilisant la définition de la matrice fondamentale

$$
\begin{aligned}
\bar{x} &= (Id - Q)^{-1}(2Q\bar{n} + \bar{1}) \\
&= 2RQ\bar{n} + \bar{n}
\end{aligned}
$$

Or $R = \sum_{n=0}^{\infty} Q^n$. Donc $RQ = \sum_{n=1}^{\infty} Q^n = R - Id$, et $\bar{x} = (2R - Id)\bar{n}$. De l'expression matricielle

$$E(N^2) = \pi_0 \left[(2R - Id)\bar{n} \right]$$

on déduit donc le résultat énoncé pour la variance de N. □

2.4 Distribution du nombre d'itérations des algorithmes itératifs

La moyenne et la variance du nombre d'itérations effectuées par les algorithmes itératifs sont les seuls moments que l'on puisse facilement déterminer. La variance permet, grâce à l'inégalité de Tchebychev, d'approximer la probabilité pour que le nombre d'itérations N soit supérieur à une valeur donnée. Il s'agit d'un renseignement important puisqu'il estime la probabilité de décoder un code ou de trouver un mot de poids minimal quand le temps de calcul dont on dispose est limité. Toutefois, l'approximation fournie par l'inégalité de Tchebychev est en général trop grossière pour être réellement exploitable. Lorsque la taille de la matrice de transition P de la chaîne de Markov est raisonnable, il est donc préférable de calculer directement cette probabilité en diagonalisant la matrice P.

Proposition 4.14 *Soit P la matrice de transition de la chaîne de Markov $\{X_i\}_{i\in\mathbb{N}}$ associée à un des algorithmes itératifs et ρ le cardinal de l'espace de ses états. Si $(\lambda_1, \ldots, \lambda_\rho)$ sont les valeurs propres de P et $(L_1, \ldots, L_\rho)$ est une base de vecteurs propres, alors la probabilité pour que l'algorithme ait résolu le problème du décodage — ou trouvé un mot de poids minimal — après N itérations est*

$$\sum_{i\in\mathcal{E}} \pi_0(i)(L^{-1}\Lambda^N L)_{i,j}$$

où $\Lambda^N = diag(\lambda_1^N, \ldots, \lambda_\rho^N)$ *et*

$$L = \begin{pmatrix} L_1 \\ L_2 \\ \vdots \\ L_\rho \end{pmatrix}$$

Il est donc possible, pour un problème donné, de déterminer de cette manière la distribution du nombre d'itérations effectuées par chacun des algorithmes. Ainsi, la figure 4.4 représente, par tranches de 500 itérations, le pourcentage de réussite du décodage d'un code binaire aléatoire de longueur 256 et de dimension 128 par l'algorithme de Stern itératif (version GD), utilisé avec les paramètres $p = 1$ et $\sigma = 7$. Le nombre moyen d'itérations, calculé à l'aide de la proposition 4.12, est 4139, et son écart-type vaut 4158. Il est important de constater que la médiane est par contre de 2500, ce qui signifie que l'effort de calcul nécessaire pour décoder un code [256,128] avec une probabilité 1/2 ne représente que 60 % du facteur de travail de l'algorithme.

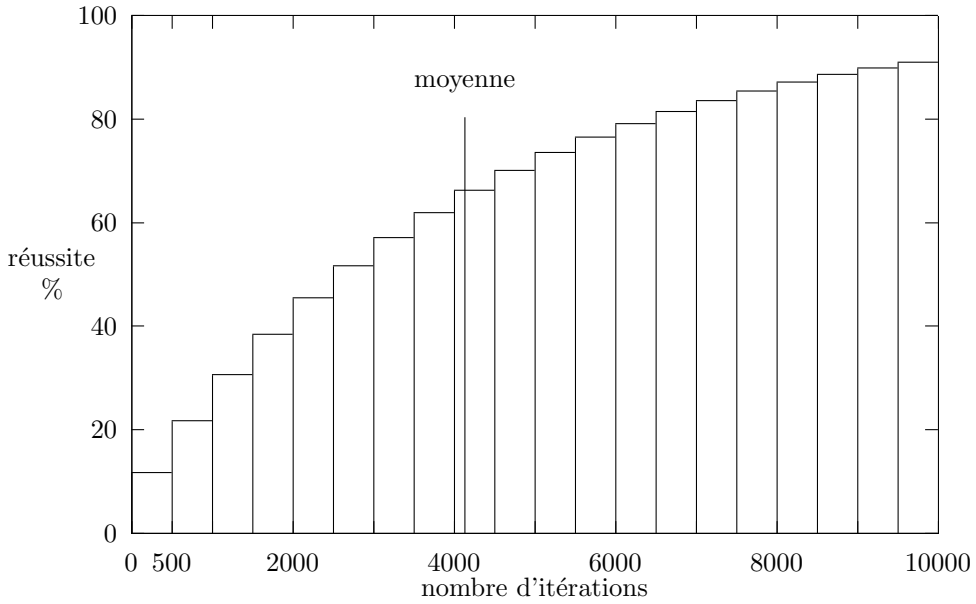


FIG. 4.4 –: *Taux théorique de réussite du décodage d'un code binaire aléatoire [256,128] jusqu'à sa capacité de correction en fonction du nombre d'itérations effectuées par l'algorithme de Stern itératif (version GD) pour* $p = 1$ *et* $\sigma = 7$

3 Facteur de travail des algorithmes itératifs

Le calcul du nombre moyen d'itérations des algorithmes itératifs grâce à leur modélisation par un processus markovien permet donc déterminer leur facteur de travail. L'estimation du nombre d'opérations binaires par itération est similaire à celle effectuée au chapitre précédent pour les algorithmes originaux : la seule modification concerne naturellement le coût de la mise sous forme systématique de la matrice génératrice puisque la procédure employée dans les algorithmes itératifs représente $k(n-k)/2$ opérations binaires au lieu des $k^2n/2$ nécessitées par l'élimination de Gauss.

Le facteur de travail des différentes versions de l'algorithme DCP et de celui de Stern dans le cas itératif se déduit donc des propositions 3.13 et 3.14 du chapitre 3.

Proposition 4.15 (Facteur de travail des différentes versions de l'algorithme DCP itératif) *Le facteur de travail de l'algorithme DCP itératif est :*

- *lorsqu'il est utilisé pour décoder un code $[n, nR]$ jusqu'à la distance w*

$$\begin{aligned} W_{DCP_{GD}(p,\sigma,s)} &= E(N_{DCP_D})\left[\frac{n^2R(1-R)}{2} + \frac{n(1-R)}{2} + J_{DCP_D(p,\sigma,s)}\right] \\ W_{DCP_{PD}(p,\sigma,s)} &= E(N_{DCP_D})\left[\frac{n^2R(1-R)}{2} + n(1-R)\left(\frac{1}{2}+nR\right)\right. \\ &+ \left. J_{DCP_D(p,\sigma,s)}\right] \end{aligned}$$

$$\text{avec } J_{DCP_D(p,\sigma,s)} = \sum_{i=0}^{p}(i+1)\binom{k}{i}\left(\sigma + (n(1-R)-\sigma)\frac{1}{2^\sigma}\sum_{j=0}^{s-i}\binom{\sigma}{j}\right)$$

  et où $E(N_{DCP_D})$ est le nombre moyen d'itérations, obtenu à l'aide la proposition 4.12 appliquée à la matrice de transition décrite à la proposition 4.8.

- *lorsqu'il est utilisé pour rechercher un mot de poids minimal w dans un code $[n, nR]$*

$$\begin{aligned} W_{DCP_{GM}(p,\sigma,s)} &= E(N_{DCP_M})\left[\frac{n^2R(1-R)}{2} + J_{DCP_M(p,\sigma,s)}\right] \\ W_{DCP_{PM}(p,\sigma,s)} &= E(N_{DCP_M})\left[\frac{n^2R(1-R)}{2} + n^2R(1-R) + J_{DCP_M(p,\sigma,s)}\right] \end{aligned}$$

$$\text{avec } J_{DCP_M(p,\sigma,s)} = \sum_{i=1}^{p} i\binom{k}{i}\left(\sigma + (n(1-R)-\sigma)\frac{1}{2^\sigma}\sum_{j=0}^{s-i}\binom{\sigma}{j}\right)$$

et où $E(N_{DCP_M})$ est le nombre moyen d'itérations, obtenu à l'aide la proposition 4.12 appliquée à la matrice de transition de la chaîne de Markov du processus décrit à la proposition 4.6.

Proposition 4.16 (Facteur de travail des différentes versions de l'algorithme de Stern itératif) *Le facteur de travail de l'algorithme de Stern itératif est :*

– *lorsqu'il est utilisé pour décoder un code $[n, nR]$ jusqu'à la distance w*

$$\begin{aligned} W_{S_{GD}(p,\sigma,s)} &= E(N_S)\left[\frac{n^2R(1-R)}{2} + \frac{n(1-R)}{2} + J_{S_D(p,\sigma,s)}\right] \\ W_{S_{PD}(p,\sigma,s)} &= E(N_S)\left[\frac{n^2R(1-R)}{2} + n(1-R)\left(\frac{1}{2}+nR\right)\right. \\ &+ \left. J_{S_D(p,\sigma,s)}\right] \end{aligned}$$

$$\begin{aligned} \text{avec } J_{S_D(p,\sigma,s)} &= (2p+1)\sigma\binom{nR/2}{p} + (2p+1)(n(1-R)-\sigma)\frac{\binom{nR/2}{p}^2}{2^\sigma} \\ &+ K\left(\binom{k/2}{p} + 2^\sigma\right) \end{aligned}$$

et où $E(N_S)$ est le nombre moyen d'itérations, obtenu à l'aide la proposition 4.12 appliquée à la matrice de transition décrite à la proposition 4.9.

– *lorsqu'il est utilisé pour rechercher un mot de poids minimal w dans un code $[n, nR]$*

$$\begin{aligned} W_{S_{GM}(p,\sigma,s)} &= E(N_S)\left[\frac{n^2R(R-1)}{2} + J_{S_M(p,\sigma,s)}\right] \\ W_{S_{PM}(p,\sigma,s)} &= E(N_S)\left[\frac{n^2R(1-R)}{2} + n^2R(1-R) + J_{S_M(p,\sigma,s)}\right] \end{aligned}$$

$$\text{avec } J_{S_M(p,\sigma,s)} = 2p\sigma\binom{nR/2}{p} + 2p(n(1-R)-\sigma)\frac{\binom{nR/2}{p}^2}{2^\sigma} + K\left(\binom{k/2}{p} + 2^\sigma\right)$$

L'expression du facteur de travail de chacun de ces algorithmes permet donc d'en optimiser les paramètres pour une taille de code donnée.

Cette optimisation est essentielle car, si l'on utilise ces algorithmes avec les mêmes paramètres pour toutes les tailles de code, leur complexité s'avère rapidement catastrophique. Ainsi, la figure 4.5 représente, pour le décodage avec l'algorithme de Stern itératif, la différence entre le facteur de travail obtenu avec

des paramètres fixes et avec les paramètres optimaux, lorsque la longueur du code $\mathcal{C}$ varie. Si l'on utilise par exemple cet algorithme avec les paramètres $p = 1$ et $\sigma = 5$, on minimise le coût moyen du décodage d'un code [128,64] mais on multiplie par contre par un facteur 13,5 le coût minimal du décodage d'un code [1024,512].

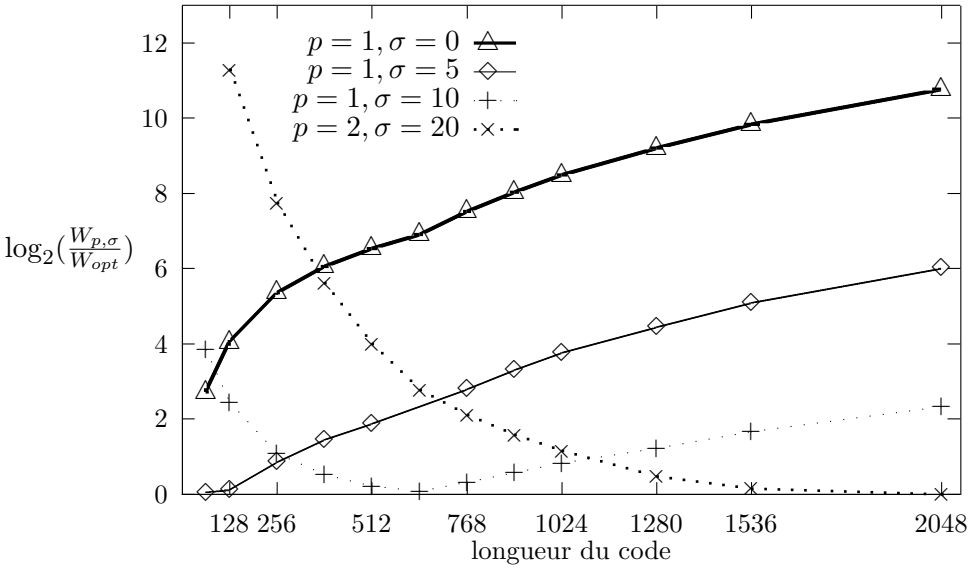


FIG. 4.5 –: *Influence des paramètres p et σ sur le facteur de travail de l'algorithme de Stern (version GM) utilisé pour décoder un code binaire aléatoire de longueur n et de dimension $n/2$*

4 Résultats expérimentaux

Afin de m'assurer de la pertinence de l'analyse théorique de complexité que je viens de mener, j'ai effectué un grand nombre de simulations pour résoudre un problème de petite taille : celui du décodage jusqu'à sa capacité de correction d'un code binaire aléatoire de longueur 256 et de dimension 128 — $w = 14$.

Les résultats exposés à la table 4.4 concernent le temps d'exécution de l'algorithme de Stern, version GM, pour ce problème de décodage, sur une station de travail DEC alpha à 175 MHz. Pour chaque couple de paramètres (p, σ), j'ai calculé le nombre moyen d'itérations pour 1000 instances du problème.

On voit bien que, pour tous les jeux de paramètres testés, le nombre moyen d'itérations effectuées expérimentalement est très proche de celui qui est donné par la modélisation markovienne. De plus, les paramètres qui théoriquement minimisent le facteur de travail sont également optimaux dans la pratique, et ils permettent de décoder un code [256,128] en un temps moyen de 2 secondes.

paramètres	facteur de travail théorique $\log_2(W)$	moyenne théorique du nombre d'itérations	moyenne expérimentale du nombre d'itérations	écart %	temps CPU moyen (s)
$p = 1$, $\sigma = 5$	27.37	3961	4072	+2.80	2.76
$p = 1$, $\sigma = 6$	26.80	4045	3985	-1.48	2.11
$p = 1$, $\sigma = 7$	26.51	4139	4190	+1.23	2.07
$p = 1$, $\sigma = 8$	26.56	4244	4338	+2.21	2.13
$p = 1$, $\sigma = 9$	26.95	4362	4417	+1.26	2.90
$p = 2$, $\sigma = 10$	29.80	432	433	+0.35	13.84
$p = 2$, $\sigma = 11$	29.04	442	470	+6.51	13.00
$p = 2$, $\sigma = 12$	28.51	454	446	-1.76	12.13
$p = 2$, $\sigma = 13$	28.37	466	487	+4.51	17.70
$p = 2$, $\sigma = 14$	28.68	480	508	+5.83	29.40

TAB. 4.4 –: *Résultats expérimentaux obtenus pour le décodage d'un code de longueur 256 et de dimension 128 avec différents paramètres de l'algorithme de Stern itératif (version GM)*

Pour ces paramètres optimaux ($p = 1$ et $\sigma = 7$), j'ai également examiné la répartition du nombre d'itérations effectuées expérimentalement en traçant à la figure 4.6 un histogramme similaire à celui de la figure 4.4.

Le pourcentage de réussite après un nombre fixé d'itérations obtenu expérimentalement par l'algorithme de Stern itératif correspond alors exactement à celui que j'ai déterminé par la théorie des chaînes de Markov. Le nombre d'itérations moyen sur 1000 expériences est de 4158 alors que sa valeur théorique est de 4139, et l'écart-type vaut 4141 au lieu d'une valeur théorique de 4158.

5 Approximation explicite du facteur de travail de l'algorithme de Stern itératif

Les facteurs de travail exprimés aux proposition 4.15 et 4.16 permettent certes d'estimer très précisément le coût du décodage et de la recherche de mots de poids minimal une fois que les paramètres de l'algorithme sont optimisés. Toutefois, le calcul du nombre moyen d'itérations à l'aide de la matrice de transition de la chaîne de Markov associée à l'algorithme n'est pas très aisé, d'autant plus qu'il est nécessaire de l'effectuer pour un grand nombre de paramètres.

Il serait alors souhaitable de disposer d'une formule explicite évaluant le facteur de travail des algorithmes itératifs optimisés en fonction de la taille du code étudié. C'est pourquoi j'établis ici une approximation du facteur de travail de l'algorithme de Stern itératif (version GM) — qui s'avère généralement le plus rapide — pour deux problèmes classiques : le décodage d'un code binaire

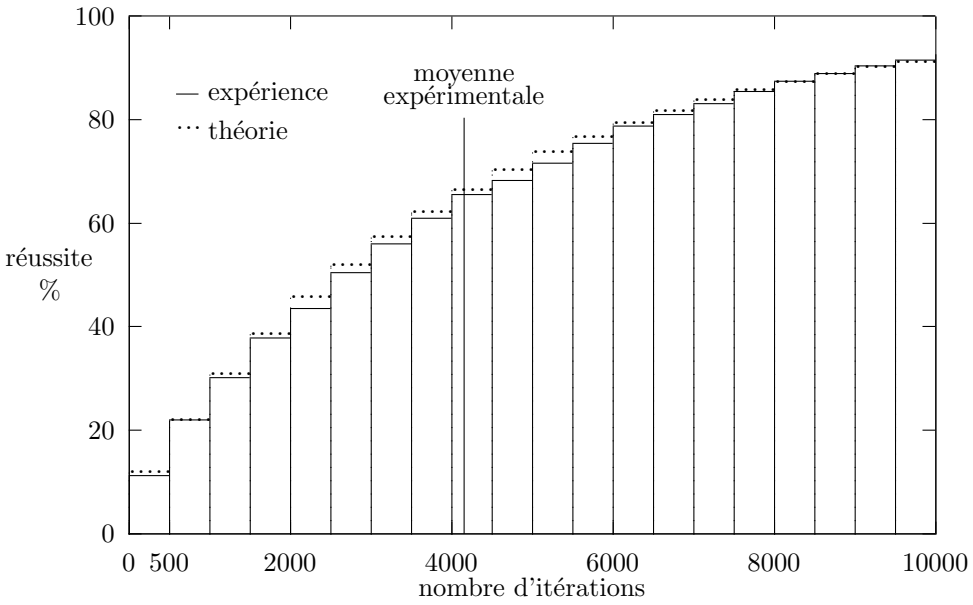


FIG. 4.6 –: *Taux expérimental de réussite du décodage d'un code binaire aléatoire [256,128] jusqu'à sa capacité de correction en fonction du nombre d'itérations effectuées par l'algorithme de Stern itératif (version GD) pour* $p = 1$ *et* $\sigma = 7$

code	[64,32,7]	[128,64,15]	[256,128,29]	[512,256,57]
t	3	7	14	28
paramètres optimaux	$p=1$ $\sigma=4$	$p=1$ $\sigma=6$	$p=1$ $\sigma=7$	$p=1$ $\sigma=9$
facteur de travail optimisé	$2^{15,39}$	$2^{19,36}$	$2^{26,51}$	$2^{40,48}$

[768,384,85]	[1024,512,113]	[1536,768,170]	[2048,1024,226]
42	56	84	112
$p=2$ $\sigma=17$	$p=2$ $\sigma=18$	$p=2$ $\sigma=19$	$p=2$ $\sigma=20$
$2^{54,55}$	$2^{68,51}$	$2^{96,87}$	$2^{125,50}$

TAB. 4.5 –: *Paramètres optimaux et facteur de travail de l'algorithme de Stern itératif (version GM) pour décoder des codes binaires aléatoires de longueur n et de dimension $n/2$*

aléatoire jusqu'à sa capacité de correction et la recherche de mots de poids minimal dans un tel code. Ces résultats permettent en particulier de se rendre compte immédiatement s'il est réaliste de tenter de décoder ou de trouver un mot de poids minimal dans un code donné.

5.1 Décodage d'un code aléatoire

Je m'intéresse ici au problème du décodage d'un code binaire aléatoire de longueur n et de dimension nR jusqu'à sa capacité de correction, estimée à l'aide de la borne de Gilbert-Varshamov.

La table 4.5 donne les paramètres optimaux de l'algorithme de Stern itératif, version GM, et le facteur de travail correspondant pour certaines longueurs de codes de taux d'expansion $R = 1/2$.

L'évolution du facteur de travail de l'algorithme de Stern itératif optimisé en fonction de la longueur n du code est représentée à la figure 4.7 pour différents taux d'expansion R. Il apparaît alors que, pour un taux d'expansion donné, le logarithme du facteur de travail optimisé dépend linéairement de n — contrairement aux cas où p et σ ne sont pas optimisés, comme l'a montré la figure 4.5. J'exprimerai donc ce facteur de travail sous la forme

$$W_{opt} \simeq 2^{n\alpha(R)+b}$$

De plus, le coefficient de complexité $\alpha(R)$, représenté à la figure 4.8 semble très proche de la fonction entropie $H_2(R)$ multipliée par une constante.

J'obtiens par conséquent l'approximation suivante :

Proposition 4.17 (Approximation du facteur de travail théorique de l'algorithme de Stern itératif pour le décodage) *Le facteur de travail*

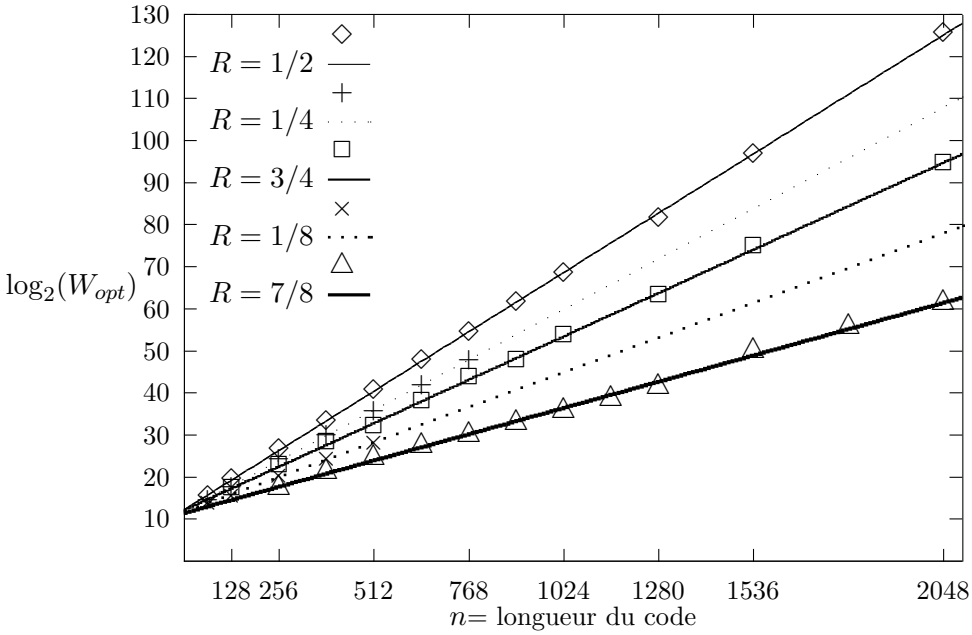


FIG. 4.7 –: *Evolution du facteur de travail de l'algorithme de Stern itératif optimisé pour le décodage de codes binaires aléatoires* $[n, nR]$

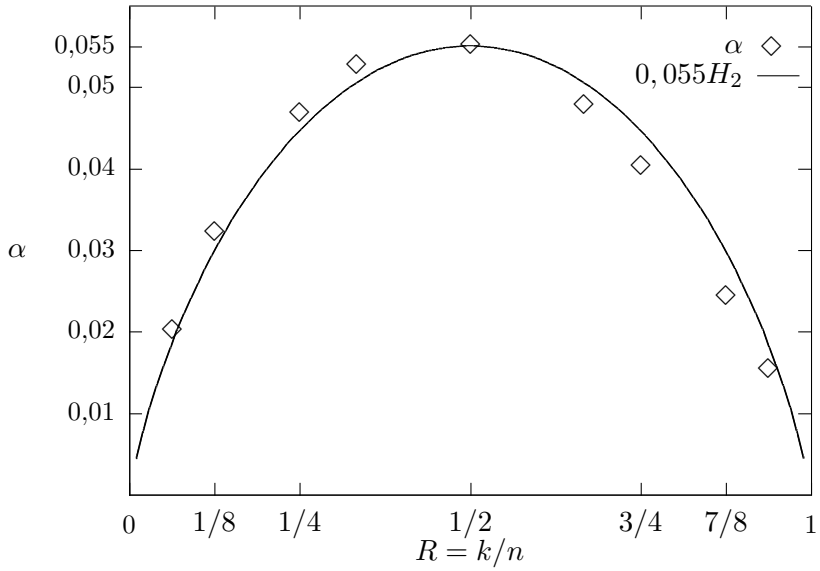


FIG. 4.8 –: *Evolution du coefficient de complexité de l'algorithme de Stern itératif pour le décodage de codes binaires aléatoires en fonction du taux d'expansion* R *du code*

code	[64,32,7]	[128,64,15]	[256,128,29]
paramètres optimaux	$p = 1$ $\sigma = 4$	$p = 1$ $\sigma = 5$	$p = 1$ $\sigma = 7$
facteur de travail optimisé	$2^{17,93}$	$2^{25,55}$	$2^{40,65}$

[384,192,43]	[512,256,57]	[640,320,71]
$p = 2$ $\sigma = 14$	$p = 2$ $\sigma = 15$	$p = 2$ $\sigma = 16$
$2^{55,77}$	$2^{70,72}$	$2^{85,78}$

TAB. 4.6 –: *Paramètres optimaux et facteur de travail de l'algorithme de Stern itératif (version GM) pour trouver un mot de poids minimal dans des codes binaires aléatoires de longueur n et de dimension $n/2$*

théorique requis par l'algorithme de Stern itératif (version GM) pour décoder un code binaire aléatoire de longueur n de dimension nR jusqu'à sa capacité de correction peut être estimé par la formule suivante

$$W_{opt} \simeq 2^{naH_2(R)+b} \text{ avec } a = 5,511.10^{-2} \text{ et } b = 12$$

5.2 Recherche de mots de poids minimal dans un code aléatoire

Je considère maintenant le problème de la recherche d'un mot de poids minimal dans un code binaire aléatoire. Je supposerai dans toute la suite que le code étudié ne contient qu'un seul mot de poids minimal. Je donne à la table 4.6 les paramètres optimaux de l'algorithme de Stern itératif (version GM) pour la recherche de mots de poids minimal dans des codes aléatoires de longueur n et de dimension $n/2$.

Comme pour le problème du décodage, le logarithme du facteur de travail de l'algorithme de Stern itératif optimisé semble dépendre linéairement de la longueur du code (cf. figure 4.9). Il peut donc également être écrit sous la forme

$$W_{opt} \simeq 2^{n\gamma(R)+d}$$

Le coefficient de complexité $\gamma(R)$ peut, cette fois, être approximé par la fonction entropie $H_2(R)$ multipliée par une constante et légèrement translatée, comme le montre la figure 4.10.

Le nombre d'opérations binaires nécessaires pour trouver un mot de poids minimal dans un code binaire suit donc l'approximation suivante :

Proposition 4.18 (Approximation du facteur de travail théorique de l'algorithme de Stern itératif pour la recherche d'un mot de poids minimal) *Le facteur de travail théorique requis par l'algorithme de Stern itératif (version GM) pour trouver un mot de poids minimal dans un code binaire*

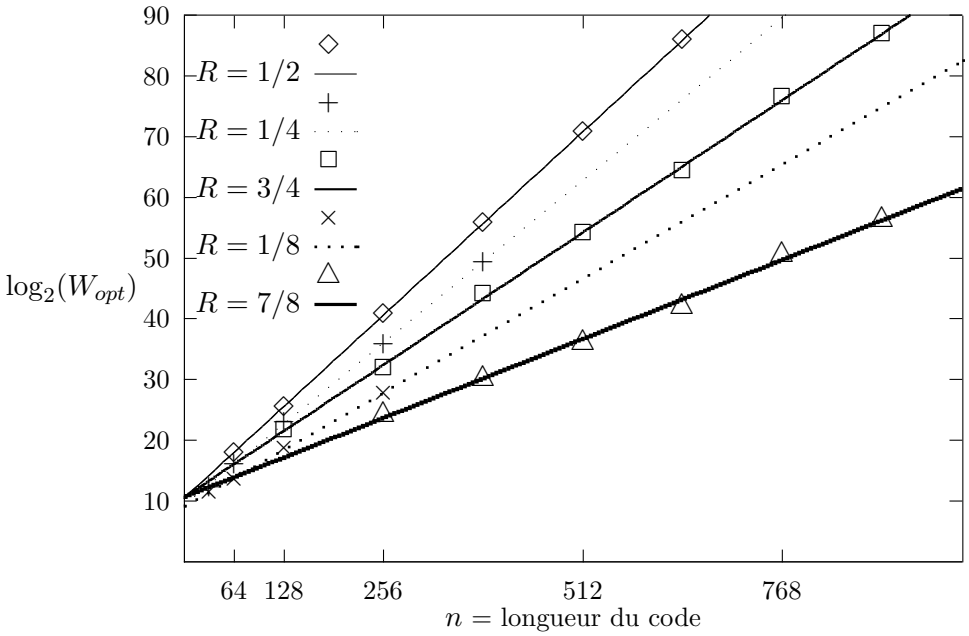


FIG. 4.9 –: *Evolution du facteur de travail de l'algorithme de Stern itératif optimisé pour la recherche d'un mot de poids minimal dans un code binaire aléatoire* $[n, nR]$

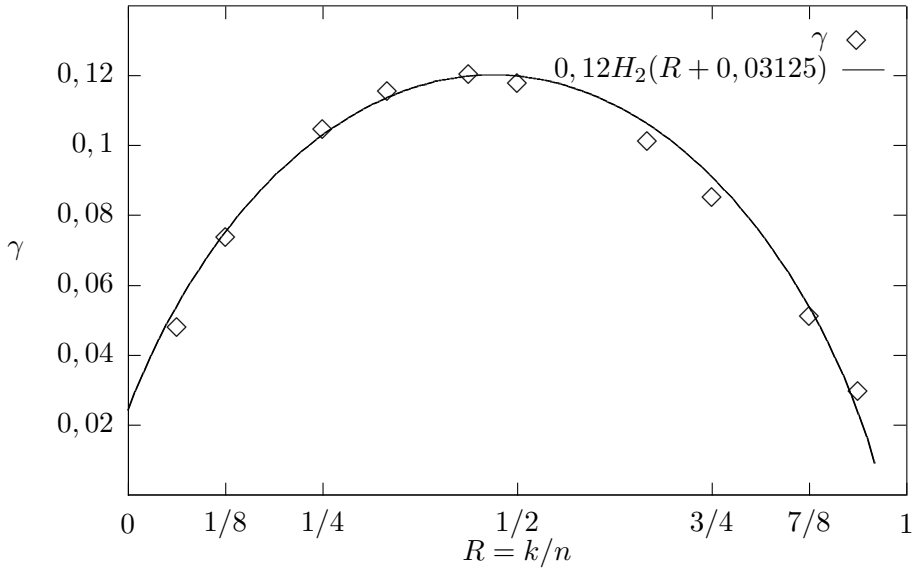


FIG. 4.10 –: *Evolution du coefficient de complexité de l'algorithme de Stern itératif pour la recherche d'un mot de poids minimal dans un code binaire aléatoire en fonction du taux d'expansion* R *du code*

aléatoire de longueur n de dimension nR peut être estimé par la formule suivante

$$W_{opt} \simeq 2^{ncH_2(R+R_0)+d} \text{ avec } c = 0,12, \quad d = 10 \text{ et } R_0 = 3,125.10^{-2}$$

6 Application à l'attaque des cryptosystèmes à mots de poids faible

Les algorithmes itératifs que je viens de présenter constituent actuellement l'attaque la plus performante des différents cryptosystèmes à mots de poids faible. Le calcul précis de leur facteur de travail permet d'évaluer la sécurité de ces systèmes et de déterminer quels paramètres les rendent hors de portée de la cryptanalyse.

6.1 Facteur de travail de l'attaque des cryptosystèmes à mots de poids faible par les algorithmes itératifs

Les tables 4.8 et 4.9 donnent les paramètres optimaux et le nombre d'opérations binaires requis pour attaquer les cryptosystèmes à mots de poids faible à l'aide des versions itératives de l'algorithme DCP et de celui de Stern.

La cryptanalyse du système de McEliece avec ses paramètres originaux nécessite donc $2^{64,2}$ opérations binaires ; cette nouvelle attaque est certes encore irréalisable avec la puissance de calcul dont on dispose actuellement, mais elle est 128 fois plus rapide que celle de Lee et Brickell, qui était généralement considérée comme la plus performante.

Ces résultats montrent également qu'il n'est pas envisageable de réduire la taille des clefs publiques utilisées dans le système de McEliece en employant des codes de longueur inférieure à 1024. Ainsi, le décodage d'un code de Goppa de longueur 512 et de dimension 260 jusqu'à sa capacité de correction ($w = 28$) ne nécessite que $2^{40,1}$ opérations binaires (avec l'algorithme de Stern itératif. version GM, $p = 1$, $\sigma = 9$).

Toutefois, l'écart-type du nombre d'itérations pour ces algorithmes est toujours du même ordre que sa moyenne (cf. table 4.7). Cette grande dispersion du nombre d'itérations effectuées par l'algorithme implique en conséquence qu'un facteur de travail moyen inaccessible ne suffit à garantir la solidité des cryptosystèmes à mots de poids faible : en effet l'algorithme pourrait, en un temps raisonnable, déchiffrer une proportion non négligeable de messages ou retrouver certaines clefs secrètes des schémas d'identification. Il apparaît donc nécessaire d'étudier plus précisément la distribution du nombre d'itérations de l'algorithme itératif suivant la méthode développée à la section 2.4.

6.1.1 Systèmes de chiffrement de McEliece et de Niederreiter

Tout comme pour le problème du décodage d'un code de longueur 256 et de dimension 128 étudié théoriquement et expérimentalement à la figure 4.6, le

cryptosystèmes	McEliece	Stern	Véron
code utilisé	[1024,524]	[512,256]	[512,120]
distance de décodage w	50	56	114
meilleur algorithme	Stern GM $p=2$ $\sigma=18$ $2^{64,2}$	Stern GD $p=2$ $\sigma=15$ $2^{69,8}$	Stern GD $p=2$ $\sigma=13$ $2^{60,9}$
nb moyen d'itérations	$9,85.10^{11}$	$2,16.10^{14}$	$1,74.10^{12}$
écart-type	$9,85.10^{11}$	$2,16.10^{14}$	$1,74.10^{12}$

TAB. 4.7 –: *Moyenne et écart-type du nombre d'itérations nécessaires à l'attaque des cryptosystèmes à mots de poids faible*

facteur de travail nécessaire pour décoder une fois sur deux avec succès un code de longueur 1024 et de dimension 524 jusqu'à la distance 50 — et donc pour cryptanalyser le système de McEliece avec une probabilité 1/2 — ne représente que 69 % du facteur de travail moyen. La figure 4.11 montre aussi que le nombre moyen d'itérations correspond en fait à un taux de réussite de plus de 60 %.

J'ai donc été amenée à conjecturer qu'un nombre d'itérations raisonnable pouvaient suffire à cryptanalyser une proportion non négligeable des messages chiffrés par le système de McEliece. Il apparaît par exemple à la figure 4.12 que, s'il se limite à 2^{51} opérations binaires, c'est-à-dire à 10^8 itérations, l'algorithme de Stern itératif avec ses paramètres optimaux déchiffre un message sur 10000. Sachant que 1000 itérations de l'algorithme prennent de l'ordre de 10 minutes sur une station de travail DEC alpha 500/266, il suffit par exemple de 2 mois et 25 jours pour déchiffrer un message sur 10000, lorsque l'on dispose de dix stations de ce type.

Il s'agit donc d'une faiblesse importante du système de chiffrement de McEliece utilisé avec ses paramètres originaux car la proportion de messages déchiffrés en un temps raisonnable est relativement élevée dès lors que l'attaquant dispose de quelques dizaines de stations de travail rapides.

6.1.2 Schémas d'identification de Stern et de Véron

Une étude similaire (figure 4.13) montre que les paramètres du schéma d'identification de Stern le rendent beaucoup plus solide cryptographiquement. En effet, onze mois de calculs sur dix stations DEC alpha 500/266 ne permettent de retrouver la clef secrète d'un utilisateur que dans un cas sur 100 000. Cela signifie malgré tout qu'il est nécessaire de changer les clefs secrètes relativement souvent : la durée de vie d'une clef ne doit raisonnablement pas excéder un an.

Les paramètres choisis par P. Véron ont l'avantage de réduire considérablement le nombre de bits transmis lors de chaque procédure d'identification ; toutefois ils imposent une durée de vie des clefs beaucoup plus faible. La figure 4.14 montre en effet que 56 jours sur dix stations DEC alpha 500/266 permettent de retrouver la clef d'un utilisateur avec une probabilité supérieure

cryptosystèmes	McEliece	Stern	Véron
code utilisé	[1024,524]	[512,256]	[512,120]
distance de décodage w	50	56	114
algorithme DCP, GD	$p=1$, $\sigma=11$, $s=2$, $2^{68,4}$	$p=1$, $\sigma=17$, $s=5$, $2^{72,3}$	$p=2$, $\sigma=12$, $s=4$, $2^{61,9}$
algorithme DCP, PD	$p=2$, $\sigma=8$, $s=2$, $2^{68,7}$	$p=2$, $\sigma=6$, $s=2$, $2^{72,7}$	$p=2$, $\sigma=12$, $s=4$, $2^{62,0}$
algorithme DCP, GM	$p=2$, $\sigma=8$, $s=2$, $2^{68,1}$	$p=2$, $\sigma=6$, $s=2$, $2^{72,4}$	$p=2$, $\sigma=12$, $s=4$, $2^{61,8}$
algorithme DCP, PM	$p=2$, $\sigma=8$, $s=2$, $2^{68,3}$	$p=2$, $\sigma=6$, $s=2$, $2^{72,5}$	$p=2$, $\sigma=12$, $s=4$, $2^{62,0}$
Meilleure version	GM, $p=2$, $\sigma=8$, $s=2$, $2^{68,1}$	GM, $p=2$, $\sigma=6$, $s=2$, $2^{72,4}$	GM, $p=2$, $\sigma=12$, $s=4$, $2^{61,8}$

TAB. 4.8 –: *Facteur de travail des différentes versions optimisées de l'algorithme DCP itératif pour cryptanalyser les systèmes à mots de poids faibles*

cryptosystèmes	McEliece	Stern	Véron
code utilisé	[1024,524]	[512,256]	[512,120]
distance de décodage w	50	56	114
algorithme de Stern, GD	$p=2$, $\sigma=18$, $2^{64,3}$	$p=2$, $\sigma=15$, $2^{69,8}$	$p=2$, $\sigma=13$, $2^{60,9}$
algorithme de Stern, PD	$p=2$, $\sigma=18$, $2^{64,3}$	$p=2$, $\sigma=15$, $2^{69,8}$	$p=2$, $\sigma=13$, $2^{61,0}$
algorithme de Stern, GM	$p=2$, $\sigma=18$, $2^{64,2}$	$p=2$, $\sigma=15$, $2^{69,9}$	$p=2$, $\sigma=13$, $2^{61,2}$
algorithme de Stern, PM	$p=2$, $\sigma=18$, $2^{64,2}$	$p=2$, $\sigma=15$, $2^{69,9}$	$p=2$, $\sigma=13$, $2^{61,2}$
Meilleure version	GM, $p=2$, $\sigma=18$, $2^{64,2}$	GD, $p=2$, $\sigma=15$, $2^{69,8}$	GD, $p=2$, $\sigma=13$, $2^{60,9}$

TAB. 4.9 –: *Facteur de travail des différentes versions optimisées de l'algorithme de Stern itératif pour cryptanalyser les systèmes à mots de poids faibles*

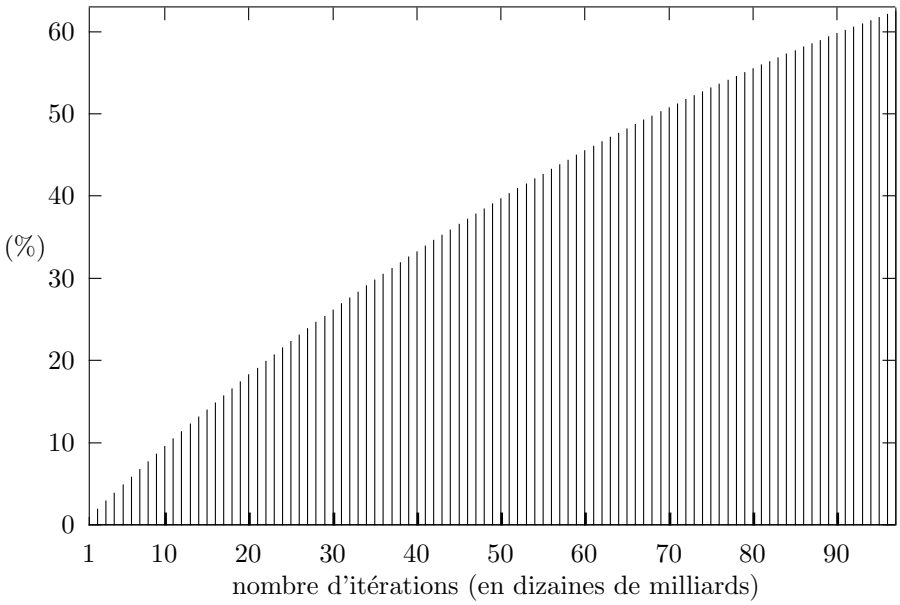


FIG. 4.11 –: *Taux de réussite de l'attaque du système de chiffrement de McEliece en fonction du nombre d'itérations effectuées par l'algorithme de Stern itératif optimisé*

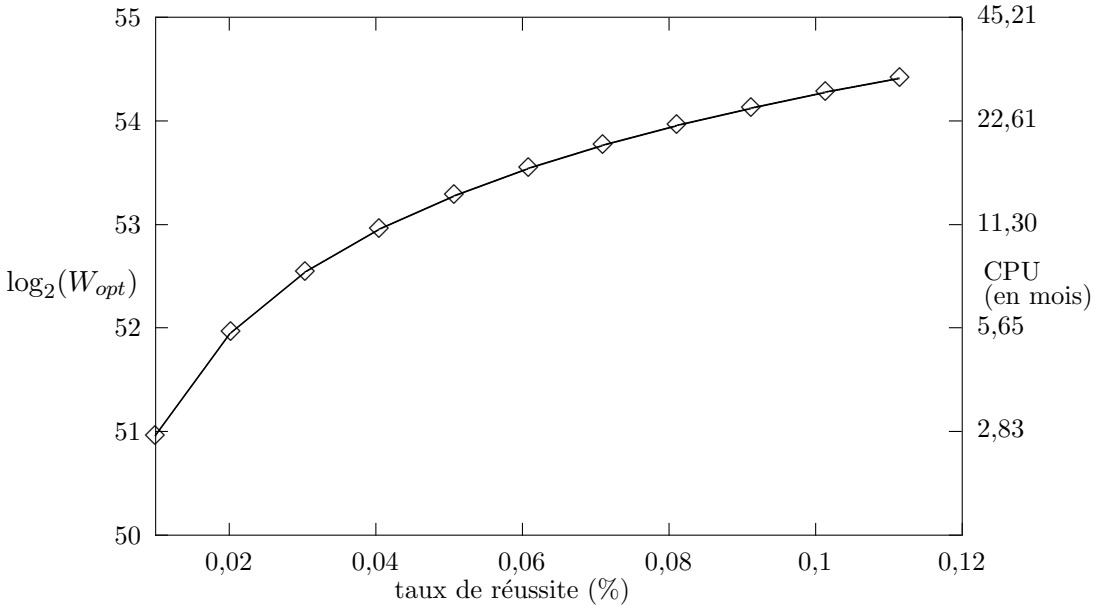


FIG. 4.12 –: *Effort de calcul nécessaire pour cryptanalyser le système de McEliece en fonction du taux de déchiffrement : le temps CPU, exprimé en mois, est donné pour 10 DEC alpha500/266 en parallèle*

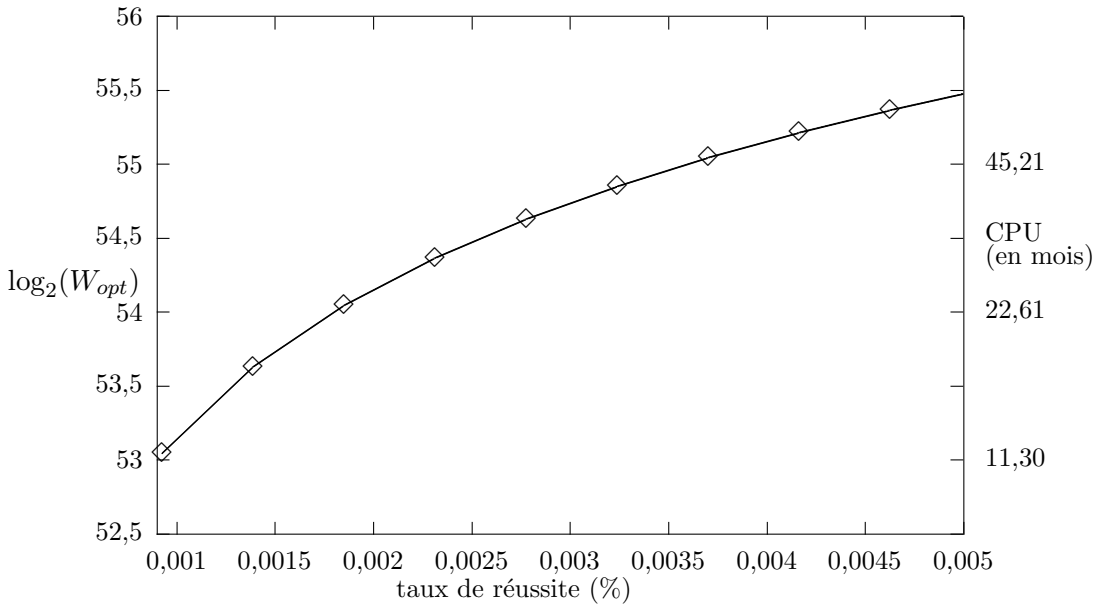


FIG. 4.13 –: *Effort de calcul nécessaire pour cryptanalyser le schéma d'identification de Stern en fonction du taux de réussite : le temps CPU, exprimé en mois, est donné pour 10 DEC alpha500/266 en parallèle*

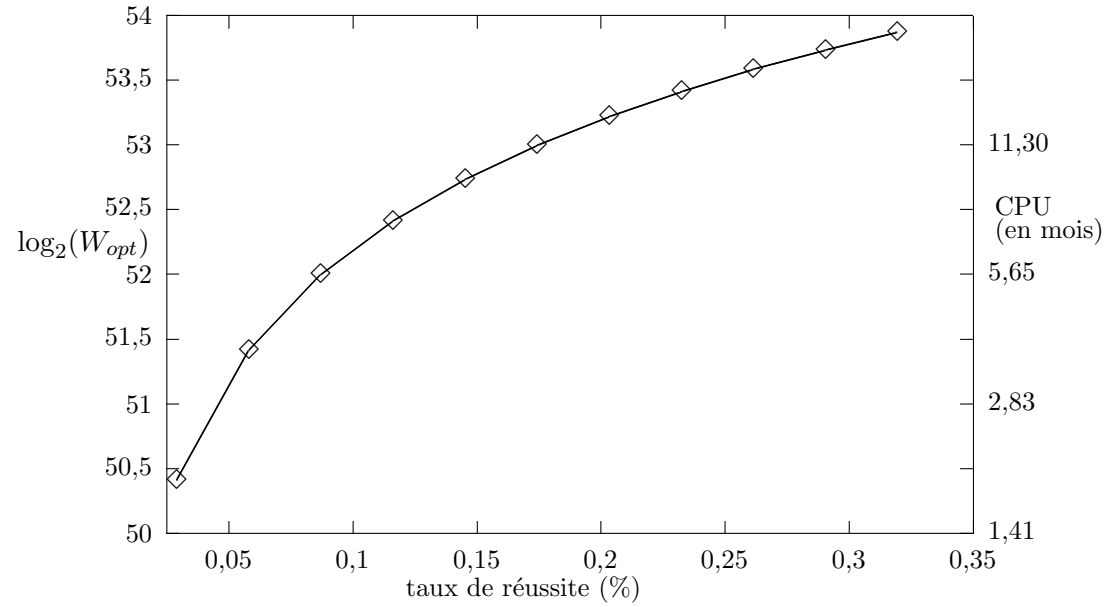


FIG. 4.14 –: *Effort de calcul nécessaire pour cryptanalyser le schéma d'identification de Véron en fonction du taux de réussite : le temps CPU, exprimé en mois, est donné pour 10 DEC alpha500/266 en parallèle*

à 1/3500.

6.2 Attaques partielles des cryptosystèmes à mots de poids faible

Les systèmes de chiffrement à mots de poids faible présentent par ailleurs d'autres faiblesses dans la mesure où la connaissance d'un petit nombre de bits du texte clair suffit à en retrouver l'intégralité.

6.2.1 Dimensions accessibles

Il est relativement raisonnable, lorsque l'on évalue la sécurité d'un système de chiffrement, de supposer que l'attaquant connaît une petite partie du texte clair : il est par exemple assez probable qu'un message débute par "Bonjour" ou se termine par une formule de politesse. Cette situation se produit également si les messages clairs suivent un format usuel, tels des transactions bancaires, des actes notariaux, des résultats médicaux ... Dans ce cas, seuls quelques bits du texte clair sont inconnus de l'attaquant — ceux qui spécifient par exemple le montant de la transaction. Il est alors très important que la connaissance d'une partie du texte clair ne permette pas de déterminer l'intégralité du message. Une attaque de ce type, récemment développée sur le système RSA par Coppersmith [Cop96], a ainsi conduit à proscrire l'usage de 3 comme exposant public.

Pour le système de McEliece, la connaissance de certains bits du texte clair permet de réduire d'autant la dimension du code considéré lors de la cryptanalyse. Le problème est donc de déterminer la dimension à partir de laquelle un code de longueur 1024 peut être décodé par les algorithmes itératifs.

Je considérerai comme il en est d'usage qu'un facteur de travail de 2^{50} opérations binaires correspond à un effort de calcul réaliste. En utilisant l'algorithme de Stern itératif (version GM), il est alors possible de décoder jusqu'à la distance $w = 50$ un code linéaire de longueur 1024 et de dimension 404 puisque le facteur de travail correspondant est de $2^{50,1}$ avec les paramètres $p = 1$ et $\sigma = 10$. Cela signifie donc que la connaissance de 120 bits, c'est-à-dire 22,9 %, du texte clair suffit à retrouver le message tout entier en un temps raisonnable.

6.2.2 Poids accessibles

Une attaque similaire du système de Niederreiter revient à supposer que certaines positions d'erreurs sont connues. Il s'agit donc ici de déterminer jusqu'à quelle distance un code [1024,524] peut être décodé en un temps réaliste.

La table 4.10 montre alors que la connaissance de 15 positions d'erreurs sur les 50 introduites dans les systèmes de McEliece et de Niederreiter suffit à déchiffrer un message. Cette proportion relativement faible implique entre autres qu'il peut s'avérer dangereux par exemple d'utiliser un canal bruité pour générer le vecteur d'erreurs puisque cette méthode pourrait parfois produire des erreurs de poids inférieur à 50.

Code	[1024,524]	[512,260]
capacité de correction	50	28
distance de décodage accessible	35	37
facteur de travail	$p = 2$ $\sigma = 18$ $2^{49,9}$	$p = 2$ $\sigma = 15$ $2^{50,1}$

TAB. 4.10 –: *Distance de décodage accessible par l'algorithme de Stern itératif (version GM)*

Pour les schémas d'identification, je donne de manière similaire les valeurs minimale et maximale admissibles pour le poids de la clef secrète d'un utilisateur. Pour les poids w supérieurs à la distance minimale du code, le facteur de travail de l'algorithme doit évidemment être divisé par le nombre de mots de poids w dans le code, estimé pour un code binaire aléatoire $[n, k]$ à $\frac{\binom{n}{w}}{2^{n-k}}$

Code	[512,256]	[512,120]
distance minimale	57	115
premier poids inaccessible	37	91
facteur de travail	$p = 2$ $\sigma = 15$ $2^{49,1}$	$p = 2$ $\sigma = 13$ $2^{49,9}$
dernier poids inaccessible	72	128
facteur de travail	$p = 2$ $\sigma = 13$ $2^{49,5}$	$p = 2$ $\sigma = 13$ $2^{49,9}$

TAB. 4.11 –: *Poids accessibles par l'algorithme de Stern itératif (version GM)*

6.3 Optimisation des paramètres du système de McEliece

Adams et Meijer [AM87] avaient déjà remarqué que, pour des codes de Goppa de longueur 1024, la dimension 524 choisie par McEliece n'était pas celle qui maximisait le facteur de travail de l'attaque par l'algorithme de décodage par ensembles d'information, et qu'il était préférable d'utiliser la famille des codes de Goppa irréductibles de longueur 1024, de dimension 654 et de capacité de correction 37.

La nouvelle attaque que je viens de développer me conduit de la même façon à affiner les paramètres du système de McEliece et à préférer aux paramètres originaux ceux qui maximisent le coût de l'algorithme itératif le plus performant, c'est-à-dire

$$n = 1024,\ k = 614,\ w = 41$$

Le facteur de travail correspondant à l'algorithme de Stern optimisé (version GM, $p = 2$, $\sigma = 18$) est alors de $2^{66,0}$ opérations binaires.

En revanche, comme le montre l'étude menée à la section 5.2, le coût de la recherche d'un mot de poids minimal dans un code binaire aléatoire est

maximal lorsque le taux d'expansion du code vaut 1/2. Pour des codes de longueur 512, les paramètres choisis par Stern sont donc optimaux dans ce sens. Ceux du schéma de Véron ne le sont évidemment pas puisqu'il s'agit, parmi les paramètres résistant à la cryptanalyse, de ceux qui minimisent le nombre de bits échangés lors d'une transaction.

Chapitre 5

Distance minimale des codes BCH de longueur 511

Les algorithmes itératifs décrits au chapitre précédent sont actuellement les plus performants pour attaquer les cryptosystèmes à mots de poids faible, mais aussi, plus généralement, pour trouver un mot de poids minimal dans un code linéaire. Ils peuvent donc apporter de précieuses informations sur la distance minimale de certains codes classiques lorsqu'aucun résultat théorique n'est parvenu à la déterminer et que la taille des codes rend impossible toute recherche exhaustive.

Ainsi les codes BCH sont une famille de codes cycliques pour laquelle seule une borne inférieure sur la distance minimale est connue. Ces codes ont été étudiés intensivement depuis leur découverte en 1959 [Hoc59, BRC60], et différents auteurs sont parvenus à définir des conditions suffisantes pour que cette borne soit atteinte [PW61, KLP66, KL72]. Toutefois, aucune méthode générale ne permet actuellement de calculer leur distance minimale, même s'il apparaît qu'elle est très souvent égale à la borne BCH. Aussi, dans certains cas, seule un algorithme probabiliste de recherche de mots de poids minimal semble-t-il pouvoir fournir de plus amples renseignements. La détermination de la distance minimale de certains codes BCH primitifs constitue donc un défi que chaque nouvel algorithme de recherche de mots de poids faible tente de relever depuis près de 35 ans.

La table des paramètres des BCH de longueur 255, établie par MacWilliams et Sloane [MS77], fut finalement complétée par D. Augot, P. Charpin et N. Sendrier en 1992. Ces derniers entamèrent alors celle des BCH de longueur 511 et, depuis leurs travaux, la distance de douze de ces 57 codes restait à déterminer. Grâce à l'algorithme de Stern itératif, j'ai alors réussi à montrer que six d'entre eux atteignent la borne BCH. Ce résultat met en valeur les performances de ce nouvel algorithme puisque les diverses tentatives de recherche de mots de poids minimal avaient jusqu'à présent échoué pour ces codes et se cantonnaient à la longueur 255 (voir par exemple [BB95]).

1 Définitions

Je m'intéresse ici uniquement aux codes BCH primitifs au sens strict définis sur $\mathbb{F}_2$, qui sont ceux qui furent originellement introduits par Hocquenghem [Hoc59], Bose et Ray-Chaudhuri [BRC60].

Définition 5.1 (Codes BCH) *Un code cyclique binaire de longueur n et de polynôme générateur g est un* code BCH de distance construite δ *si δ est le plus grand entier tel qu'il existe un entier b pour lequel*

$$g(\alpha^b) = g(\alpha^{b+1}) = \ldots = g(\alpha^{b+\delta-2})$$

où α est un élément primitif de $\mathbb{F}_{2^m}$, m étant le plus petit entier tel que n divise $2^m - 1$. Lorsque l'entier b vaut 1, un tel code est code BCH au sens strict *et il est noté $B(n, \delta)$.*

Cette définition dépend du choix de l'élément primitif α mais la même construction pour un autre choix de α produit un code équivalent. On peut immédiatement remarquer que tout code BCH au sens strict de distance construite δ contient tous les mots des codes BCH de distance construite supérieure.

Je considère ici plus particulièrement les codes BCH binaires *primitifs*, c'est-à-dire ceux dont la longueur s'écrit $n = 2^m - 1$. La dimension d'un code BCH primitif de longueur $n = 2^m - 1$ est donnée par le degré de son polynôme générateur, $k = n - deg(g)$. Comme $g(x)$ divise $x^n - 1$, l'ensemble de définition de $B(n, \delta)$ est égal à la réunion des classes cyclotomiques modulo n, $Cl(1) \cup Cl(2) \cup \ldots \cup Cl(\delta - 1)$. Lorsque $\delta - 2 < 2^{\lceil m/2 \rceil}$, toutes les classes $Cl(i)$, pour i premier compris entre 1 et $\delta - 1$, sont distinctes et contiennent chacune m éléments [MS77, page 262]. Dans ce cas, $k = 2^m - 1 - m\frac{\delta-1}{2}$.

La construction de ces codes induit immédiatement une borne sur leur distance minimale, connue sous le nom de *borne BCH* [Hoc59, BRC60].

Proposition 5.2 (Borne BCH) *La distance minimale d d'un code BCH de distance construite δ vérifie*

$$d \geq \delta$$

Cette borne s'avère en fait très puissante puisqu'elle est la plupart du temps atteinte — le plus petit code BCH primitif binaire au sens strict dont la distance minimale est strictement supérieure à sa distance construite est $B(127, 29)$, et il s'agit du seul pour cette longueur. C'est pourquoi aucune des autres bornes générales sur les codes cycliques n'apporte de renseignements supplémentaires sur la valeur exacte de la distance minimale des codes BCH.

Ces codes étant très structurés, ils possèdent cependant de nombreuses propriétés. En particulier, MacWilliams a montré que le groupe d'automorphisme de l'étendu d'un code BCH binaire primitif contient le groupe affine, c'est-à-dire

l'ensemble des permutations de $\mathbb{F}_{2^m}$ de la forme $x \mapsto ax + b$, avec $a, b \in \mathbb{F}_{2^m}$, et $a \neq 0$. Ceci implique notamment la proposition suivante :

Proposition 5.3 *Soit $(a_0, \ldots, a_n)$ la distribution des poids d'un code BCH primitif binaire $B(n, \delta)$. Alors, pour tout j, on a*

$$(n + 1 - 2j)a_{2j-1} = 2ja_{2j}$$

preuve : Le groupe affine est transitif, c'est-à-dire que, pour tout couple d'éléments (i, j) de $\mathbb{F}_{2^m}$, il existe une permutation affine π telle que $\pi(i) = j$. Le tableau formé par les mots de poids $2j$ du code étendu $\hat{B}(n, \delta)$ est donc un tableau orthogonal de force 1 : toutes ses colonnes possèdent le même nombre de 1, ce nombre étant égal au nombre de mots de poids $(2j - 1)$ de $B(n, \delta)$. Comme le tableau contient par définition $a_{2j-1} + a_{2j}$ mots de poids $2j$, on en déduit que

$$a_{2j-1} = \frac{2j(a_{2j-1} + a_{2j})}{n + 1}$$

□

Corollaire 5.4 *La distance minimale d'un code BCH primitif binaire est impaire.*

2 Conditions suffisantes pour que la borne BCH soit atteinte

Divers auteurs ont établi des conditions suffisantes sur la longueur et la distance construite d'un code BCH pour que sa distance minimale atteigne la borne BCH. La première d'entre elles, due à Farr, se déduit de la borne de Hamming, qui minore le nombre de mots d'un code en fonction de sa longueur et de sa distance minimale.

Proposition 5.5 [Far] *La distance minimale d'un code BCH binaire primitif de longueur $(2^m - 1)$ est égale à sa distance construite $\delta = 2t + 1$ si*

$$\sum_{i=0}^{t+1} \binom{2^m - 1}{i} > 2^{mt}$$

Tous les codes BCH de longueur 511 et de distance construite inférieure ou égale à 9 atteignent par conséquent la borne BCH.

Peterson [Pet65] montra que tous les codes BCH $B(n, \delta)$ où δ est un diviseur de n atteignent également cette borne. Cette propriété fut ensuite généralisée par Kasami, Lin et Peterson [KLP66] :

Proposition 5.6 [KLP66]
Si la distance minimale du code BCH binaire $B(n, \delta)$ est exactement δ alors, pour tout entier $h > 0$, le code BCH $B(hn, h\delta)$ atteint la borne BCH.

Cette propriété permet donc dans certains cas de réduire la taille des codes étudiés et de se ramener à des longueur et dimension qui autorisent, sinon la recherche exhaustive, au moins l'utilisation d'un algorithme probabiliste de recherche de mots de poids minimal. L'entier 511 n'ayant que deux facteur premiers (7 et 73), cette proposition ne concerne malheureusement qu'un petit nombre des codes que j'étudie.

La caractérisation des coefficients du polynôme localisateur d'un mot du code BCH $B(2^m-1,\delta)$ permet également de résoudre le cas où $\delta = 2^h - 1$.

Proposition 5.7 [Pet65] *La distance minimale du code BCH primitif de longueur 2^m-1 et de distance construite $\delta = 2^h - 1$ est exactement δ.*

Certaines informations sur la distance minimale proviennent aussi de propriétés d'inclusion entre les codes BCH primitifs et les codes de Reed-Muller poinçonnés. Ces derniers sont définis dans [MS77, page 383] par :

Définition 5.8 (Codes de Reed-Muller poinçonnés) *Le poinçonné du code de Reed-Muller de longueur 2^m et d'ordre r, noté $R^\star(r,m)$ est le code cyclique de longueur 2^m-1 dont le polynôme générateur a pour racines l'ensemble des α^i, pour tous les i dont la décomposition en base 2 est de poids strictement inférieur à $m-k$.*

Par simple inclusion des ensembles de définition des codes BCH et des codes de Reed-Muller poinçonnés, on obtient donc les propositions suivantes.

Proposition 5.9 *Le code BCH binaire primitif au sens strict de longueur (2^m-1) et de distance construite δ contient le code de Reed-Muller poinçonné d'ordre, r $R^\star(r,m)$, pour tout $\delta \leq 2^{m-r}-1$.*

Proposition 5.10 *Le code BCH binaire primitif au sens strict de longueur (2^m-1) et de distance construite δ est contenu dans le code de Reed-Muller poinçonné d'ordre r, $R^\star(r,m)$, pour toute valeur de δ strictement supérieure au plus grand représentant de poids $(m-r-1)$ d'une classe cyclotomique modulo (2^m-1).*

preuve : Le code BCH $B(2^m-1,\delta)$ est inclus dans $R^\star(r,m)$ si

$$\{i < 2^m,\ 1 \leq w(i) \leq m-r-1\} \subset \bigcup_{i=1}^{\delta-1} Cl(i)$$

Comme les éléments de la classe cyclotomique modulo (2^m-1) d'un entier i sont les entiers dont l'écriture en base 2 est obtenue par permutation circulaire des m bits de i, il suffit que $\delta-1$ soit supérieur ou égal au plus grand représentant de poids $m-r-1$ d'une classe cyclotomique. □

De la liste des représentants des classes cyclotomiques modulo 511, établie par exemple dans [Fon95, page 106], je déduis donc, à la table 5.1, pour les

δ	plus petit entier r_1 tel que $R^\star(r_1, 9) \subset B(511, \delta)$	plus grand entier r_2 tel que $B(511, \delta) \subset R^\star(r_2, 9)$
$\delta = 3$	7	7
$5 \leq \delta \leq 7$	6	7
$7 \leq \delta \leq 15$	5	7
$\delta = 17$	4	7
$19 \leq \delta \leq 31$	4	6
$33 \leq \delta \leq 63$	3	6
$65 \leq \delta \leq 73$	2	6
$75 \leq \delta \leq 83$	2	5
$85 \leq \delta \leq 127$	2	4
$129 \leq \delta \leq 171$	1	4
$173 \leq \delta \leq 219$	1	3
$221 \leq \delta \leq 239$	1	2
$\delta = 255$	1	1

TAB. 5.1 –: *Relations d'inclusion entre les codes BCH au sens strict et les codes de Reed-Muller poinçonnés en longueur 511*

différentes distances construites, la plus grande valeur de r_1 et la plus petite valeur de r_2 telles que $R^\star(r_1, 9) \subset B(511, \delta) \subset R^\star(r_2, 9)$.

Certains résultats sur la distribution des poids des codes de Reed-Muller et de leurs poinçonnés peuvent donc être appliqués aux codes BCH. Kasami et Lin [KL72] ont par exemple mis en évidence une nouvelle classe de codes BCH atteignant la borne.

Proposition 5.11 [KL72] *Pour tous les entiers i vérifiant $1 \leq i \leq m - s - 2$ avec $0 \leq s \leq m - 2i$, l'intersection du code BCH binaire de longueur $(2^m - 1)$ et de distance construite $\delta = 2^{m-s-1} - 2^{m-s-i-1} - 1$ avec le code de Reed-Muller poinçonné d'ordre $s + 2$, $R(s + 2, m)$, contient des mots de poids δ.*

Comme les poids du code de Reed-Muller d'ordre r sont divisibles par $2^{\lceil m/r \rceil - 1}$ [McE72], la distance minimale de certains codes BCH ne peut pas prendre n'importe quelle valeur impaire.

Proposition 5.12 [KT69] *Si le code BCH binaire au sens strict $B(2^m - 1, \delta)$ est inclus dans le poinçonné du code de Reed-Muller d'ordre r, alors sa distance minimale d vérifie*

$$d \equiv -1 \bmod 2^{\lceil m/r \rceil - 1}$$

Cette propriété n'a d'intérêt que si le code BCH est inclus dans le poinçonné d'un code de Reed-Muller d'ordre strictement inférieur à $m/2$. Il s'ensuit par exemple que les codes BCH de longueur 511 et de distance construite 93, 109, 117 et 127 n'atteignent pas la borne BCH puisqu'ils sont inclus dans le poinçonné du code de Reed-Muller d'ordre 4.

En longueur 511, ces diverses propositions ne couvrent cependant qu'une petite partie des codes BCH au sens strict.

3 Autres méthodes pour déterminer la distance minimale des codes BCH

Il est donc nécessaire d'employer d'autres méthodes pour calculer la distance minimale des codes BCH n'entrant pas dans les catégories précédentes. Mise à part la recherche exhaustive, trois techniques ont été mises au point à ce jour.

Étude d'un code raccourci

Il s'agit de rechercher un mot de poids δ ou $\delta + 1$ dans certains codes raccourcis de $B(n, \delta)$. Ainsi, en trouvant un mot de poids 13 dans un code [327,274] raccourci de $B(511, 13)$ et un mot de poids 25 dans un code [347,240] raccourci de $B(511, 25)$, Helgert et Stinaff [HS73] ont montré que la distance minimale de ces deux codes BCH était égale à leur distance construite.

Étude des équations de Newton

Les mots du code BCH primitif au sens strict de distance construite δ sont caractérisés par la forme de leur polynôme localisateur σ. Ils correspondent en effet aux polynômes σ dont les coefficients des termes de degré impair strictement inférieur à δ sont nuls [ACS92, lemme 1]. En écrivant les identités de Newton, qui relient les fonctions puissances élémentaires et les fonctions symétriques élémentaires du polynôme localisateur d'un mot de poids w du code BCH, D. Augot, P. Charpin et N. Sendrier [ACS92] ont établi un système d'équations, *les équations de Newton*, dont la résolution produit un mot de poids w du code. Ils ont examiné ces équations à l'aide d'un système de calcul formel afin d'y détecter des contradictions, qui prouvent que le code ne contient pas de mots de poids w. Ils ont ainsi montré que le code $B(511, 123)$ ne contenait pas de mot de poids 123. Comme sa distance minimale d est telle que $d-1$ est un multiple de 4 (cf. proposition 5.12) et que le code $B(511, 127)$ atteint la borne BCH, ils ont pu en déduire que la distance minimale de $B(511, 123)$ était exactement 127.

Recherche d'idempotents

Cette technique développée par Daniel Augot et Nicolas Sendrier consiste à chercher un mot de poids δ ou $\delta + 1$ parmi les *idempotents* de $B(n, \delta)$, *i.e.* les mots dont le polynôme localisateur est à coefficients dans $\mathbb{F}_2$. En effet, la forme simple prise par les équations de Newton quand le polynôme est à coefficients binaires facilite leur résolution. Par une recherche exhaustive sur les idempotents, Augot et Sendrier [AS94] ont donc prouvé que les codes $B(511, \delta)$ atteignaient la borne BCH pour $\delta \in \{19, 39, 45, 53, 57, 79, 83, 91, 103\}$.

4 Applications des algorithmes itératifs de recherche de mots de poids faible

Suite aux travaux de D. Augot, P. Charpin et N. Sendrier, il restait donc douze codes BCH au sens strict de longueur 511 dont la distance n'était pas connue. J'ai donc employé l'algorithme de Stern itératif, décrit au chapitre précédent, pour tenter de trouver dans ces codes un mot de poids égal à la distance construite δ, ou à $\delta + 1$. J'ai ainsi obtenu le résultat suivant :

Théorème 5.13 *La distance minimale des codes BCH au sens strict de longueur 511 et de distance construite* $\delta \in \{29, 37, 41, 43, 51, 87\}$ *est exactement* δ.

preuve : Pour chacun de ces codes, j'ai trouvé un mot de poids $\delta+1$ — comme les mots de poids $\delta + 1$ sont $(n - \delta)/(\delta + 1)$ fois plus nombreux que ceux de poids δ, il est beaucoup plus facile d'en trouver un de manière probabiliste.

Étant donné un élément primitif α de $\mathbb{F}_{2^9}$ défini par $\alpha^9 + \alpha^4 + 1 = 0$, le support de ces mots correspond à l'ensemble des α^i pour les valeurs de i suivantes.

- $\delta = 29$**:**
  (26, 31, 38, 51, 64, 72, 112, 126, 139, 142, 157, 188, 222, 227 , 265, 270, 301, 306, 307, 317, 347, 354, 368, 369, 412, 415, 423, 431, 494, 498)

- $\delta = 37$**:**
  (4, 13, 27, 48, 56, 94, 102, 103, 115, 118, 132, 149, 152, 159, 197, 202, 215, 232, 240, 249, 250, 251, 290, 324, 327, 349, 359 , 360, 367, 383, 396, 423, 461, 493, 494, 499, 504, 509)

- $\delta = 41$**:**
  (9, 20, 30, 37, 38, 42, 43, 53, 66, 68, 83, 93, 95, 106, 108, 110, 111, 175, 185, 202, 234, 250, 262, 270, 321, 342, 362, 363 , 379, 382, 385, 401, 402, 410, 426, 436, 462, 467, 478, 482, 499, 507)

- $\delta = 43$**:**
  (0, 16, 35, 38, 56, 57, 58, 80, 82, 87, 115, 134, 147, 148, 156 , 165, 167, 190, 196, 206, 229, 240, 242, 258, 269, 284, 295, 296, 309, 317, 321, 322, 324, 325, 326, 361, 375, 394, 405, 418 , 429, 444, 460, 492)

- $\delta = 51$**:**
  (6, 10, 11, 17, 22, 54, 57, 64, 76, 79, 85, 87, 93, 97, 101, 121, 122, 139, 140,144, 154, 171, 177, 182, 198, 258, 287, 290, 294, 299, 309, 313, 333, 335, 350, 359, 361, 369, 370, 371, 395, 399, 405, 412, 435, 437, 452, 469, 474, 488, 491, 508)

- $\delta = 87$**:**
  (18, 19, 23, 25, 27, 43, 50, 51, 64, 70, 73, 77, 81, 88, 96, 101 , 102, 116, 117, 143, 146, 152, 158, 163, 165, 166, 173, 179, 192, 193, 195, 197, 199,

203, 210, 212, 225, 230, 240, 244, 252 , 263, 272, 283, 287, 290, 292, 293, 295, 297, 301, 306, 320, 323, 327, 330, 339, 353, 361, 382, 385, 392, 394, 400, 411, 414 , 417, 421, 432, 436, 446, 453, 459, 461, 466, 474, 475, 476, 480, 481, 483, 487, 489, 492, 503, 504, 505, 510)

□

Pour obtenir ces mots de poids $\delta + 1$, j'ai donc utilisé un programme C mettant en œuvre l'algorithme de Stern itératif (version GM) que j'avais au préalable optimisé, en fonction du code étudié, suivant la méthode développée au chapitre précédent.

Comme le montre l'approximation du facteur de travail de cet algorithme obtenue à la proposition 4.17, page 95, le temps de calcul nécessaire pour trouver un mot de poids minimal est d'autant plus élevé que le taux d'expansion du code est proche de 1/2. En effet, pour la distance construite 29, il a fallu moins d'une minute sur une station de travail DEC alpha3000/900 pour trouver un mot de poids 30, pour la distance construite 37 il a fallu trois jours, et pour les distances 41, 43 et 87 entre une et trois semaines. Le mot de poids 52 de $B(511, 51)$ a, lui, été trouvé par F. Chabaud au bout de 50 jours de calculs répartis sur 35 stations de travail (principalement des SPARC 5).

Une rapide comparaison entre ces temps de calcul et le facteur de travail requis pour trouver un mot de poids $\delta + 1$ laisse penser que, pour ces distances construites, le nombre de mots de poids minimal est très élevé. Ainsi, pour la distance construite 29, seules 35 377 itérations ont été nécessaires à l'algorithme de Stern itératif avec les paramètres $p = 1, \sigma = 8$ pour trouver un mot de poids 30 ; s'il n'y avait eu qu'un seul mot de poids 29 — et donc 16 mots de poids 30 —, il aurait fallu en moyenne 7.10^{15} itérations. J'estime par conséquent à 2^{37} le nombre de mots de poids minimal de $B(511, 29)$.

La table 5.2 des codes BCH au sens strict de longueur 511 récapitule tous ces résultats. Pour cette longueur, seules les distances minimales des six codes BCH de distance construite 59, 61, 75, 77, 85 et 107 restent à déterminer.

J'ai également tenté avec l'algorithme de Stern itératif de trouver des mots de poids 107 ou 108 dans le code BCH de distance construite 107. Le facteur de travail moyen requis pour trouver un mot de poids 108 dans $B(511, 107)$ ne représente que 9 % du nombre d'opérations nécessaires pour trouver un mot de poids 88 dans $B(511, 87)$. Or, après trois mois de calcul sur un DEC alpha3000/900, mon programme n'a trouvé aucun mot de poids 107 ou 108 alors que moins de trois semaines avaient suffi pour trouver un mot de poids 88 dans $B(511, 88)$. Deux interprétations (non exclusives) viennent alors à l'esprit : soit le code BCH de distance construite 107 n'atteint pas la borne BCH — et sa distance minimale est alors 111 —, soit son nombre de mots de poids minimal est très faible. Cette seconde hypothèse est de toute façon étayée par un résultat de D. Augot [Aug93, théorème III.7] qui montre que les mots de poids minimal des BCH se raréfient quand la distance construite est proche de $(2^{m-2} - 1$, c'est-à-dire ici autour de 127. En effet, l'ensemble des mots de poids 127 de

n	k	δ	d	argument	réf.	n	k	δ	d		argument	réf.
511	502	3	3	H	[Far]	511	241	73	73		SP(7,1)	[KLP66]
	493	5	5	H	[Far]		238	75	≥ 75		_	_
	484	7	7	H	[Far]		229	77	≥ 77		_	_
	475	9	9	H	[Far]		220	79	79		ID	[ACS92]
	466	11	11	RM(6)	[KL72]		211	83	83		ID	[ACS92]
	457	13	13	R	[HS73]		202	85	≥ 85		_	_
	448	15	15	RM(15)	[KL72]		193	87	87		**	**
	439	17	17	RE	[ACS92]		184	91	91		ID	[ACS92]
	430	19	19	ID	[ACS92]		175	93	95	#	4-DI	[KT69]
	421	21	21	SP(73,3)	[KLP66]		166	95	95		RM(3)	[KL72]
	412	23	23	RM(5)	[KL72]		157	103	103		ID	[ACS92]
	403	25	25	R	[HS73]		148	107	≥ 107		_	_
	394	27	27	RM(5)	[KL72]		139	109	111	#	4-DI	[KT69]
	385	29	29	**	**		130	111	111		RM(3)	[KL72]
	376	31	31	RM(4)	[KL72]		121	117	119	#	4-DI	[KT69]
	367	35	35	SP(73,5)	[KLP66]		112	119	119		RM(3)	[KL72]
	358	37	37	**	**		103	123	127	##	EN	[ACS92]
	349	39	39	ID	[ACS92]		94	125	127	#	4-DI	[KT69]
	340	41	41	**	**		85	127	127		RM(2)	[KL72]
	331	43	43	**	**		76	171	171		RE	[ACS92]
	322	45	45	ID	[ACS92]		67	175	175		RE	[ACS92]
	313	47	47	RM(4)	[KL72]		58	183	183		RE	[ACS92]
	304	51	51	**	**		49	187	187		RE	[ACS92]
	295	53	53	EN	[ACS92]		40	191	191		RM(2)	[KL72]
	286	55	55	RM(4)	[KL72]		31	219	219		SP(7,3)	[KLP66]
	277	57	57	ID	[ACS92]		28	223	223		RM(2)	[KL72]
	268	59	≥ 59	_	_		19	239	239		RM(2)	[KL72]
	259	61	≥ 61	_	_		10	255	255		RM(1)	[KL72]
	250	63	63	RM(3)	[KL72]							

#	$d = \delta + 2$
##	$d = \delta + 4$

**	nouveau résultat obtenu avec l'algorithme de Stern itératif
H	borne de Hamming (prop. 5.5)
RE	recherche exhaustive
ID	recherche d'un idempotent
EN	contradiction dans les équations de Newton
RM(s)	intersection avec le code de Reed-Muller poinçonné d'ordre s (prop. 5.11)
R	code raccourci
SP(n_2, δ_2)	sous-code produit : $n = n_1 n_2$, $\delta = n_1 \delta_2$ et la distance minimale de $B(n_2, \delta_2)$ est exactement δ_2
4-DI	4-divisibilité des poids de RM(4), car $B(511, \delta)$ est inclus dans le code de Reed-Muller poinçonné d'ordre 4 pour $\delta > 85$ (prop. 5.12)

TAB. 5.2 –: *Distance minimale des codes BCH au sens strict de longueur 511*

$B(511, 127)$ est réduit aux mots de poids minimal du code de Reed-Muller poinçonné d'ordre 2, $RM^{\star}(2, 9)$.

Conclusion sur la sécurité des cryptosystèmes à mots de poids faible

J'ai donc conçu deux nouveaux algorithmes probabilistes pour décoder un code linéaire jusqu'à une distance donnée ou y trouver un mot de poids minimal. Ce sont actuellement les plus performants dans ce domaine comme en témoigne leur capacité à déterminer la distance minimale de certains codes BCH de longueur 511 qu'aucune autre méthode n'avait pu trouver.

Une analyse très précise de leur complexité grâce à leur modélisation par un processus markovien m'a permis d'optimiser les différents paramètres dont ils dépendent. J'ai alors pu réduire de façon substantielle le facteur de travail moyen nécessaire pour attaquer les principaux systèmes à mots de poids faible ; ainsi l'algorithme de Stern itératif optimisé fournit une cryptanalyse du système de chiffrement de McEliece 128 fois plus rapide que celle de Lee et Brickell puisqu'elle nécessite en moyenne 2^{64} opérations binaires.

Bien que ce facteur de travail semble impliquer que cette attaque n'est actuellement pas réalisable, j'ai montré qu'elle révélait d'importantes faiblesses dans la sécurité du système tel qu'il fut à l'origine décrit par McEliece. Il suffit en effet de moins de trois mois de calculs sur dix stations DEC alpha500/266 pour déchiffrer avec mon algorithme un message sur 10 000. L'utilisation des paramètres proposés par McEliece (un code de Goppa de longueur 1024, de dimension 524 et de capacité de correction 50) rend donc le système vulnérable à tout attaquant disposant de quelques dizaines de stations de travail rapides.

Une seconde faiblesse importante de ce système de chiffrement est que la connaissance de 23 % du texte clair suffit à retrouver le message tout entier en un temps raisonnable. Vu la forte redondance du langage, il est donc dangereux de chiffrer intégralement un fichier ; l'emploi du système de McEliece avec ses paramètres originaux impose par conséquent que seules les parties réellement confidentielles d'un message — le montant des transactions, les noms propres,... — soient chiffrées. Pour la même raison, l'utilisation d'un padding constant est à proscrire.

J'ai en revanche conclu à la solidité cryptographique du schéma d'identification de Stern dès lors que les clefs secrètes des utilisateurs sont renouvelées suffisamment fréquemment, par exemple tous les ans. La variante de ce système

proposée par Véron afin de minimiser le nombre de bits échangés lors d'une transaction est, elle, beaucoup vulnérable : j'ai en effet montré que 56 jours de calculs sur dix stations DEC alpha500/266 suffisaient à retrouver la clef secrète d'un utilisateur avec une probabilité supérieure à 1/3500. Pour garantir une sécurité convenable, il faudrait alors que la durée de vie des clefs ne dépasse pas une quinzaine de jours, ce qui semble beaucoup trop éphémère pour un tel système.

Partie II

Construction de fonctions résilientes sur un alphabet fini

Introduction

Réaliser un système de chiffrement à clef secrète ou une fonction de hachage qui résiste à la cryptanalyse pourrait paraître relativement simple : ne suffit-il pas de modifier et de mélanger les données de manière suffisamment inextricable et irrégulière pour qu'il soit impossible d'appréhender simplement le système dans sa globalité ? Mais toute la complexité de la cryptographie se cache derrière ces deux adjectifs qui semblent bien peu objectifs : "inextricable" et "irrégulier". Qu'est-ce donc qu'une opération "irrégulière", qu'une suite de calculs "inextricable" ? Définir ou quantifier mathématiquement de telles propriétés paraît à première vue incongru. Il est pourtant évident que certains systèmes sont plus "inextricables" que d'autres, comme en témoigne leur longévité. Par quel prodige le DES résiste-t-il aux attaques conjuguées de toute la communauté scientifique devient presque 30 ans, alors que tant d'autres chiffrements à clef secrète n'ont survécu que quelques mois ? Cette sécurité du DES intrigue d'autant plus que les recherches qui ont abouti à sa conception demeurent toujours secrètes.

Même s'ils échouent encore face au DES, les cryptographes ont cependant réussi à définir un certain nombre de critères qui conditionnent la sécurité d'un système. Un générateur pseudo-aléatoire résultant de l'assemblage de plusieurs registres à décalage à rétroaction linéaire n'est résistant que si la fonction de combinaison est à la fois non-linéaire [Mas69] et sans-corrélation [Sie85] ; une primitive cryptographique conventionnelle, telle une fonction de hachage ou un chiffrement par blocs, doit associer des propriétés de confusion et de diffusion. Les premières peuvent être obtenues grâce à des fonctions hautement non-linéaires, comme les fonctions courbes [Rot76, MS90], et les secondes par le biais de fonctions appelées multipermutations [SV95, Vau95a]. Contrairement à ce que laisse présager leur diversité, toutes ces propriétés apportent en fait des réponses aux deux grands problèmes définis par Shannon [Sha49] : comment casser la structure (algébrique, statistique,...) des données, et comment diffuser l'information.

C'est essentiellement à ce second problème, celui de la *diffusion*, que je m'intéresse dans cette partie, qui résulte notamment d'un travail mené avec Paul Camion. La finalité principale de cette étude est d'unifier les différents critères cryptographiques de diffusion — fonctions sans-corrélation, générateurs aléatoires localement parfaits, multipermutations — à travers la notion de fonction sans-corrélation, ou celle plus restrictive de fonction résiliente. L'intérêt de cette

description réside principalement dans le fait que ces fonctions peuvent être caractérisées, dans un cadre très général, de diverses manières : par des structures combinatoires, par des propriétés de leur transformée de Fourier ou par des relations matricielles [GS95, CC96a]. Cette richesse apporte par conséquent de multiples outils de démonstration : alors que la caractérisation combinatoire implique de façon naturelle des bornes sur l'ordre maximal de résilience d'une fonction, la caractérisation à l'aide de la transformée de Fourier m'a, elle, permis de mettre en évidence une nouvelle construction de fonctions résilientes [CC96a]. La représentation d'une fonction sans-corrélation q-aire par sa forme algébrique normale [Mul54, Ree54] fournit également de précieuses propriétés puisqu'elle permet entre autres de lier l'ordre de non-corrélation d'une fonction et son ordre de non-linéarité [Sie85, CC96b].

Le premier chapitre de cette partie est donc consacré aux diverses manières d'appréhender les fonctions résilientes : j'y caractérise de trois façons différentes les fonctions sans-corrélation définies dans un cadre très général — *i.e.* sans imposer la moindre structure algébrique sur leurs ensembles de départ et d'arrivée —. Je donne également, en me fondant sur les travaux de Philippe Delsarte [Del72, Del73b], des bornes sur l'ordre maximal de résilience que peut atteindre une fonction. Dans le second chapitre, je m'intéresse aux propriétés de non-linéarité des fonctions résilientes définies sur un corps fini. Le résultat fondamental de ce chapitre est la mise en valeur d'un compromis entre l'ordre de non-corrélation d'une fonction q-aire et son ordre de non-linéarité. Ses conséquences sont nombreuses en cryptographie car des valeurs simultanément élevées de ces deux paramètres sont souvent souhaitables. Je parviens néanmoins à construire des familles infinies de fonctions t-résilientes de non-linéarité optimale. Le chapitre suivant montre, lui, que d'autres objets cryptographiques, les générateurs aléatoires localement parfaits et les multipermutations, peuvent aussi être définis en termes de fonctions sans-corrélation. Grâce à cette équivalence, je développe un nouveau critère de sécurité pour les primitives cryptographiques conventionnelles qui enrichit la notion de multipermutation introduit par Claus Schnorr et Serge Vaudenay. Enfin, j'expose une nouvelle méthode, développée dans un article présenté à Eurocrypt'96 [CC96a], qui permet de construire des fonctions ayant à la fois un grand nombre de variables et un ordre de résilience élevé.

Chapitre 6

Caractérisations des fonctions résilientes sur un alphabet fini

La notion de fonction *résiliente* a été introduite au milieu des années 80 indépendamment par Bennett, Brassard et Robert [BBR88] et par Chor *et al.* [CGH$^+$85] dans le cadre d'applications très précises.

Dans le premier cas, les fonctions résilientes apportaient une solution au problème de distribution des clefs en cryptographie quantique. Dans le second, il s'agissait de concevoir une méthode efficace permettant à plusieurs processeurs dans un système distribué de partager une même suite aléatoire. La solution triviale qui consiste à ce qu'un seul de ces processeurs génère la suite et la transmette aux autres impose un fonctionnement parfait de tous les processeurs. Elle ne tolère pas qu'un des processeurs devienne défectueux et fournisse, par exemple, une suite aléatoire biaisée. Pour pallier cette éventualité, on peut additionner bit à bit différentes suites aléatoires produites indépendamment par chacun des n processeurs. La suite ainsi obtenue est donc bien une suite aléatoire tant qu'au moins l'un des processeurs n'est pas défectueux. Cette méthode est cependant très coûteuse puisque le rapport entre le nombre de bits utilisables et le nombre de bits produits est seulement $\frac{1}{n}$. L'utilisation d'une fonction t-résiliente conduit alors à un compromis entre le nombre autorisé de processeurs défectueux et la proportion de bits utilisables : elle permet de générer une suite aléatoire résistant à la panne de t processeurs avec un taux de bits utilisables égal à $\frac{n-t}{n}$.

Toutefois, avant l'apparition de cette terminologie, le concept de fonction résiliente était déjà sous-jacent dans les travaux de Siegenthaler [Sie84] à travers la notion plus générale de *fonction sans corrélation.* De telles fonctions étaient alors utilisées pour assembler plusieurs registres à décalage à rétroaction linéaire afin d'obtenir un générateur pseudo-aléatoire solide cryptographiquement.

Mais, dans tous ces travaux, la propriété de résilience n'était envisagée que pour des fonctions booléennes, c'est-à-dire définies sur le corps $\mathbb{F}_2$. Dans ce cas, elle a été caractérisée de différentes manières successivement par Xiao et Massey [XM88] (caractérisation par la transformée de Fourier de la fonction) et par

Camion, Carlet, Charpin et Sendrier [CCCS92] (caractérisation combinatoire). Ces deux caractérisations ont été récemment généralisées aux fonctions définies sur n'importe quel corps fini par Gopalakrishnan et Stinson [GS95]. Ces derniers ont également introduit une troisième caractérisation exprimée, elle, en termes de matrices.

Dans ce chapitre, je montre que les notions de fonctions sans corrélation et de fonctions t-résilientes peuvent être étendues aux fonctions définies sur un alphabet fini quelconque. Je généralise ainsi les travaux de Gopalakrishnan et Stinson en caractérisant ces propriétés de trois manières différentes : en termes combinatoires, en termes de transformée de Fourier et en termes de matrices. J'expliciterai ensuite le lien entre les tableaux orthogonaux et les codes correcteurs d'erreurs, mis en évidence par Philippe Delsarte [Del72], et je verrai qu'il induit une méthode de construction de fonctions résilientes à partir de codes. Enfin, je donnerai différentes bornes sur la taille d'un tableau orthogonal et sur l'ordre de résilience d'une fonction. Ces bornes sont des généralisations pour des fonctions définies sur un groupe abélien fini des bornes exposées dans le cas binaire par Bierbrauer, Gopalakrishnan et Stinson [BGS94].

1 Définitions

Dans toute la suite $\mathcal{F}$ désigne un alphabet fini à q éléments, avec $q \geq 2$, et E un ensemble fini. Je considère une fonction f définie sur $\mathcal{F}^n$ et à valeurs dans E dont l'entrée forme un ensemble $\{X_1, \ldots, X_n\}$ de n variables aléatoires indépendantes à valeurs dans $\mathcal{F}$ uniformément distribuées. Cela signifie donc que chaque vecteur de l'ensemble $\mathcal{F}^n$ apparaît en entrée de la fonction avec une probabilité $\frac{1}{q^n}$.

Je m'intéresse alors aux propriétés suivantes de la fonction f :

Définition 6.1 *Soit f une fonction définie sur $\mathcal{F}^n$ et à valeurs dans E.*

- *f est* équilibrée *si la variable aléatoire $Y = f(X_1, \ldots, X_n)$ est uniformément distribuée.*
- *f est* sans corrélation par rapport à l'ensemble d'indices $T \subset \{1, \ldots, n\}$ *si la distribution de probabilités de la sortie Y est inchangée quand les entrées $(X_i)_{i \in T}$ sont fixées et que les $(X_i)_{i \notin T}$ forment un ensemble de variables aléatoires indépendantes uniformément distribuées.*
- *f est* sans corrélation d'ordre t *(sur l'alphabet $\mathcal{F}$) si, pour tout ensemble d'indices T de cardinal inférieur ou égal à t, f est sans corrélation par rapport à T.*
- *f est* t-résiliente *si f est sans corrélation d'ordre t et équilibrée.*

Ainsi, la fonction booléenne à n variables f définie par $f(X_1, \ldots, X_n) = X_1 + \ldots + X_n$ est une fonction $(n-1)$-résiliente. En effet, si $n-1$ variables d'entrée, par exemple $X_1, \ldots, X_{n-1}$, sont fixées et que X_n est tirée uniformément dans $\mathbb{F}_2$, alors la probabilité que $f(X_1, \ldots, X_n)$ soit égale à 1 (ou 0) est exactement $\frac{1}{2}$.

2 Caractérisation en termes de tableaux orthogonaux

Les fonctions sans corrélation et les fonctions résilientes sont des objets très proches d'une structure combinatoire bien connue, introduite en 1947 par Rao sous le nom de *tableau orthogonal*.

Définition 6.2 (tableau orthogonal) *Un* tableau orthogonal $\mathcal{A}$ de taille M, à n contraintes, de force t et d'indice λ *sur l'alphabet $\mathcal{F}$ (ou à q niveaux) est un tableau d'éléments de $\mathcal{F}$ à M lignes et n colonnes tel que, dans tout ensemble de t colonnes de $\mathcal{A}$, chacun des t-uplets de $\mathcal{F}^t$ apparaît dans exactement λ lignes. Un tel tableau est noté (M, n, q, t). On a trivialement $M = \lambda q^t$.*

Je ne considérerai par la suite que des tableaux orthogonaux *simples*, c'est-à-dire des tableaux dont toutes les lignes sont distinctes.

Exemple 6.3: Le tableau 8×6 suivant

$$\begin{array}{c} 0\ 0\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 1\ 1\ 0 \\ 0\ 1\ 0\ 1\ 0\ 1 \\ 0\ 0\ 1\ 0\ 1\ 1 \\ 1\ 1\ 0\ 0\ 1\ 1 \\ 1\ 0\ 1\ 1\ 0\ 1 \\ 0\ 1\ 1\ 1\ 1\ 0 \\ 1\ 1\ 1\ 0\ 0\ 0 \end{array}$$

est un tableau orthogonal de taille 8, à 6 contraintes, de force 2 et d'indice 2 sur l'alphabet $\{0, 1\}$. En effet, dans les lignes du tableau formé par n'importe quel ensemble de 2 colonnes, chacun des 4 couples binaires apparaît exactement 2 fois.

Dans la mesure où tout tableau orthogonal de force t et d'indice λ est aussi un tableau orthogonal de force $t-1$ et d'indice λq, le seul paramètre pertinent est la force maximale du tableau.

Le lien entre les fonctions booléennes sans corrélation et les tableaux orthogonaux a été mis en évidence dans [CCCS92] à travers la propriété suivante : une fonction booléenne f à n variables est sans corrélation d'ordre t si et seulement si le tableau dont les lignes sont les n-uplets de $f^{-1}(1)$ est un tableau orthogonal de force t. Dans [GS95], Gopalakrishnan et Stinson ont prouvé directement qu'une fonction $f : \mathbb{F}_q^n \to \mathbb{F}_q^\ell$ est sans corrélation d'ordre t si, pour tout y de $\mathbb{F}_q^\ell$, le tableau dont les lignes sont les éléments de $f^{-1}(y)$ est un tableau orthogonal de force t. En fait, caractériser les fonctions sans corrélation en termes de tableaux orthogonaux revient simplement à traduire la définition probabiliste

initiale en une définition combinatoire. Cette caractérisation ne requiert donc aucune structure particulière sur les ensembles $\mathcal{F}$ et E.

Proposition 6.4 (caractérisation en termes de tableaux orthogonaux) *Soit $f : \mathcal{F}^n \to E$, où $\mathcal{F}$ et E sont des ensembles finis.*
La fonction f est sans corrélation d'ordre t sur $\mathcal{F}$ si et seulement si, $\forall y \in E$, les éléments de $f^{-1}(y)$ constituent les lignes d'un tableau orthogonal $\mathcal{A}_y$ de force t sur $\mathcal{F}$.
La fonction f est t-résiliente si, en plus, tous les tableaux $(\mathcal{A}_y)_{y\in E}$ sont de même taille.

Nous verrons par la suite que cette caractérisation combinatoire est particulièrement adaptée à la détermination de l'ordre maximal de résilience possible pour une fonction définie sur un alphabet donné.

3 Caractérisation en termes de transformée de Fourier

Dans [XM88], Xiao et Massey ont caractérisé les fonctions booléennes sans corrélation à l'aide d'une condition sur la transformée de Fourier de la fonction. Une telle caractérisation a l'avantage d'être beaucoup plus maniable qu'une définition probabiliste. Cette propriété a été généralisée par Gopalakrishnan et Stinson dans le cas où les ensembles $\mathcal{F}$ et E sont des corps finis. Je montre ici qu'une propriété similaire peut être exprimée dès lors que les ensembles $\mathcal{F}$ et E sont munis d'une structure de groupe abélien.

3.1 Caractères d'un groupe abélien

Soit G un groupe abélien fini dont la loi de groupe est notée additivement. Un *caractère* χ de G est un homomorphisme de G dans le groupe multiplicatif $\mathbb{C}^\star$ des nombres complexes. Tout caractère χ vérifie donc $\chi(g+g') = \chi(g)\chi(g')$, pour tout couple (g, g') d'éléments de G.

Notons e l'exposant du groupe G, c'est-à-dire le plus petit entier tel que $\forall g \in G,\ eg = 0$. On a alors, pour tout élément g de G, $\chi(g)^e = \chi(eg) = 1$, ce qui implique que tout caractère de G est à valeurs dans le groupe cyclique des racines e-ièmes de l'unité.

On définit alors la somme de deux caractères χ et χ' par

$$\forall g \in G,\ (\chi + \chi')(g) = \chi(g)\chi'(g)$$

Comme le groupe des racines e-ièmes de l'unité est commutatif, l'ensemble des caractères de G forme un groupe abélien additif, noté G', dont l'élément neutre est le caractère principal χ_0 défini par $\forall g \in G,\ \chi_0(g) = 1$. Le groupe G' est, de plus, isomorphe à G. Une démonstration de cette propriété classique est donnée

par exemple dans [Lan84]. La proposition suivante résume donc l'ensemble de ces propriétés classiques.

Proposition 6.5 *L'ensemble des caractères d'un groupe abélien fini G est un groupe abélien additif G' isomorphe à G ; G est lui-même le groupe des caractères de G'.*

On peut donc identifier G et son groupe des caractères, par exemple en indexant les caractères par les éléments du groupe. C'est pourquoi, pour tout élément x de G et tout caractère χ_y, je noterai désormais $< x, y >$ l'image de x par le caractère χ_y. Cette notation, employée par exemple dans [Del73a], est motivée par le fait que l'application

$$\begin{array}{ccc} G \times G & \rightarrow & \mathbb{C} \\ (x, y) & \mapsto & < x, y > \end{array}$$

est bilinéaire.

Exemples :

- Si G est le groupe additif $(\mathbb{F}_q, +)$ avec $q = p^s$ et p premier, alors

  $$< x, y >= \theta^{Tr_{\mathbb{F}_q/\mathbb{F}_p}(xy)}$$

  où θ est une racine primitive p-ième de l'unité dans $\mathbb{C}$.
- Si G est le groupe cyclique d'ordre q, $(\mathbb{Z}/q\mathbb{Z}, +)$, alors

  $$< x, y >= \theta^{xy}$$

  où θ est une racine q-ième de l'unité dans $\mathbb{C}$. Le produit xy est ici effectué dans l'anneau $\mathbb{Z}/q\mathbb{Z}$.

Le lemme suivant explicite la forme du groupe des caractères du produit cartésien de deux groupes abéliens.

Lemme 6.6 *Soient F et G deux groupes abéliens finis et F' et G' leurs groupes de caractères respectifs. Alors le groupe des caractères de $F \times G$ est $F' \times G'$, et pour $h = (f, g)$ et $h' = (f', g')$ dans $F \times G$, on a $< h, h' >=< f, f' >< g, g' >$.*

preuve : L'application

$$\begin{array}{rccc} \psi_{f',g'} : & F \times G & \rightarrow \mathbb{C} & \\ & (f, g) & \mapsto & < f, f' >< g, g' > \end{array}$$

est un caractère de $F \times G$. On peut alors considérer l'homomorphisme Φ qui, à tout couple de caractères $(\chi_{f'}, \chi_{g'})$ de $F' \times G'$, associe le caractère $\psi_{f',g'}$ dans $(F \times G)'$. Supposons que l'image de $(\chi_{f'}, \chi_{g'})$ par Φ soit le caractère principal de $F \times G$. On a alors, $\forall h = (f, g)$, $< f, f' >< g, g' >= 1$. En appliquant cette égalité à tous les $h = (f, 0)$ (resp. $h = (0, g)$), on en déduit que $f' = 0$ (resp. $g' = 0$), et par conséquent que Φ est injective. Les groupes finis $F' \times G'$ et $(F \times G)'$ étant de même taille, Φ est bien un isomorphisme. □

On en déduit en particulier que si F est un groupe abélien fini et F' son groupe des caractères, alors le groupe des caractères de F^n est $(F')^n$. Le groupe des caractères vérifie également la relation d'orthogonalité suivante :

Proposition 6.7 *Soit G un groupe abélien fini. Alors*

$$\sum_{x \in G} < x, y >= \begin{cases} |G| & si\ y = 0 \\ 0 & sinon \end{cases}$$

Corollaire 6.8 *La* matrice des caractères *S d'un groupe abélien fini G définie par $\forall (x, y) \in G \times G, S(x, y) =< x, y >$ vérifie*

$$SS^\star = S^\star S = |G| Id$$

où $S^\star$ est la transposée de la matrice conjuguée de S.

3.2 Transformation de Fourier

Dès lors que l'on est amené à manier des caractères, il est particulièrement commode d'utiliser la *transformation de Fourier*.

Définition 6.9 *L'algèbre de groupe $\mathbb{C}G$ d'un groupe abélien fini G sur le corps des complexes est constituée de l'ensemble des séries formelles*

$$\mathbf{a} = \sum_{x \in G} a_x Z^x, \ \text{où } a_x \in \mathbb{C}$$

Les opérations dans $\mathbb{C}G$ sont définies de manière usuelle :

$$\begin{aligned} \sum_{x \in G} a_x Z^x + \sum_{x \in G} b_x Z^x &= \sum_{x \in G} (a_x + b_x) Z^x \\ \forall r \in \mathbb{C},\ r \sum_{x \in G} a_x Z^x &= \sum_{x \in G} r a_x Z^x \\ \text{et } \sum_{x \in G} a_x Z^x \sum_{y \in G} b_y Z^y &= \sum_{z \in G} Z^z \Big(\sum_{x+y=z} a_x b_y \Big) \end{aligned}$$

Tout caractère peut alors être étendu par linéarité à l'algèbre $\mathbb{C}G$ par :

$$< \mathbf{a}, y >=< \sum_{x \in G} a_x Z^x, y >= \sum_{x \in G} a_x < x, y >$$

On note $\hat{a}_y$ le nombre complexe $< \mathbf{a}, y >$, appelé *coefficient de Fourier* de $\mathbf{a}$. La *transformation de Fourier* est alors l'application linéaire

$$\begin{array}{rcl} \mathbb{C}G & \rightarrow & \mathbb{C}G' \\ \mathbf{a} & \mapsto & \sum_{y \in G} \hat{a}_y Z^y \end{array}$$

L'orthogonalité de la matrice des caractères de G explicitée dans le corollaire 6.8 implique alors l'existence d'une transformation inverse. La transformation de Fourier est donc un isomorphisme d'algèbre de $\mathbb{C}G$ dans $\mathbb{C}G'$ où la multiplication dans $\mathbb{C}G'$ est le produit de Hadamard (*i.e.* le produit composantes à composantes). Tout élément $\mathbf{a}$ de $\mathbb{C}G$ est donc déterminé de façon unique par ses coefficients de Fourier $(\hat{a}_y)_{y \in G}$.

3.3 Caractérisation des fonctions sans corrélation à l'aide de caractères

L'utilisation des caractères permet de généraliser la caractérisation donnée par Gopalakrishnan et Stinson pour les fonctions sans corrélation définies sur $\mathbb{F}_q$. Ce résultat se déduit immédiatement d'un théorème démontré par Philippe Delsarte [Del73a], qui traduit la propriété combinatoire de tableau orthogonal en une condition à base de caractères. Je donne ici un énoncé légèrement différent de celui de Delsarte qui, lui, renvoie à la propriété de *t-design*. Je considère un alphabet fini $\mathcal{F}$ muni d'une structure de groupe abélien et l'espace métrique formé par $\mathcal{F}^n$ muni de la distance de Hamming. Pour tout élément x de $\mathcal{F}^n$, je note $w_H(x)$ le nombre de composantes non nulles de x dans $\mathcal{F}$.

Théorème 6.10 (caractérisation des tableaux orthogonaux à l'aide de caractères [Del73a, théorème 4.4, page 43]) *Soit $\mathcal{F}$ un groupe abélien fini à q éléments. Un ensemble $\mathcal{M}$ de λq^t vecteurs de $\mathcal{F}^n$ forme les lignes d'un tableau orthogonal à n contraintes, de force t et d'indice λ sur $\mathcal{F}$ si et seulement si*

$$\forall y \in \mathcal{F}^n,\ 1 \leq w_H(y) \leq t,\ \sum_{x \in \mathcal{M}} < x, y >= 0 \tag{6.1}$$

preuve : Soit $y \in \mathcal{F}^n$ tel que $1 \leq w_H(y) \leq t$ et $T = supp(y) = \{i_1, ..., i_\tau\}$. On a alors

$$\sum_{x \in \mathcal{M}} < x, y >= \sum_{x \in \mathcal{M}} \prod_{i \in T} < x_i, y_i >$$

Considérons $\omega_1, \ldots, \omega_\tau$ τ valeurs fixées dans $\mathcal{F}$ et $N_T(\omega_1, \ldots, \omega_\tau)$ le cardinal de l'ensemble des vecteurs x de $\mathcal{M}$ tels que $\forall 1 \leq j \leq \tau,\ x_{i_j} = \omega_j$. On a donc

$$\sum_{x \in \mathcal{M}} < x, y >= \sum_{(\omega_1, \ldots, \omega_\tau) \in \mathcal{F}^\tau} N_T(\omega_1, \ldots, \omega_\tau) \prod_{j=1}^{\tau} < \omega_j, y_{i_j} >$$

La condition 6.1 peut donc s'écrire

$$\forall T \text{ tel que } 1 \leq |T| \leq t,\ \forall y \in \mathcal{F}^{|T|},\ (\hat{N}_T)_y = 0$$

Cette nouvelle condition équivaut alors à dire que, pour chacun de ces ensembles T, la fonction N_T est constante sur $\mathcal{F}^{|T|}$, et donc par définition que $\mathcal{M}$ est un tableau orthogonal de force t.

□

Je peux maintenant en déduire une caractérisation générale des fonctions sans corrélation en termes de transformée de Fourier.

Théorème 6.11 (caractérisation des fonctions sans corrélation à l'aide de caractères) *Soient $\mathcal{F}$ et E deux groupes abéliens finis.*
La fonction $f : \mathcal{F}^n \to E$ est sans corrélation d'ordre t sur $\mathcal{F}$ si et seulement si

$$\forall v \in E,\ \forall u \in \mathcal{F}^n,\ 1 \leq w_H(u) \leq t \quad \sum_{x \in \mathcal{F}^n} < x, u >< f(x), v >= 0 \qquad (6.2)$$

preuve : Notons $\hat{a}_{y,u}$ la somme $\sum_{x \in f^{-1}(y)} < x, u >$, avec la convention $\hat{a}_{y,u} = 0$ quand $f^{-1}(y) = \emptyset$. La condition 6.2 s'écrit alors

$$\forall v \in E,\ \forall u \in \mathcal{F}^n,\ 1 \leq w_H(u) \leq t \quad \sum_{y \in E} \hat{a}_{y,u} < y, v >= 0$$

La matrice des caractères du groupe abélien E étant, d'après le corollaire 6.8, inversible, cette condition équivaut à

$$\forall y \in E,\ \forall u \in \mathcal{F}^n,\ 1 \leq w_H(u) \leq t,\ \ \hat{a}_{y,u} = 0$$

D'après le théorème de Delsarte, ceci revient à dire que, pour tout y de E, les éléments de $f^{-1}(y)$ forment les lignes d'un tableau orthogonal de force t sur $\mathcal{F}$.
□

4 Caractérisation en termes de matrices

Gopalakrishnan et Stinson donnent une troisième caractérisation des fonctions sans corrélation et des fonctions résilientes qui, elle, s'exprime à l'aide de matrices. Cette caractérisation résulte du lemme dit *lemme de la combinaison linéaire*. Ce lemme énoncé par Gopalakrishnan et Stinson dans le cas où l'alphabet $\mathcal{F}$ est un corps fini est une généralisation du *XOR-lemma* prouvé indépendamment par Xiao et Massey [XM88], et Chor *et al.* [CGH$^+$85]. En m'inspirant de la démonstration très concise donnée par Brynielsson [Bry89], je le généralise au cas où l'alphabet $\mathcal{F}$ à q éléments est muni de la structure d'anneau $\mathbb{Z}_q$ ou de corps $\mathbb{F}_q$.

Lemme 6.12 (Lemme de la combinaison linéaire) *Soit Y une variable aléatoire discrète à valeurs dans un ensemble fini quelconque E et $X_1, \ldots, X_n$ n variables aléatoires indépendantes à valeurs dans un ensemble fini $\mathcal{F}$ muni soit de la structure de corps $\mathbb{F}_q$, soit de la structure d'anneau $\mathbb{Z}_q$. La variable Y est indépendante des n variables $X_1, X_2, \ldots, X_n$ si et seulement si elle est indépendante de la somme $h_1X_1 + h_2X_2 + \ldots + h_nX_n$ pour tout choix de $h_1, h_2, \ldots, h_n$ dans $\mathcal{F}$ non tous nuls.*

preuve : La condition énoncée est clairement nécessaire puisque l'indépendance de Y et des X_i implique que, pour tous $\alpha \in \mathcal{F}$ et $\beta \in E$,

$$\begin{aligned} Pr(h_1X_1 + \ldots + h_nX_n = \alpha | Y = \beta) &= \sum_{h.g=\alpha} Pr(X_1 = g_1, \ldots, X_n = g_n | Y = \beta) \\ &= \sum_{h.g=\alpha} Pr(X_1 = g_1, \ldots, X_n = g_n) \\ &= Pr(h_1X_1 + \ldots + h_nX_n = \alpha) \end{aligned}$$

Elle est également suffisante : notons G l'ensemble $\mathcal{F}^n$ et e l'exposant du groupe abélien $(G, +)$. Étant donné $\beta \in E$, je pose $x_g = Pr(X_1 = g_1, \ldots, X_n = g_n | Y = \beta)$ et $y_g = Pr(X_1 = g_1, \ldots, X_n = g_n)$. Je considère, dans l'algèbre de groupe $\mathbb{C}G$, les séries formelles $\mathbf{x} = \sum_{g \in G} x_g Z^g$ et $\mathbf{y} = \sum_{x \in G} y_g Z^g$. Je vais alors montrer que, pour tout h dans G, les coefficients de Fourier $\hat{x}_h$ et $\hat{y}_h$ sont égaux. Pour h non nul, j'écris

$$\begin{aligned} \hat{x}_h &= \sum_{g \in G} Pr(X_1 = g_1, \ldots, X_n = g_n | Y = \beta) < g, h > \\ &= \sum_{\alpha \in \mathcal{F}} \sum_{h.g=\alpha} Pr(X_1 = g_1, \ldots, X_n = g_n | Y = \beta)\ \theta^{\alpha} \text{ si } \mathcal{F} = \mathbb{Z}_q \\ &= \sum_{\alpha \in \mathcal{F}} \sum_{h.g=\alpha} Pr(X_1 = g_1, \ldots, X_n = g_n | Y = \beta)\ \theta^{Tr_{\mathbb{F}_q/\mathbb{F}_p}(\alpha)} \text{ si } \mathcal{F} = \mathbb{F}_{p^s} \end{aligned}$$

où θ est une racine primitive e-ième de l'unité.
Or, comme la variable aléatoire Y est indépendante de $h_1X_1 + \ldots + h_nX_n$ pour $h \neq 0$, on a

$$\begin{aligned} \sum_{h.g=\alpha} Pr(X_1 = g_1, \ldots, X_n = g_n | Y = \beta) &= Pr(h_1X_1 + \ldots + h_nX_n = \alpha | Y = \beta) \\ &= Pr(h_1X_1 + \ldots + h_nX_n = \alpha) \\ &= \sum_{h.g=\alpha} Pr(X_1 = g_1, \ldots, X_n = g_n) \end{aligned}$$

On en déduit donc que, pour h non nul, les coefficients de Fourier $\hat{x}_h$ et $\hat{y}_h$ sont égaux.
De plus, pour $h = 0$, on a trivialement $\hat{x}_0 = \sum_{g \in G} x_g = 1 = \sum_{g \in G} y_g = \hat{y}_0$. □

A l'instar de Gopalakrishnan et Stinson, je peux déduire de ce lemme une caractérisation des fonctions sans corrélation et des fonctions résilientes en termes matriciels.

Théorème 6.13 (caractérisation des fonctions résilientes à l'aide de matrices) *Soit $\mathcal{F}$ un alphabet fini à q éléments muni soit de la structure d'anneau $\mathbb{Z}_q$, soit de la structure de corps $\mathbb{F}_q$, E un ensemble fini quelconque et f une fonction de $\mathcal{F}^n$ dans E. Soit $N(u) = (\eta_{i,j})_{i,j \in \mathcal{F}}$ la matrice réelle définie par*

$$\eta_{i,j} = q^n Pr(u_1X_1 + \ldots + u_nX_n = i \text{ et } f(X) = j)$$

- *la fonction f est sans corrélation d'ordre t sur $\mathcal{F}$ si et seulement si, pour tout $u \in \mathcal{F}^n$ tel que $1 \leq w_H(u) \leq t$, toutes les lignes de la matrice $N(u)$ sont identiques.*
- *la fonction f est t-résiliente sur $\mathcal{F}$ si et seulement, pour tout $u \in \mathcal{F}^n$ tel que $1 \leq w_H(u) \leq t$, les éléments de la matrice $N(u)$ valent tous $\frac{q^{n-1}}{|E|}$.*

preuve : Soit u un élément de $\mathcal{F}^n$ de poids de Hamming $1 \leq w_H(u) \leq t$ et de support T. Par définition, la fonction f est sans corrélation d'ordre t si et seulement si sa sortie $f(X)$ est indépendante des entrées $(X_i)_{i \in T}$. Cela équivaut, d'après le lemme de la combinaison linéaire, à

$$\forall i, j \;\; \eta_{i,j} = q^n Pr(u_1 X_1 + \ldots + u_n X_n = i) Pr(f(X) = j)$$

Toutes les variables d'entrée de la fonction étant indépendantes et uniformément distribuées, on a d'autre part $Pr(u_1 X_1 + \ldots + u_n X_n = i) = \frac{1}{q}$. Une condition nécessaire et suffisante pour que f soit sans corrélation d'ordre t est donc que

$$\forall i, \;\; \eta_{i,j} = q^{n-1} Pr(f(X) = j)$$

La fonction f est, en plus, équilibrée si et seulement si on a $\forall j \in E, \; Pr(f(X) = j) = 1/|E|$. Tous les éléments de matrice $N(u)$ sont alors égaux à $q^{n-1}/|E|$.

□

5 Construction de fonctions résilientes à partir de codes

La caractérisation des fonctions résilientes par les tableaux orthogonaux fait apparaître un lien entre ces fonctions et les codes correcteurs d'erreurs.

5.1 Codes et tableaux orthogonaux

Ce lien provient directement du fait que les mots d'un code forment les lignes d'un tableau orthogonal dont la force est définie par un paramètre fondamental du code, sa *distance duale*. Il est en effet immédiat que le tableau formé par les mots d'un code linéaire $\mathcal{C}$ est un tableau orthogonal de force $d^\perp - 1$ où $d^\perp$ est la distance minimale du code dual $\mathcal{C}^\perp$. Ainsi énoncée, cette proposition n'a évidemment de sens que dans le cas d'un code additif, puisqu'un code non-additif n'a pas de dual. Toutefois, elle reste pertinente lorsque la distance duale $d^\perp$ est définie à partir de la *transformée de MacWilliams* de l'énumérateur de Hamming des distances du code $\mathcal{C}$.

Je reprends ici les différentes étapes de la démonstration de cette propriété remarquable, mise en évidence par Philippe Delsarte dans [Del72, Del73b], qui montre que les mots d'un code (additif ou non) sur un groupe abélien fini forment les lignes d'un tableau orthogonal de force $d^\perp - 1$.

Proposition 6.14 (transformation de MacWilliams) *Soient n et q deux entiers positifs, $q \geq 1$. Le* polynôme de Krawtchouk $P_k(x)$ *est le polynôme de degré k défini sur $\mathbb{Q}$ par :*

$$\forall\, 0 \leq k \leq n,\ \ P_k(x) = \sum_{i=0}^{k} (-1)^i (q-1)^{k-i} \binom{x}{i} \binom{n-x}{k-i}$$

La transformation de MacWilliams *est alors l'application*

$$A = (A_0, \ldots, A_n) \quad \mapsto \quad A' = (A'_0, \ldots, A'_n)$$

$$\text{où } \forall\, 0 \leq k \leq n,\ \ A'_k = \textstyle\sum_{i=0}^{n} A_i P_k(i)$$

La transformation de MacWilliams est bijective et vérifie, pour tout $(n+1)$-uplet A, $(A')' = q^n A$.

Elle peut également s'écrire sous la forme polynômiale suivante :

$$A'(X,Y) = A(X + (q-1)Y, X - Y)$$

où $A(X,Y) = \sum_{i=0}^{n} A_i X^{n-i} Y^i$ et $A'(X,Y) = \sum_{i=0}^{n} A'_i X^{n-i} Y^i$.

preuve : Il suffit d'écrire l'expression polynômiale :

$$A'(X,Y) = \sum_{i=0}^{n} A'_i X^{n-i} Y^i = \sum_{j=0}^{n} A_j (\sum_{i=0}^{n} P_i(j) X^{n-i} Y^i)$$

Comme $(1-Z)^j (1+(q-1)Z)^{n-j}$ est la fonction génératrice des polynômes de Krawtchouk au point i [Sze59], on en déduit

$$\sum_{i=0}^{n} P_i(j) X^{n-i} Y^i = (X + (q-1)Y)^{n-j} (X-Y)^j$$

ce qui donne $A'(X,Y) = A(X + (q-1)Y, X - Y)$ □

On définit alors la distance duale d'un code à partir de la transformée de MacWilliams de sa distribution des distances.

Proposition 6.15 [Del72] *Soit $\mathcal{C}$ un code de longueur n et de taille M défini sur un groupe abélien $\mathcal{F}$ à q éléments. Soit $A = (A_0, \ldots, A_n)$ sa distribution des distances, c'est-à-dire*

$$\forall i,\ \ A_i = \frac{1}{M} |\{(x,y) \in \mathcal{C} \times \mathcal{C},\ d_H(x,y) = i\}|$$

et $A' = (A'_0, \ldots, A'_n)$ l'image de A par la transformation de MacWilliams. Alors, pour tout k, $0 \leq k \leq n$, $A'_k \geq 0$.

Définition 6.16 (Distance duale d'un code) *Soit $\mathcal{C}$ un code de longueur n et de taille M défini sur un groupe abélien $\mathcal{F}$ à q éléments. Soit $A = (A_0,\ldots, A_n)$ sa distribution des distances. On appelle* distance duale *du code $\mathcal{C}$, notée $d^\perp$, le plus petit indice $i > 0$ tel que $A'_i > 0$ où $A' = (A'_0, \ldots, A'_n)$ est l'image de $(A_0, \ldots, A_n)$ par la transformation de MacWilliams.*

C'est donc ce paramètre qui détermine la force du tableau orthogonal formé par les mots du code $\mathcal{C}$. J'utiliserai pour cette démonstration le lemme suivant :

Lemme 6.17 *Soit $\mathcal{F}$ un groupe abélien fini à q éléments et $X_k = \{x \in \mathcal{F}^n, w_H(x) = k\}$. Alors, pour tout $x \in X_i$, on a*

$$\sum_{y\in X_k} < x, y >= P_k(i)$$

où $P_k(x)$ est le polynôme de Krawtchouk défini à la proposition 6.14.

preuve : Soit $x \in \mathcal{F}^n$ de poids i et S_x son support. Alors on a

$$\begin{aligned}\sum_{y\in X_k} < x, y > &= \sum_{\substack{S\subset\{1,\ldots,n\}\\|S|=k}} \sum_{\substack{y\in\mathcal{F}^n\\ supp(y)=S}} < x, y > \\ &= \sum_{\substack{S\subset\{1,\ldots,n\}\\|S|=k}} \prod_{j\in S}\sum_{y_j\in\mathcal{F}\setminus\{0\}} < x_j, y_j >\end{aligned}$$

Or, d'après la proposition 6.7, on a

$$\sum_{y_j\in\mathcal{F}\setminus\{0\}} < x_j, y_j >= \begin{cases} q-1 & \text{si } x_j = 0\\ -1 & \text{sinon}\end{cases}$$

$$\text{D'où } \sum_{y\in X_k} < x, y >= \sum_{\substack{S\subset\{1,\ldots,n\}\\|S|=k}} (-1)^{|S_x\cap S|}(q-1)^{k-|S_x\cap S|}$$

Or, comme x est de poids i, il y a exactement $\binom{i}{j}\binom{n-i}{k-j}$ choix de S qui vérifient $|S_x \cap S| = j$. On en déduit donc

$$\sum_{y\in X_k} < x, y >= \sum_{j=0}^{k}(-1)^j(q-1)^{k-j}\binom{i}{j}\binom{n-i}{k-j} = P_k(i)$$

□

Théorème 6.18 [Del73b] *Soit $\mathcal{C}$ un code de longueur n et de taille M défini sur un groupe abélien fini $\mathcal{F}$. Le tableau $\mathcal{A}$ dont les lignes sont les mots du code $\mathcal{C}$ est un tableau orthogonal de taille M, à n contraintes et de force t sur $\mathcal{F}$ si et seulement si $1 \le t \le d^\perp - 1$, où $d^\perp$ est la distance duale de $\mathcal{C}$.*

preuve : On utilise ici la caractérisation des tableaux orthogonaux en termes de caractères donnée au théorème 6.10. Soit $y \in \mathcal{F}^n, w_H(y) = k$. On a alors, d'après le lemme précédent,

$$\begin{aligned}\sum_{x\in\mathcal{C}} < x, y > &= \sum_{i=0}^{n} \sum_{\substack{x\in\mathcal{C} \\ w_H(x)=i}} < x, y > \\ &= \sum_{i=0}^{n} A_i P_k(i) = A'_k\end{aligned}$$

Donc, par définition de la distance duale, on en déduit que

$$\forall y \in \mathcal{F}^n,\ 1 \le w_H(y) \le t,\ \sum_{x\in\mathcal{C}} < x, y >= 0 \text{ ssi } 1 \le t \le d^\perp - 1$$

□

Lorsque $\mathcal{C}$ est un code linéaire, on sait, depuis les travaux de MacWilliams [Mac63], que la distance duale définie par la transformée de MacWilliams est en réalité égale à la distance minimale du dual de $\mathcal{C}$. Cette propriété a été généralisée par Delsarte dans le cas où $\mathcal{C}$ est un *code additif*, c'est-à-dire un code dont les mots forment un sous-groupe du groupe abélien $(\mathcal{F}^n, +)$. On définit alors son dual de la façon suivante :

Définition 6.19 (dual d'un sous-groupe) *Soit H un sous-groupe du groupe abélien fini G. Son* dual *est le sous-groupe $H^\perp$ de G défini par*

$$H^\perp = \{y \in G,\ \forall x \in H, < x, y >= 1\}$$

On a alors la relation d'orthogonalité :

Proposition 6.20 *Soit H un sous-groupe du groupe abélien fini G et $H^\perp$ son dual. On a alors*

$$\sum_{x\in H} < x, y >= \begin{cases} |H| & \text{si } y \in H^\perp \\ 0 & \text{sinon} \end{cases}$$

Dans le cas où $\mathcal{C}$ est un code additif, sa distribution des distances est réduite à sa distribution de poids, c'est-à-dire le $(n+1)$-uplet $(A_0, \ldots, A_n)$ défini par, pour tout i, $A_i = |\{x \in \mathcal{C},\ w_H(x) = i\}|$. En utilisant les propriétés précédentes, on obtient alors le théorème suivant.

Théorème 6.21 (distance duale d'un code additif) [Mac63, Del72] *Soit $\mathcal{C}$ un code additif de longueur n et de taille M défini sur le groupe abélien $\mathcal{F}$ à q éléments, et $A(\mathcal{C})$ sa distribution de poids. Alors la distribution de poids $A(\mathcal{C}^\perp)$ de son dual vérifie*

$$A(\mathcal{C}^\perp) = \frac{1}{M} A'(\mathcal{C})$$

où $A'(\mathcal{C})$ est l'image de $A(\mathcal{C})$ par la transformation de MacWilliams.

preuve : Le code $\mathcal{C}$ étant un code additif, sa distribution des distances coïncide avec sa distribution de poids. On a donc, d'après le lemme 6.17,

$$\begin{aligned} A'_k(\mathcal{C}) &= \sum_{i=0}^{n} A_i(\mathcal{C}) P_k(i) \\ &= \sum_{i=0}^{n} \sum_{\substack{x \in \mathcal{C} \\ w_H(x)=i}} \sum_{\substack{y \in \mathcal{F}^n \\ w_H(y)=k}} < x, y > \\ &= \sum_{\substack{y \in \mathcal{F}^n \\ w_H(y)=k}} (\sum_{x \in \mathcal{C}} < x, y >) \end{aligned}$$

En utilisant la propriété précédente, on obtient alors

$$A'_k(\mathcal{C}) = M A_k(\mathcal{C}^\perp)$$

□

Corollaire 6.22 *La distance duale d'un code additif défini sur un groupe abélien fini est égale à la distance minimale de son dual.*

Exemple 6.23: Soit $\mathcal{C}$ le code linéaire de longueur 4 et de dimension 2 défini sur le groupe abélien $(\mathbb{Z}_4, +)$ par la matrice génératrice

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 3 \end{bmatrix}$$

Son dual, $\mathcal{C}^\perp$ est un code linéaire sur $\mathbb{Z}_4$ dont une matrice génératrice est

$$G' = \begin{bmatrix} 3 & 3 & 1 & 0 \\ 3 & 1 & 0 & 1 \end{bmatrix}$$

La distance minimale de $\mathcal{C}^\perp$ est donc égale à 2, ce qui peut être vérifié à l'aide de la transformée de MacWilliams de l'énumérateur des poids de $\mathcal{C}$: on a, en effet,

$$A_{\mathcal{C}}(X, Y) = X^4 + X^2Y^2 + 10XY^3 + 4Y^4$$

ce qui donne

$$A_{\mathcal{C}^\perp}(X, Y) = X^4 + X^2Y^2 + 10XY^3 + 4Y^4$$

Le tableau $\mathcal{A}$ formé par les mots de $\mathcal{C}$ est donc un tableau orthogonal de taille 16, à 4 contraintes, de force 1 et d'indice 4 sur $\mathbb{Z}_4$.

$$\mathcal{A} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 2 & 0 & 2 & 2 \\ 3 & 0 & 3 & 3 \\ 0 & 1 & 1 & 3 \\ 1 & 1 & 2 & 0 \\ 2 & 1 & 3 & 1 \\ 3 & 1 & 0 & 2 \\ 0 & 2 & 2 & 2 \\ 1 & 2 & 3 & 3 \\ 2 & 2 & 0 & 0 \\ 3 & 2 & 1 & 1 \\ 0 & 3 & 3 & 1 \\ 1 & 3 & 0 & 2 \\ 2 & 3 & 1 & 3 \\ 3 & 3 & 2 & 0 \end{bmatrix}$$

Considérons maintenant le code $\phi(\mathcal{C})$, image du code $\mathcal{C}$ par la fonction de Gray ϕ :

$$\begin{array}{rccc} \phi : & \mathbb{Z}_4 & \rightarrow & \mathbb{F}_2 \times \mathbb{F}_2 \\ & 0 & \mapsto & 00 \\ & 1 & \mapsto & 01 \\ & 2 & \mapsto & 11 \\ & 3 & \mapsto & 10 \end{array}$$

Le tableau $\phi(\mathcal{A})$ formé par les mots de $\phi(\mathcal{C})$ est donc :

$$\phi(\mathcal{A}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Le code $\phi(\mathcal{C})$ n'est pas un code additif : par exemple, la somme binaire des mots de code 00010110 et 10001010 n'est pas un mot du code. Son énumérateur des distances est

$$A_{\phi(\mathcal{C})} = X^8 + 4X^5Y^3 + 5X^4Y^4 + 4X^3Y^5 + 2X^2Y^6$$

dont la transformée de MacWilliams vaut

$$A'_{\phi(\mathcal{C})} = 16[X^8 + 4X^5Y^3 + 5X^4Y^4 + 4X^3Y^5 + 2X^2Y^6]$$

Le tableau $\phi(\mathcal{A})$ est donc un tableau orthogonal de taille 16, à 8 contraintes, de force 2 et d'indice 4 sur $\mathbb{F}_2$.

Exemple 6.24: Soit $\mathcal{C}_{a,b}$ le code défini sur le groupe cyclique $(\mathbb{Z}_q, +)$, $q \geq 2$, par la fonction d'encodage

$$\begin{array}{rccc} f_{a,b} : & \mathbb{Z}_q^2 & \to & \mathbb{Z}_q^4 \\ & (x_1, x_2) & \mapsto & (x_1, x_2, x_1 + ax_2, x_1 + bx_2) \end{array}$$

où a et b sont des éléments non nuls de $\mathbb{Z}_q$.

Le code $\mathcal{C}_{a,b}$ est un code additif. Par conséquent, la force du tableau orthogonal $\mathcal{A}_{a,b}$ constitué des mots de $\mathcal{C}_{a,b}$ est déterminée par la distance minimale $d^\perp$ de son dual $\mathcal{C}_{a,b}^\perp$. La borne de Singleton impose $d^\perp \leq 3$. Le tableau $\mathcal{A}_{a,b}$ est donc un tableau orthogonal de force maximale 2 si et seulement si $\mathcal{C}_{a,b}^\perp$ ne contient aucun mot de poids inférieur ou égal à 2, c'est-à-dire, d'après la définition 6.19,

$$\forall y \in \mathbb{Z}_q^4,\ 1 \leq w_H(y) \leq 2,\ \exists x \in \mathcal{C}_{a,b},\ \ \theta^{xy} \neq 1$$

où θ est une racine primitive q-ième de l'unité. Cela équivaut à dire que, pour tout couple (α, β) d'éléments non nuls de $\mathbb{Z}_q$, on a

$$\begin{array}{ll} \exists (x_1, x_2) \in \mathbb{Z}_q, & x_1(\alpha + \beta) + x_2(a\beta) \neq 0 \\ \exists (x_1, x_2) \in \mathbb{Z}_q, & x_1(\alpha + \beta) + x_2(b\beta) \neq 0 \\ \exists (x_1, x_2) \in \mathbb{Z}_q, & x_1(\alpha + \beta) + x_2(a\alpha + b\beta) \neq 0 \end{array}$$

c'est-à-dire

$$\forall \beta \in \mathbb{Z}_q^\star,\ a\beta \neq 0 \text{ et } b\beta \neq 0 \text{ et } (a-b)\beta \neq 0$$

On en déduit donc que $\mathcal{A}_{a,b}$ est un tableau orthogonal de force 2 sur $\mathbb{Z}_q$ si et seulement si a, b et $a - b$ ne sont pas des diviseurs de zéro.

Ainsi le tableau formé des quadruplets $(x_1, x_2, x_1 + 4x_2, x_1 + 11x_2)$ est un tableau orthogonal de force 2 sur $\mathbb{Z}_{15}$.

Cela implique également que la force d'un tel tableau orthogonal $\mathcal{A}_{a,b}$ défini sur $\mathbb{Z}_q$ est strictement inférieure à 2 lorsque q est pair. On retrouve ainsi directement un résultat donné par S. Vaudenay [Vau95a, page 100].

5.2 Codes et fonctions résilientes

Une fonction t-résiliente de $\mathcal{F}^n$ dans $\mathcal{F}^\ell$, où $\mathcal{F}$ est un groupe abélien à q éléments, est caractérisée par un ensemble de q^ℓ tableaux orthogonaux de force t et de même taille formant une partition de $\mathcal{F}^n$. D'après les résultats de Ph. Delsarte, une telle partition peut être obtenue à partir des q^ℓ cosets d'un code linéaire dont le dual a pour distance minimale $t+1$.

Ce résultat prouvé dans [Sti93] dans le cas d'un code sur un corps fini, peut être énoncé plus généralement pour tout code linéaire sur un anneau fini :

Proposition 6.25 *Soit $\mathcal{F}$ un anneau fini, $\mathcal{C}$ un code linéaire de longueur n et de dimension k sur $\mathcal{F}$,* i.e. *un sous-module libre de $\mathcal{F}^n$ de dimension k, et H une matrice de parité de $\mathcal{C}$. La fonction syndrome associée à $\mathcal{C}$ définie par*

$$\begin{array}{rccc} f: & \mathcal{F}^n & \to & \mathcal{F}^{n-k} \\ & x & \mapsto & x^t H \end{array}$$

est une fonction $(d^\perp - 1)$-résiliente sur $\mathcal{F}$ où $d^\perp$ est la distance minimale du dual de $\mathcal{C}$.

preuve : Soit $s \in \mathcal{F}^{n-k}$. L'ensemble M_s des mots de syndrome s est un code translaté de $\mathcal{C}$. Sa distribution des distances étant la même que celle de $\mathcal{C}$, sa distance duale est égale à $d^\perp$. D'après le théorème 6.18, les mots de M_s forment un tableau orthogonal de taille q^k, à n contraintes et de force $d^\perp - 1$ sur $\mathcal{F}$. La fonction f est donc $(d^\perp - 1)$-résiliente. □

Toute fonction linéaire $f : \mathcal{F}^n \to \mathcal{F}^{n-k}$, t-résiliente s'identifie donc à la fonction syndrome d'un code $[n, k]$ de distance duale $t+1$.

La partition de $\mathcal{F}^n$ par les cosets n'existe malheureusement pas dans le cas d'un code non-linéaire. Toutefois, Massey et Stinson [SM95] ont montré qu'il était possible de la généraliser dans le cas d'un *code systématique*, c'est-à-dire un code admettant au moins un ensemble d'information : il est en effet possible de partitionner l'espace par les translatés $\mathcal{C} + \bar{e}$ d'un code systématique $\mathcal{C}$, quand $\bar{e}$ parcourt l'ensemble des vecteurs de $\mathcal{F}^n$ s'annulant sur un ensemble d'information du code. On obtient alors le résultat suivant.

Théorème 6.26 [SM95] *Soit $\mathcal{F}$ un groupe abélien à q éléments, $\mathcal{C}$ un code systématique de longueur n et de taille q^k sur $\mathcal{F}$, et I un ensemble d'information de $\mathcal{C}$. La fonction f définie par*

$$\begin{array}{rccc} f: & \mathcal{F}^n & \to & \mathcal{F}^{n-k} \\ & x & \mapsto & e \text{ ssi } x \in \mathcal{C} + \bar{e} \end{array}$$

où $\bar{e}$ est le vecteur de $\mathcal{F}^n$ s'annulant sur I et dont la restriction à $\{1, \ldots, n\} \setminus I$ vaut e, est une fonction $(d^\perp - 1)$-résiliente où $d^\perp$ est la distance duale du code $\mathcal{C}$.

preuve : Soit $e \in \mathcal{F}^{n-k}$. La distribution des distances du code translaté $\mathcal{C} + \bar{e}$ est égale à celle de $\mathcal{C}$. Le tableau formé par les éléments de $\mathcal{C} + \bar{e}$ est donc un tableau orthogonal de taille q^k, à n contraintes et de force $d^\perp - 1$ sur $\mathcal{F}$, d'après le théorème 6.18.

De plus, tous les $(\mathcal{C} + \bar{e})_{e \in \mathcal{F}^{n-k}}$ sont disjoints : en effet, l'existence de deux mots de code c_1 et c_2, et de deux vecteurs distincts e_1 et e_2 de $\mathcal{F}^{n-k}$ vérifiant $c_1 + \bar{e}_1 = c_2 + \bar{e}_2$ impliquerait que c_1 et c_2 prennent la même valeur sur l'ensemble d'information I, et donc que $c_1 = c_2$.

La réunion des $(\mathcal{C} + \bar{e})_{e \in \mathcal{F}^{n-k}}$ contient donc tous les q^n vecteurs de l'espace. D'après la proposition 6.4, f est par conséquent une fonction $(d^\perp - 1)$-résiliente sur $\mathcal{F}$. □

Ce résultat a notamment permis à Massey et Stinson de construire des fonctions résilientes à partir des codes binaires non-linéaires de Kerdock et de Preparata.

> **Exemple 6.27: [SM95]** Soit r un entier impair supérieur à 3. Le code de Kerdock, $\mathcal{K}(r+1)$ est un code binaire non-linéaire systématique de longueur 2^{r+1}, de taille 2^{2r+2} et de distance duale 6. La construction précédente aboutit donc l'existence de la famille de fonctions 5-résilientes
>
> $$f_{\mathcal{K}(r+1)} : \mathbb{F}_2^{2^{r+1}} \to \mathbb{F}_2^{2^{r+1}-2r-2}$$
>
> De plus, comme tout code linéaire $[2^{r+1}, 2^{r+1} - 2r - 2]$ est de distance minimale au plus 5 [GS72, BT93], toute fonction binaire à composantes linéaires ayant les mêmes paramètres que $f_{\mathcal{K}(r+1)}$ est au mieux 4-résiliente.
>
> Massey et Stinson ont, avec cet exemple, réfuté la conjecture formulée dans [BBR88, page 227], selon laquelle l'existence d'une fonction t-résiliente $f : \mathbb{F}_2^n \to \mathbb{F}_2^\ell$ impliquerait l'existence pour les mêmes paramètres d'une fonction t-résiliente à composantes linéaires.

6 Bornes sur l'ordre de résilience d'une fonction

Le lien entre les codes et les tableaux orthogonaux permet d'établir des bornes sur la taille maximale d'un tableau orthogonal dont le nombre de contraintes et la force sont fixés et, par conséquent, sur l'ordre de résilience d'une fonction. On peut alors déduire des bornes supérieures sur l'ordre de résilience d'une fonction à partir de bornes inférieures sur la taille des tableaux orthogonaux correspondants.

6.1 Bornes classiques

Les deux premières bornes évoquées ici sont des bornes classiques qui ont été établies dans les premiers articles qui ont défini la structure de tableau orthogonal.

Proposition 6.28 (borne de Rao) [Rao47] *Soit $\mathcal{F}$ un alphabet à q éléments et $\mathcal{A}$ un tableau orthogonal de taille M, à n contraintes et de force t sur $\mathcal{F}$. Alors*

$$\begin{array}{lcll} M \geq & 1+\sum_{i=1}^{t/2}\binom{n}{i}(q-1)^i & & \text{si } t \text{ est pair} \\ M \geq & 1+\binom{n-1}{(t-1)/2}(q-1)^{(t+1)/2}+\sum_{i=1}^{(t-1)/2}\binom{n}{i}(q-1)^i & & \text{si } t \text{ est impair} \end{array}$$

La borne de Bush, démontrée en 1952, ne concerne que les tableaux orthogonaux d'indice 1, et par conséquent les fonctions $f : \mathcal{F}^n \to \mathcal{F}^\ell$ dont l'ordre de résilience est $n - \ell$.

Proposition 6.29 (borne de Bush) [Bus52] *Soit $\mathcal{F}$ un alphabet à q éléments et $\mathcal{A}$ un tableau orthogonal d'indice 1, de taille q^t, à n contraintes et de force t sur $\mathcal{F}$. Alors*

$$\begin{array}{lcl} n \leq & q+t-1 & \text{si } q \geq t \text{ et } q \text{ pair} \\ n \leq & q+t-2 & \text{si } q \geq t \geq 3 \text{ et } q \text{ impair} \\ n \leq & t+1 & \text{si } q \leq t \end{array}$$

6.2 Borne de la programmation linéaire

La meilleure borne connue sur la taille d'un code est la *borne de la programmation linéaire*, établie par Ph. Delsarte. En utilisant les propriétés de la distance duale d'un code rappelées précédemment, on peut l'appliquer aux tableaux orthogonaux.

Proposition 6.30 (borne de la programmation linéaire) [Del72] *Soit $\mathcal{F}$ un groupe abélien à q éléments et $\mathcal{A}$ un tableau orthogonal de taille M, à n contraintes et de force t sur $\mathcal{F}$. Alors*

$$M \geq q^n \frac{1}{1+U^\star}$$

où $U^\star$ est la valeur de la fonction de coût $U = \sum_{i=t+1}^{n} x_i$ obtenue pour une solution du programme linéaire suivant :

$$(PL) : \begin{cases} \text{maximiser } U = \sum_{i=t+1}^{n} x_i \\ \text{sous les contraintes} \\ \forall\ 0 \leq k \leq n,\ \ \sum_{i=t+1}^{n} x_i P_k(i) \geq -\binom{n}{k}(q-1)^k \\ \text{et } \forall t+1 \leq i \leq n,\ \ x_i \geq 0 \end{cases}$$

preuve : Soit $\mathcal{C}$ le code de longueur n et de taille M sur $\mathcal{F}$ formé par les lignes de $\mathcal{A}$, et $A = (A_0, \ldots, A_n)$ sa distribution des distances. Soit $(B_0, \ldots, B_n) = \frac{1}{M}(A'_0, \ldots, A'_n)$ où A' est la transformée de MacWilliams de A. Le théorème 6.18

implique que, $\forall 1 \leq i \leq t,\ B_i = 0$. La transformation de MacWilliams étant bijective, on peut écrire

$$\begin{aligned} A_i &= \frac{M}{q^n} \sum_{i=0}^{n} B_i P_K(i) \\ &= \frac{M}{q^n} (B_0 P_k(0) + \sum_{i=t+1}^{n} B_i P_K(i)) \end{aligned}$$

Or, on déduit de l'expression de la fonction génératrice des polynômes de Krawtchouk que $P_k(0) = (q-1)^k \binom{n}{k}$. De plus, $A'_0 = \sum_{i=0}^{n} A_i P_0(i) = \sum_{i=0}^{n} A_i = M$. On a donc

$$\frac{q^n}{M} A_i = (q-1)^k \binom{n}{k} + \sum_{i=t+1}^{n} B_i P_k(i) \geq 0$$

Comme la transformée de MacWilliams de la distribution des distances d'un code est positive, $(B_{t+1}, \ldots, B_n)$ vérifie les contraintes du problème (PL). Par conséquent, $\sum_{i=0}^{n} B_i \leq 1 + U^\star$.
De plus $\sum_{i=0}^{n} B_i = \frac{1}{M} \sum_{k=0}^{n} A_k \sum_{i=0}^{n} P_i(k)$. Comme l'expression de la fonction génératrice des polynômes de Krawtchouk implique que $\sum_{i=0}^{n} P_i(k) = \delta_{k,0} q^n$, où $\delta_{i,j}$ est le symbole de Kronecker, on en déduit

$$\sum_{i=0}^{n} B_i = \frac{q^n}{M}$$

On obtient ainsi pour M la borne inférieure énoncée. □

La valeur $U^\star$ donnée par une solution du programme (PL) peut être également obtenue par le programme dual (DPL) suivant :

Proposition 6.31 (borne de la programmation linéaire, programme dual) [Del72] *Soit $\mathcal{F}$ un groupe abélien à q éléments et $\mathcal{A}$ un tableau orthogonal de taille M, à n contraintes et de force t sur $\mathcal{F}$. Alors*

$$M \geq q^n \frac{1}{1 + V^\star}$$

où $V^\star$ est la fonction de coût obtenue pour une solution du programme linéaire suivant :

$$(DPL) : \begin{cases} \text{minimiser } V = \sum_{k=1}^{n} y_k (q-1)^k \binom{n}{k} \\ \text{sous les contraintes} \\ \forall\ t+1 \leq i \leq n,\ \ \sum_{k=1}^{n} y_k P_k(i) \leq -1 \\ \text{et } \forall\ 1 \leq i \leq n,\ \ y_i \geq 0 \end{cases}$$

6.3 Bornes explicites dérivées de la borne de la programmation linéaire

La borne de la programmation linéaire n'est toutefois pas très aisée à manier bien qu'elle puisse être calculée en un temps polynômial. Il est alors très utile de disposer de bornes explicites aussi proches que possible de la borne de la programmation linéaire. Ces bornes sont obtenues en calculant la fonction du coût du programme pour une solution particulière des contraintes. Aussi toutes les bornes supérieures sur la taille d'un code dérivées de la borne de la programmation linéaire peuvent-elles être adaptées aux tableaux orthogonaux.

Cette technique permet en particulier d'appliquer aux tableaux orthogonaux la borne de Plotkin [Ber68]. Ce résultat, démontré par Bierbrauer, Gopalakrishnan et Stinson [BGS94] dans le cas binaire, se déduit immédiatement de la démonstration donnée par Ph. Delsarte [Del72, théorème 14] qui fait dériver la borne de Plotkin de celle de la programmation linéaire. Nous donnons ici l'expression générale de cette borne pour un tableau orthogonal défini sur groupe abélien fini.

Proposition 6.32 (borne de Plotkin) *Soit $\mathcal{F}$ un groupe abélien à q éléments et $\mathcal{A}$ un tableau orthogonal de taille M, à n contraintes et de force t sur $\mathcal{F}$. Alors*

$$M \geq q^n \left(1 - \frac{(q-1)n}{q(t+1)}\right)$$

preuve : Considérons le n-uplet $(y_1, \ldots, y_n)$ défini par

$$y_1 = \frac{1}{q(t+1) - n(q-1)} \text{ et } \forall\, 2 \leq k \leq n,\ \ y_k = 0$$

Alors, dans le cas où $q(t+1) \geq n(q-1)$, $(y_1, \ldots, y_n)$ vérifie les contraintes du programme (DPL). En effet, pour tout $t+1 \leq i \leq n$,

$$\sum_{k=1}^{n} y_k P_k(i) = \frac{P_1(i)}{q(t+1) - n(q-1)} = \frac{n(q-1) - qi}{q(t+1) - n(q-1)} \leq -1$$

Donc, d'après la proposition précédente, on obtient

$$V^\star \leq \sum_{k=1}^{n} y_k (q-1)^k \binom{n}{k} = \sum_{k=1}^{n} y_k P_k(0) = \frac{n(q-1)}{q(t+1) - n(q-1)}$$

On en déduit donc la borne

$$M \geq q^n \left(1 - \frac{(q-1)n}{q(t+1)}\right)$$

Dans le cas où $q(t+1) < n(q-1)$, cette borne prend une valeur négative. Par conséquent, l'inégalité énoncée est toujours satisfaite. □

Cette borne est particulière intéressante dans le cas binaire puisque Bierbrauer, Gopalakrishnan et Stinson ont montré qu'elle était parfois aussi puis-

sante — c'est-à-dire qu'elle prend la même valeur — que la borne de la programmation linéaire.

Proposition 6.33 [BGS94] *Soit $\mathcal{A}$ un tableau orthogonal de taille M, à n contraintes et de force t sur $\mathbb{F}_2$. Si t est impair et $t+1 \leq n \leq 2t+1$, la borne de Plotkin*

$$M \geq 2^n \left(1 - \frac{n}{2(t+1)}\right)$$

est aussi puissante que la borne de la programmation linéaire.

Une seconde borne explicite a été établie par la même méthode dans le cas binaire par Bierbrauer, Gopalakrishnan et Stinson.

Proposition 6.34 (borne de Bierbrauer) [BGS94, page] *Soit $\mathcal{A}$ un tableau orthogonal de taille M, à n contraintes et de force t sur $\mathbb{F}_2$, avec t pair. Alors*

$$M \geq 2^n \left(1 - \frac{n+1}{2(t+2)}\right)$$

preuve : Considérons le n-uplet $(y_1, \ldots, y_n)$ défini par

$$y_1 = y_n = \frac{1}{2(t+1) - n + 1} \text{ et, } \forall\, 1 < k < n, \;\; y_k = 0$$

Dans le cas où $2t - n + 3 > 0$, ce n-uplet vérifie les contraintes du programme (DPL). En effet, pour tout $t + 1 < i \leq n$,

$$\sum_{i=1}^{n} y_k P_k(i) = \frac{n - 2i + (-1)}{2(t+1) - n + 1} \leq -1$$

et, pour $i = t + 1$, cette inégalité n'est vraie que quand t est pair.

On en déduit donc

$$V^\star \leq \sum_{k=1}^{n} y_k P_k(0) = \frac{n+1}{2(t+1) - n + 1}$$

et la borne énoncée sur la taille du tableau orthogonal. Dans le cas où $2t - n + 3 \leq 0$, la proposition est encore vraie puisque la borne prend une valeur négative. □

Remarque 6.35 *On peut généraliser cette borne au cas q-aire, comme nous l'avons fait pour la borne de Plotkin. On obtient alors*

$$M > q^n \left(1 - \frac{(q-1)^n + (q-1)n}{q(t+1) - (q-1)^{n-t-2} + q + (q-1)^n}\right)$$

Toutefois, cette borne n'a d'intérêt que pour $q = 2$ puisque, pour $q > 2$, elle est toujours inférieure à la borne de Plotkin.

A l'instar de la borne de Plotkin, cette borne est aussi puissante que celle de la programmation linéaire pour certains tableaux orthogonaux définis sur $\mathbb{F}_2$.

Proposition 6.36 [BGS94] *Soit $\mathcal{A}$ un tableau orthogonal de taille M, à n contraintes et de force t sur $\mathbb{F}_2$. Si t est pair et $t+1 \leq n \leq 2t+2$, la borne de Bierbrauer*

$$M \geq 2^n \left(1 - \frac{n+1}{2(t+2)}\right)$$

est aussi puissante que la borne de la programmation linéaire.

6.4 Tables donnant l'indice minimal d'un tableau orthogonal et l'ordre de résilience maximal d'une fonction

Les différentes bornes sur la taille d'un tableau orthogonal peuvent être aisément comparées. La table récapitulative suivante donne ainsi la borne à utiliser pour obtenir la taille minimale d'un tableau orthogonal en fonction de sa force dans le cas binaire.

force t	taille M
$\lceil \frac{n-2}{2} \rceil \leq t \leq n-1$, t pair	borne de Bierbrauer $M \geq 2^n \left(1 - \frac{n+1}{2(t+2)}\right)$
$\lceil \frac{n-1}{2} \rceil \leq t \leq n-1$, t impair	borne de Plotkin $M \geq 2^n \left(1 - \frac{n}{2(t+1)}\right)$
sinon	borne de la programmation linéaire

TAB. 6.1 –: *Table récapitulative des bornes sur la taille d'un tableau orthogonal binaire*

A partir de ces bornes, on peut alors dresser une table donnant une borne inférieure, λ^-_{min}, sur le plus petit indice admissible pour un tableau orthogonal défini sur un groupe abélien à q éléments, dont le nombre de contraintes et la force sont fixés.

Je donne ici de telles tables pour $n \leq 20$ et $q = 2, 4$ et 8 (Tables 6.2, 6.3 et 6.4).

Il est également possible, à partir des tables de Brouwer et Verhoeff [BV93], de donner une borne supérieure pour λ_{min} lorsque q vaut 2 ou 4. Cette borne, λ^+_{min}, est l'indice minimal obtenu pour un tableau orthogonal composé des mots d'un code linéaire connu sur $\mathbb{F}_q$. Dans les tables suivantes, la valeur λ^+_{min} est notée entre parenthèses lorsqu'elle est différente de λ^-_{min}.

De manière similaire, je donne des tables sur l'ordre de résilience maximal, t_{max}, admissible pour une fonction $f : \mathcal{F}^n \to \mathcal{F}^\ell$, en fonction de n et ℓ, pour $q = 2, 4$ et 8 (Tables 6.5, 6.6 et 6.7).

n,t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2	1	1																	
3	1	1	1																
4	1	2	1	1															
5	1	2	2	1	1														
6	1	2	2	2	1	1													
7	1	2	2	3(4)	2	1	1												
8	1	3(4)	2	4	3(4)	2	1	1											
9	1	3(4)	3(4)	6(8)	4	3(4)	2	1	1										
10	1	3(4)	3 (4)	6(8)	6(8)	5(8)	3(4)	2	1	1									
11	1	3(4)	3(4)	6(8)	6(8)	8	5(8)	4	2	1	1								
12	1	4	3(4)	7(8)	6(8)	12(16)	8	6(8)	4	2	1	1							
13	1	4	4	8	7(8)	16	12(16)	10(16)	6(8)	4	2	1	1						
14	1	4	4	8	8	16	16	16	10(16)	6(8)	4	2	1	1					
15	1	4	4	8	8	16	16	26(32)	16	11(16)	6(8)	4	2	1	1				
16	1	5(8)	4	10(16)	8	21(32)	16	39(64)	26(32)	19(32)	11(16)	7(8)	4	2	1	1			
17	1	5(8)	5(8)	12(16)	10(16)	26(32)	21(32)	52(64)	39(64)	32	19(32)	12(16)	7(8)	4	2	1	1		
18	1	5(8)	5(8)	13(32)	12(16)	29(32)	26(32)	52(128)	52(64)	54(64)	32	21(32)	12(16)	7(8)	4	2	1	1	
19	1	5(8)	5(8)	14(32)	13(32)	29(32)	29(32)	52(128)	52(128)	86(128)	54(64)	37(64)	21(32)	12(16)	7(8)	4	2	1	1
20	1	6(8)	5(8)	15(32)	14(32)	29(32)	29(32)	64(128)	42(128)	128	86(128)	64	37(64)	22(32)	12(16)	7(8)	4	2	1

TAB. 6.2 –: *Bornes supérieure et inférieure sur l'indice minimal d'un tableau orthogonal binaire à n contraintes et de force t*

n,t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2	1	1																	
3	1	1	1																
4	1	1	1	1															
5	1	1	1	1	1														
6	1	2(4)	1	2(4)	1	1													
7	1	2(4)	2(4)	2(4)	2(4)	1	1												
8	1	2(4)	2(4)	2(4)	2(4)	3(4)	1	1											
9	1	2(4)	2(4)	2(4)	2(4)	3(4)	3(4)	1	1										
10	1	3(4)	2(4)	2(4)	2(4)	3(16)	4	3(4)	1	1 1									
11	1	3(4)	3(4)	3(4)	2(4)	4(16)	4(16)	6(16)	3(4)	1	1								
12	1	3(4)	3(4)	4(16)	3(16)	4(16)	5(16)	6(16)	7(16)	3(4)	1	1							
13	1	3(4)	3(4)	4(16)	4(16)	4(16)	5(16)	6(16)	7(16)	8(16)	3(4)	1	1						
14	1	3(4)	3(4)	4(16)	4(16)	4(16)	5(16)	7(16)	7(16)	12(16)	8(16)	4	1	1					
15	1	4	3(4)	5(16)	4(16)	4(16)	5(16)	7(64)	11(16)	12(16)	16	9(16)	4	1	1				
16	1	4	4	6(16)	5(16)	6(16)	5(16)	8(64)	11(64)	14(16)	16	20(64)	10(16)	4	1	1			
17	1	4	4	6(16)	6(16)	7(16)	6(16)	8(64)	11(64)	16(64)	16	20(64)	23(64)	10(16)	4	1	1		
18	1	4	4(16)	7(16)	6(16)	7(64)	7(16)	8(256)	11(64)	16(64)	26(64)	20(64)	37(64)	26(64)	10(16)	4	1	1	
19	1	4	4(16)	7(16)	7(16)	8(64)	7(64)	8(256)	11(64)	19(256)	26(256)	35(256)	37(64)	52(64)	28	11(16)	4	1	1
20	1	4	4(16)	8(16)	7(16)	9(64)	8(64)	9(256)	12(64)	19(256)	27(256)	35(256)	43(256)	52(64)	64	31(64)	11(16)	4	1

TAB. 6.3 –: *Bornes supérieure et inférieure sur l'indice minimal d'un tableau orthogonal à n contraintes et de force t, défini sur un groupe abélien à 4 éléments*

n,t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	1	1																		
3	1	1	1																	
4	1	1	1	1																
5	1	1	1	1	1															
6	1	1	1	1	1	1														
7	1	1	1	1	1	1	1													
8	1	1	1	1	1	1	1	1												
9	1	1	1	1	1	1	1	1	1											
10	1	2	1	1	1	1	1	2	1	1										
11	1	2	2	1	1	1	1	2	3	1	1									
12	1	2	2	2	1	1	1	2	3	3	1	1								
13	1	2	2	2	2	1	1	2	3	3	4	1	1							
14	1	2	2	2	2	2	1	2	3	3	4	4	1	1						
15	1	2	2	2	2	2	2	2	3	3	4	4	5	1	1					
16	1	2	2	2	2	2	2	2	3	3	4	4	5	5	1	1				
17	1	2	2	2	2	2	2	2	3	3	4	4	5	5	5	1	1			
18	1	3	2	2	2	2	2	2	3	3	4	4	5	5	8	5	1	1		
19	1	3	3	3	2	2	2	2	3	3	4	4	5	7	8	12	5	1	1	
20	1	3	3	3	3	2	2	2	3	3	4	4	5	7	10	12	15	6	1	1

TAB. 6.4 –: *Borne supérieure sur l'indice minimal d'un tableau orthogonal à n contraintes et de force t, défini sur un groupe abélien à 8 éléments*

n,ℓ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2	1																		
3	2	1																	
4	3	1	1																
5	4	2	1	1															
6	5	3	2	1	1														
7	6	3	3	2	1	1													
8	7	4	3	3	1	1	1												
9	8	5	3	3	2	1	1	1											
10	9	5	4	3	3	2	1	1	1	1									
11	10	6	5	4	3	3	2	1	1	1									
12	11	7	5	5	4*	3	3	2	1	1	1								
13	12	7	6	5	5*	4*	3	3	2	1	1	1							
14	13	8	7	6	5	5*	4*	3	3	2	1	1	1						
15	14	9	7	7	6	5	5*	4*	3	3	2	1	1	1					
16	15	9	7	7	7	5	5	5*	3	3	3	1	1	1	1				
17	16	10	8	7	7	6	5	5	4	3	3	2	1	1	1	1			
18	17	11	9	8(7)	7	7	6	5	5	4(3)	3	3	2	1	1	1	1		
19	18	11	9	9(8)	8(7)	7	7	6	5	5(4)	4(3)	3	3	2	1	1	1	1	
20	19	12	10	9	9(8)	8(7)	7	7	6	5	5(4)	4(3)	3	3	2	1	1	1	1

* bornes inférieures obtenues à partir du code non-linéaire de Nordstrom-Robinson

TAB. 6.5 –: *Bornes supérieure et inférieure sur l'ordre de résilience maximal d'une fonction $f : \mathbf{F}_2^n \to \mathbf{F}_2^\ell$*

n,ℓ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2	1																		
3	2	1																	
4	3	2	1																
5	4	3	2	1															
6	5	3	3	1	1														
7	6	4	3	2	1	1													
8	7	5	4	3	2	1	1												
9	8	6	5	4	3	2	1	1											
10	9	7	6(5)	5	4	3	2	1	1										
11	10	7	7(6)	6(5)	5	4	3	2	1	1									
12	11	8	7	6	6(5)	5	4(3)	3	2	1	1								
13	12	9	8	7	6	6(5)	5(4)	4(3)	3	2	1	1							
14	13	10	9	8	7	6	6(5)	5(4)	4(3)	3	2	1	1						
15	14	11	10	9	8(7)	7	6	6(5)	5(4)	3	3	2	1	1					
16	15	11	11	10	9(8)	8(7)	7	6	5	4	3	3	2	1	1				
17	16	12	11	11	10(9)	9(8)	8(7)	7	6	5	4	3	3	2	1	1			
18	17	13	12	11	10(9)	10(9)	9(7)	8(7)	7	6(5)	5	4	3	3(2)	2	1	1		
19	18	14	13	12(11)	11(10)	10(9)	9(8)	9(7)	8(7)	7(6)	6(5)	5	4	3	3(2)	2	1	1	
20	19	15	14	13(12)	12(11)	11(10)	10(9)	9(8)	9(7)	8(7)	7(6)	6(5)	5	4	3	3(2)	2	1	1

TAB. 6.6 –: *Bornes supérieure et inférieure sur l'ordre de résilience maximal d'une fonction* $f : \mathbf{F}_4^n \to \mathbf{F}_4^\ell$

n,ℓ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2	1																		
3	2	1																	
4	3	2	1																
5	4	3	2	1															
6	5	4	3	2	1														
7	6	5	4	3	2	1													
8	7	6	5	4	3	2	1												
9	8	7	6	5	4	3	2	1											
10	9	7	7	6	5	4	3	1	1	1									
11	10	8	7	7	6	5	4	2	1	1									
12	11	9	8	7	7	6	5	3	2	1	1								
13	12	10	9	8	7	7	6	4	3	2	1	1							
14	13	11	10	9	8	7	7	5	4	3	2	1	1						
15	14	12	11	10	9	8	7	6	5	4	3	2	1	1					
16	15	13	12	11	10	9	8	7	6	5	4	3	2	1	1				
17	16	14	13	12	11	10	9	8	7	6	5	4	3	2	1	1			
18	17	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	1		
19	18	15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	1	
20	19	16	15	14	14	13	12	11	10	9	8	7	6	5	4	3	2	1	1

TAB. 6.7 –: *Borne supérieure sur l'ordre de résilience maximal d'une fonction* $f : \mathbf{F}_8^n \to \mathbf{F}_8^\ell$

Chapitre 7

Ordre de non-linéarité d'une fonction t-résiliente

L'intérêt premier des fonctions sans-corrélation, mis en valeur par Siegenthaler, est qu'elles constituent une famille de fonctions appropriées pour combiner les sorties de plusieurs registres à décalage à rétroaction linéaire : elles permettent en effet de construire des générateurs pseudo-aléatoires résistant aux attaques par corrélation, dans l'optique d'un chiffrement à flot. La seule propriété de non-corrélation n'est toutefois pas suffisante pour assurer la qualité cryptographique du générateur pseudo-aléatoire correspondant. Ainsi, l'éventualité d'une attaque utilisant l'algorithme de Berlekamp-Massey impose, dans le cas booléen, que la fonction de combinaison soit non-linéaire. Plus généralement, les travaux de Herlestam [Her86] et ceux de Brynielsson [Bry86] ont montré que, sous certaines conditions, une haute non-linéarité de la forme algébrique normale de la fonction combinant plusieurs registres à décalage q-aires induit une complexité linéaire élevée du générateur. Malheureusement, cette condition de haute non-linéarité s'avère souvent incompatible avec un ordre de non-corrélation élevé : Siegenthaler a en effet mis en lumière [Sie85], dans le cas de fonctions booléennes, l'existence d'un compromis entre l'ordre de non-linéarité et l'ordre de non-corrélation d'une fonction. Je généralise ici ce résultat aux fonctions définies sur un corps fini quelconque en exhibant une borne sur l'ordre de non-linéarité des fonctions sans-corrélation d'ordre t (et t-résilientes) q-aires.

Après avoir défini la forme algébrique normale d'une fonction de $\mathbb{F}_q^n$ dans $\mathbb{F}_q^\ell$, je rappellerai les principaux résultats sur la combinaison des registres à décalage à rétroaction linéaire. Il apparaîtra alors que l'assemblage de plusieurs registres à décalage à rétroaction linéaire q-aires par une fonction dont la forme algébrique normale est hautement non-linéaire peut conduire, lorsque les différents polynômes de rétroaction sont choisis avec précaution, à un générateur pseudo-aléatoire de complexité linéaire maximale. Toutefois je montrerai qu'à l'instar du cas booléen, il existe un compromis entre le degré de la forme algébrique algébrique normale d'une fonction définie sur $\mathbb{F}_q$ et son ordre de non-

corrélation. J'exhiberai entre autres une inégalité, similaire à celle de Siegenthaler, qui régit l'ordre de non-linéarité des fonctions sans-corrélation définies sur $\mathbb{F}_q$. Je construirai ensuite une famille de fonctions résilientes dont l'ordre de non-linéarité est optimal ; ces fonctions sont donc particilièrement appropriées pour combiner des registres à décalage q-aires. Je donnerai enfin un résultat semblable sur l'ordre de non-linéarité des fonctions q-aires sans-corrélation sur $\mathbb{F}_{q^k}$.

1 Forme algébrique normale d'une fonction sans corrélation

L'expression d'une fonction par sa table des valeurs n'est évidemment pas très maniable : il est difficile d'en déduire certaines propriétés essentielles pour les applications cryptographiques, tel l'ordre de non-linéarité d'une fonction booléenne qui est un paramètre fondamental, notamment lorsqu'on l'utilise pour combiner des registres à décalage à rétroaction linéaire [Rue86]. Dans ce cas, ce paramètre est directement déterminé par la *forme algébrique normale* de la fonction, puisqu'il correspond au degré du polynôme à coefficients binaires correspondant. Je rappelle ici que l'on peut également définir une expression polynômiale pour décrire toute fonction $f : \mathbb{F}_q^n \to \mathbb{F}_q^\ell$. J'identifierai par la suite l'espace vectoriel $\mathbb{F}_q^\ell$ au corps $\mathbb{F}_{q^\ell}$.

Notation 7.1 *Soit $\mathcal{M}_n$ l'algèbre $\mathbb{F}_{q^\ell}[x_1, \ldots, x_n]/(x_1^q - x_1, \ldots, x_n^q - x_n)$. Pour un polynôme θ de cette algèbre, considéré comme le polynôme à n indéterminées de plus petit degré dans sa classe, $deg_{x_i}(\theta)$ désigne le degré de θ en x_i.*

J'utiliserai alors le lemme suivant :

Lemme 7.2 *Soit θ un élément de $\mathcal{M}_n$, considéré comme une fonction polynôme définie sur $\mathbb{F}_q^n$. Si, pour tout $x \in \mathbb{F}_q^n$, $\theta(x) = 0$, alors tous les coefficients de θ sont nuls.*

preuve : Montrons cette propriété par récurrence sur le nombre d'indéterminées n.

- $n = 1$. Soit d le degré du polynôme θ. Si $d = 0$, la propriété est triviale. Pour $d \geq 1$, la condition $\theta(x) = 0$ pour tous les x de $\mathbb{F}_q$ s'exprime sous forme d'un système de q équations à d inconnues avec $d \leq q$ puisque, par définition, le degré d'un polynôme de $\mathcal{M}_1$ est inférieur ou égal à q. L'unique solution de ce système correspond donc au cas où tous les coefficients de θ sont nuls.

- Considérons maintenant un polynôme θ à n indéterminées dans $\mathcal{M}_n$. On peut décomposer $\theta(x_1, \ldots, x_n)$ sous la forme

$$\theta(x_1, \ldots, x_n) = a(x_2, \ldots, x_n)\gamma(x_1) + b(x_2, \ldots, x_n)$$

où γ est un polynôme de $\mathcal{M}_1$ dont le terme constant est nul. Pour $x_1 = 0$, on a donc $\theta(x_1, \ldots, x_n) = b(x_2, \ldots, x_n)$. Comme le polynôme b s'annule sur $\mathbb{F}_q^{n-1}$, tous ses coefficients sont nuls, par l'hypothèse de récurrence. S'il existe un élément $x_1^\star$ de $\mathbb{F}_q$ tel que $\gamma(x_1^\star) \neq 0$, alors la fonction polynôme à $n-1$ variables qui à tout $(n-1)$-uplet $(x_2, \ldots, x_n)$ associe la valeur $\theta(x_1^\star, x_2, \ldots, x_n)$ est identiquement nulle. Donc tous les coefficients de a sont nuls. Dans le cas contraire, la fonction polynôme γ s'annule sur $\mathbb{F}_q$ tout entier ; tous les coefficients de θ sont donc nuls.

□

Théorème 7.3 (Forme algébrique normale) *Pour toute fonction $f : \mathbb{F}_q^n \to \mathbb{F}_{q^\ell}$, il existe une unique fonction polynôme θ de $\mathcal{M}_n$ telle que, pour tout x de $\mathbb{F}_q^n$, $f(x) = \theta(x)$. Ce polynôme est appelé* forme algébrique normale *de la fonction f.*

preuve : Je donne une preuve constructive, par récurrence sur n.

– $n = 1$. Soit $L_\alpha(x)$ le polynôme d'interpolation de Lagrange dans l'algèbre $\mathcal{M}_1$ défini par

$$L_\alpha(x) = \prod_{\substack{\beta \in \mathbb{F}_q \\ \beta \neq \alpha}} (x - \beta)$$

Par construction, on a

$$\forall \beta \neq \alpha, L_\alpha(\beta) = 0 \text{ et } L_\alpha(\alpha) = \prod_{\gamma \in \mathbb{F}_q^*} \gamma = -1$$

Considérons maintenant le polynôme θ défini par

$$\theta(x) = \sum_{\alpha \in \mathbb{F}_q} -f(\alpha) L_\alpha(x)$$

Il s'agit d'un polynôme sur $\mathbb{F}_q$ qui vérifie $\theta(x) = f(x)$ pour tout x de $\mathbb{F}_q$. Soit θ' un autre polynôme de $\mathcal{M}_1$ vérifiant $\theta'(x) = f(x)$ pour tout x. Le polynôme $\theta' - \theta$ s'annule sur $\mathbb{F}_q$; il est donc identiquement nul d'après le lemme précédent.

– Soient $x_1^*, \ldots, x_{n-1}^*$ $n-1$ valeurs fixées dans $\mathbb{F}_q$. Nous venons de montrer que la fonction $x_n \mapsto f(x_1^*, \ldots, x_{n-1}^*, x_n)$ est réalisée par une unique fonction polynôme de $\mathcal{M}_1$. On peut donc écrire

$$f(x_1^*, \ldots, x_{n-1}^*, x_n) = \sum_{i=0}^{q-1} a_i(x_1^*, \ldots, x_{n-1}^*) x_n^i$$

Comme chaque a_i est une fonction de $\mathbb{F}_q^{n-1}$ dans $\mathbb{F}_{q^\ell}$, on peut l'identifier à une fonction polynôme de $\mathcal{M}_{n-1}$. Il existe donc un polynôme θ de $\mathcal{M}_n$ tel que, pour tout x de $\mathbb{F}_q^n$, $\theta(x) = f(x)$. En outre, le lemme précédent induit l'unicité de θ.

□

Par définition, la forme algébrique normale d'une fonction à n variables définie sur $\mathbb{F}_q$ est donc de degré au plus $q-1$ en chacune des variables. C'est cette expression qui est utilisée lorsque la fonction sert à combiner plusieurs registres à décalage à rétroaction linéaire puisqu'elle induit la valeur de la complexité linéaire du générateur pseudo-aléatoire résultant de cet assemblage.

2 Combinaison de registres à décalage à rétroaction linéaire

Tout système de chiffrement à flot repose sur la possibilité de générer une suite aléatoire de n'importe quelle longueur, qui constitue la clef secrète du système et sera ensuite additionnée digit à digit au texte clair. Comme il n'est, en général, pas envisageable de partager une clef secrète qui soit aussi longue que l'ensemble des messages à chiffrer, on a recours à un générateur pseudo-aléatoire qui produit de façon déterministe une longue suite à partir d'une clef secrète courte. L'élément de base utilisé pour constituer un tel générateur est souvent le *registre à décalage à rétroaction linéaire*. Il s'agit en effet d'un composant simple, très rapide et facile à implémenter en hardware.

2.1 Registres à décalage à rétroaction linéaire et suites à récurrence linéaire

Un registre à décalage à rétroaction linéaire q-aire de longueur N est composé d'un registre à décalage contenant une suite de N digits $(s_i, \ldots, s_{i+N-1})$ de $\mathbb{F}_q$, et d'une fonction de rétroaction linéaire.

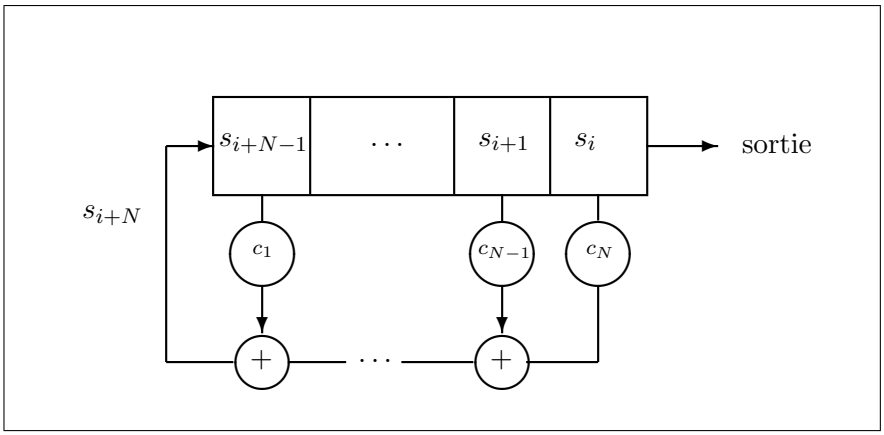


FIG. 7.1 –: *Fonctionnement d'un registre à décalage à rétroaction linéaire*

A chaque top d'horloge, le digit de poids faible s_i constitue la sortie du registre, et les autres digits sont décalés vers la droite (cf Figure 7.1). Le nouveau

digit s_{i+N} placé dans la cellule de poids fort du registre est donné par une fonction linéaire des digits $(s_i, \ldots, s_{i+N-1})$

$$s_{i+N} = c_1 s_{i+N-1} + c_2 s_{i+N-2} + \ldots + c_N s_i$$

où les coefficients de rétroaction $(c_i)_{1 \leq i \leq N}$ sont des éléments de $\mathbb{F}_q$.

Les digits $(s_0, \ldots, s_{N-1})$, qui déterminent entièrement la suite, constituent *l'état initial du registre*. J'utiliserai par la suite pour désigner un registre à décalage à rétroaction linéaire l'abréviation anglo-saxonne usuelle LFSR (pour Linear Feedback Shift Register).

La suite $(s_n)_{n \geq 0}$ produite par un LFSR de longueur N est donc une *suite à récurrence linéaire homogène d'ordre N*. Inversement, toute suite à récurrence linéaire homogène d'ordre N peut être générée au moyen d'un LFSR de longueur N.

On peut alors montrer qu'une telle suite est *ultimement périodique*, c'est-à-dire qu'il existe un indice n_0 (*la pré-période*) tel que la suite $(s_n)_{n \geq n_0}$ est périodique.

Proposition 7.4 [Man94] *Toute suite à récurrence linéaire homogène d'ordre N sur $\mathbb{F}_q$ est ultimement périodique et sa plus petite période T est inférieure ou égale à $q^N - 1$. De plus, si le coefficient de rétroaction c_N est non nul, alors la suite est périodique.*

preuve : S'il existe une itération n_0 pour laquelle les digits $(s_{n_0}, \ldots, s_{n_0+N-1})$ sont tous nuls, alors tous les digits suivants produits par le registre sont nuls. La suite est donc ultimement périodique de période 1.

Dans le cas contraire, le registre peut contenir n'importe lequel des $q^N - 1$ N-uplets de $\mathbb{F}_q^N \setminus (0, \ldots, 0)$. Il existe donc deux indices i et j, $0 \leq i < j \leq q^N - 1$ tels que $s_i = s_j$. En appliquant la relation de récurrence linéaire, on en déduit donc que $\forall n \geq i, \;\; s_{n+j-i} = s_n$. La suite $(s_n)_{n \geq 0}$ est donc ultimement périodique et sa plus petite période T vérifie $T \leq j - i \leq q^N - 1$.

Lorsque le coefficient de rétroaction c_N est non nul, la relation de récurrence donnant le terme d'indice $n_0 + T + N - 1$, où n_0 est la pré-période, peut s'écrire

$$\begin{aligned} s_{n_0+T-1} &= c_N^{-1}[s_{n_0+T+N-1} - c_1 s_{n_0+T+N-2} - \ldots - c_{N-1} s_{n_0+T}] \\ &= c_N^{-1}[s_{n_0+N-1} - c_1 s_{n_0+N-2} - \ldots - c_{N-1} s_{n_0}] \\ &= s_{n_0-1} \text{ si} n_0 \geq 1 \end{aligned}$$

En conséquence, la suite admet également $n_0 - 1$ comme pré-période. On en déduit donc que, dans ce cas, la suite $(s_n)_{n \geq 0}$ est périodique. □

En cryptographie, il est évidemment primordial de construire des suites de période maximale afin qu'elles soient aussi difficiles que possible à prévoir. Cette propriété est obtenue grâce à une condition sur le *polynôme caractéristique du registre*. Nous reprenons ici certains résultats classiques donnés par exemple

dans [LN83, Rue86], et nous adopterons la terminologie utilisée dans [LN83] et [Her86].

Définition 7.5 (Polynôme caractéristique d'un LFSR) *Soit un LFSR q-aire régi par la fonction de rétroaction*

$$\forall i \geq 0, \quad s_{i+N} = c_1 s_{i+N-1} + c_2 s_{i+N-2} + \ldots + c_N s_i \tag{7.1}$$

Son polynôme caractéristique *est le polynôme f de degré N à coefficients dans le corps $\mathbb{F}_q$*

$$f(X) = X^N - c_1 X^{N-1} - c_2 X^{N-2} - \ldots - c_N$$

Définition 7.6 (Ordre d'un polynôme) *Soit f un polynôme de $\mathbb{F}_q[X]$. Son ordre, noté $ord(f)$, est le plus petit entier t tel que $X^t \equiv 1 \bmod f(X)$.*

Proposition 7.7 *Soit un LFSR q-aire de polynôme caractéristique f. Alors la plus petite période de la suite $(s_n)_{n\geq 0}$ produite par ce LFSR divise l'ordre de f.*

preuve : Notons $S_n = (s_n, \ldots, s_{n+N-1})$. La relation de récurrence régissant la suite peut s'exprimer de manière matricielle :

$$\forall n \geq 0, \quad S_{n+1} = S_n A$$

$$\text{avec } A = \begin{pmatrix} 0 & 0 & 0 & \ldots & 0 & c_N \\ 1 & 0 & 0 & \ldots & 0 & c_{N-1} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & c_1 \end{pmatrix}$$

Si $c_N \neq 0$, la matrice A est un élément du groupe multiplicatif $GL_N(\mathbb{F}_q)$. Son ordre t dans ce groupe est alors une période de la suite $(s_n)_{n\geq 0}$: on a en effet, pour tout entier n, $S_{n+t} = S_0 A^{n+t} = S_0 A^n = S_n$. Or, la matrice A étant la matrice compagnon du polynôme caractéristique f, t est égal à l'ordre de f. On en déduit donc que $ord(f)$ est une période de la suite. Comme toute période de $(s_n)_{n\geq 0}$ est divisible par sa plus petite période, on a montré le résultat énoncé dans le cas où le coefficient de rétroaction c_N est non nul.

Dans le cas contraire, il suffit de factoriser le polynôme f sous la forme $f(X) = X^h g(X)$ avec $g(0) \neq 0$. La suite $(u_n)_{n\geq 0}$ définie par $\forall n \geq 0, \quad u_n = s_{n+h}$ est une suite à récurrence linéaire de polynôme caractéristique g et de plus petite période égale à celle de $(s_n)_{n\geq 0}$. En appliquant le résultat précédent à la suite $(u_n)_{n\geq 0}$, on obtient que la plus petite période de $(s_n)_{n\geq 0}$ divise l'ordre de g qui est lui-même égal à l'ordre de f. □

Le cas où le polynôme caractéristique f est irréductible est particulièrement intéressant puisque le terme général de la suite peut alors être exprimé de manière simple en fonction d'une racine de f dans le corps d'extension $\mathbb{F}_{q^N}$.

Proposition 7.8 *Soit $(s_n)_{n\geq 0}$ une suite de $\mathbb{F}_q$ produite par un LFSR q-aire dont le polynôme f est irréductible. Soit α une racine de f dans le corps d'extension $\mathbb{F}_{q^N}$. Alors, il existe un unique élément θ de $\mathbb{F}_{q^N}$ tel que*

$$\forall n \geq 0, \quad s_n = Tr(\theta\alpha^n)$$

où Tr désigne la trace de $\mathbb{F}_{q^N}$ sur $\mathbb{F}_q$.

preuve : Comme α est racine d'un polynôme irréductible de degré N sur $\mathbb{F}_q$, $(1, \alpha, \alpha^2, \ldots, \alpha^{N-1})$ est une base du $\mathbb{F}_q$-espace vectoriel $\mathbb{F}_{q^N}$. Les N formes linéaires

$$\begin{array}{ccc} \mathbb{F}_{q^N} & \rightarrow & \mathbb{F}_q \\ x & \mapsto & Tr(x\alpha^i) \end{array}$$

pour $0 \leq i \leq N-1$, sont alors linéairement indépendantes. Considérons en effet un N-uplet $\lambda = (\lambda_0, \ldots, \lambda_{N-1})$ tel que

$$\forall x \in \mathbb{F}_{q^N}, \quad \sum_{i=0}^{N-1} \lambda_i Tr(x\alpha^i) = 0$$

Par linéarité de la fonction trace, λ vérifie

$$\forall x \in \mathbb{F}_{q^N}, \quad Tr(x \sum_{i=0}^{N-1} \lambda_i \alpha^i) = 0$$

ce qui implique que $\sum_{i=0}^{N-1} \lambda_i \alpha^i = 0$ et donc que tous les coefficients λ_i sont nuls.

L'application $x \mapsto (Tr(x), Tr(x\alpha), \ldots, Tr(x\alpha^{N-1}))$ est donc un automorphisme du $\mathbb{F}_q$-espace vectoriel $\mathbb{F}_{q^N}$. Ainsi, à l'état initial $(s_0, \ldots, s_{N-1})$ de la suite $(s_n)_{n\geq 0}$, on peut associer un unique $\theta \in \mathbb{F}_{q^N}$ tel que

$$\forall\ 0 \leq n \leq N-1, \quad s_n = Tr(\theta\alpha^n)$$

En écrivant la relation de récurrence, on obtient alors que

$$\forall n \geq 0, \quad s_n = Tr(\theta\alpha^n)$$

□

Cette expression permet entre autres de prouver que, dans le cas où le polynôme caractéristique du LFSR est irréductible et tel que $f(0) \neq 0$, la plus petite période de la suite produite est exactement l'ordre de f, à condition que l'état initial soit non nul.

2.2 Complexité linéaire d'un registre à décalage

On peut également utiliser pour décrire une suite à récurrence linéaire la *méthode de la série génératrice* développée par exemple dans [LN83, Zie59]. Elle consiste à exprimer la suite $(s_n)_{n\geq 0}$ au moyen de la série formelle $s(X) = \sum_{n\geq 0} s_n X^n$. Cette description fait alors intervenir le *polynôme de rétroaction du registre.*

Définition 7.9 (Polynôme de rétroaction d'un LFSR) *Soit un LFSR dont la fonction de rétroaction linéaire est donnée par la relation de récurrence 7.1. Son* polynôme de rétroaction $f^\star$ *est le polynôme réciproque de son polynôme caractéristique, c'est-à-dire*

$$f^\star(X) = 1 - c_1 X - c_2 X^2 - \ldots - c_N X^N = X^N f(1/X)$$

On peut alors écrire le développement en série formelle de la suite $(s_n)_{n\geq 0}$ sous forme d'une fraction rationnelle.

Proposition 7.10 (Méthode de la série génératrice) *La suite $(s_n)_{n\geq 0}$ est produite par un LFSR dont le polynôme de rétroaction est $f^\star(X) = 1 - c_1 X - c_2 X^2 - \ldots - c_N X^N$ si et seulement si son développement en série formelle $s(X) = \sum_{n=0}^{\infty} s_n X^n$ s'écrit*

$$s(X) = \frac{g(X)}{f^\star(X)}$$

où g est un polynôme sur $\mathbb{F}_q$ tel que $deg(g) < deg(f^\star)$.

En outre le polynôme g est entièrement déterminé par l'état initial de la suite :

$$g(X) = -\sum_{i=0}^{N-1} X^i \sum_{j=0}^{i} c_{i-j} s_j$$

preuve : Posons $c_0 = -1$. On a alors

$$\begin{aligned} f^\star(X)s(X) &= (-\sum_{n=0}^{N} c_n X^n)(\sum_{n=0}^{\infty} s_n X^n) \\ &= -\sum_{n=0}^{\infty} X^n (\sum_{i=0}^{n} s_i c_{n-i}) \\ &= g(X) - \sum_{n=N}^{\infty} X^n (\sum_{i=n-N}^{n} s_i c_{n-i}) \end{aligned}$$

Donc $f^\star(X)s(X) = g(X)$ si et seulement si $\forall n \geq N$, $\sum_{i=n-N}^{n} s_i c_{n-i} = 0$, ce qui équivaut à dire que la suite $(s_n)_{n\geq 0}$ vérifie la relation de récurrence 7.1.

Comme $c_0 \neq 0$, $f^\star$ a un inverse multiplicatif dans l'anneau des séries formelles $\mathbb{F}_q[[X]]$; on en déduit donc que le développement en série formelle de la fraction rationnelle $\frac{g(X)}{f^\star(X)}$ est égal à $s(X)$. □

Afin d'obtenir une forme canonique de la série génératrice de $(s_n)_{n\geq 0}$, on définit le *polynôme caractéristique minimal de la suite* (ou du registre). En effet

si le polynôme caractéristique f se décompose sous la forme $f(X) = d(X)f_1(X)$ et que le polynôme réciproque de d, $d^\star$ divise le polynôme $g = d^\star g_1$, alors la série génératrice de la suite $(s_n)_{n\geq 0}$ peut également s'écrire

$$s(X) = \frac{g_1(X)}{f_1^\star} \text{ avec } deg(g_1) < deg(f_1^\star)$$

car le polynôme réciproque du produit df_1 est égal au produit des polynômes réciproques $d^\star$ et $f_1^\star$. Le polynôme caractéristique minimal de la suite $(s_n)_{n\geq 0}$ est donc un diviseur du polynôme caractéristique du registre ; il est le polynôme de plus bas degré parmi les polynômes caractéristiques de tous les registres capables d'engendrer $(s_n)_{n\geq 0}$.

Définition 7.11 (Complexité linéaire) *Soit $(s_n)_{n\geq 0}$ une suite à rétroaction linéaire d'ordre N sur $\mathbb{F}_q$ dont l'état initial est non nul. Son* polynôme caractéristique minimal *est l'unique polynôme unitaire $f_0 \in \mathbb{F}_q[X]$ tel qu'il existe un polynôme $g_0 \in \mathbb{F}_q[X]$, avec $deg(g_0) < N$ et $pgcd(g_0, f_0^\star) = 1$, vérifiant*

$$s(X) = \frac{g_0(X)}{f_0^\star(X)}$$

où $f_0^\star$ est le polynôme réciproque de f_0. La complexité linéaire *du LFSR produisant la suite $(s_n)_{n\geq 0}$, notée $\Lambda(s)$, est alors égale au degré du polynôme caractéristique minimal de $(s_n)_{n\geq 0}$.*

La complexité linéaire d'un LFSR produisant une suite $(s_N)_{n\geq 0}$ est donc la longueur du plus petit LFSR permettant d'engendrer cette suite. Il s'agit d'un paramètre essentiel pour beaucoup d'applications cryptographiques. En effet, J. Massey [Mas69] a montré que l'algorithme proposé par Berlekamp [Ber68] pour le décodage des codes BCH permet également de trouver le polynôme caractéristique minimal d'une suite à partir uniquement de ses $2\Lambda(s)$ premiers digits. La complexité linéaire d'un LFSR est donc un paramètre déterminant pour sa sécurité cryptographique : l'observation d'un petit nombre de digits seulement de la suite permet de la reconstituer entièrement ce qui s'avère évidemment catastrophique. Aussi s'attache-t-on généralement à utiliser des LFSRs dont la complexité linéaire est égale à la longueur. Ceci est en particulier possible si le polynôme caractéristique du registre est irréductible ; on est alors assuré qu'il est impossible d'engendrer la suite $(s_n)_{n\geq 0}$ à l'aide d'un LFSR plus court.

Exemple 7.12: Considérons la suite binaire $(s_n)_{n\geq 0}$ produite par le LFSR de longueur 10 décrit par la figure 7.2.

Le polynôme caractéristique $f \in \mathbb{F}_2[X]$ de ce registre est

$$f(X) = X^{10} + X^9 + X^7 + X^6 + X^3 + 1$$

Son polynôme de rétroaction est donc

$$f^\star(X) = X^{10} + X^7 + X^4 + X^3 + X + 1$$

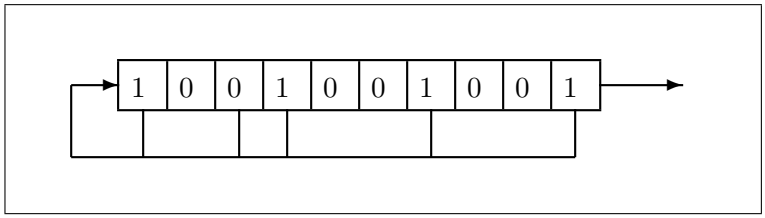


FIG. 7.2 –: *Exemple de LFSR de longueur 10*

Soit g le polynôme déterminé par l'état initial du registre $(s_0, \ldots, s_9)$: $g(X) = \sum_{n=0}^{9} X^n \sum_{i=0}^{n} c_{n-i} s_i$, où c_i est le coefficient du terme de degré i de $f^\star$. On obtient alors

$$g(X) = X^7 + X + 1$$

En appliquant la méthode de la série génératrice définie à la proposition 7.10, on peut alors calculer la série génératrice de la suite $(s_n)_{n\geq 0}$:

$$\sum_{n=0}^{\infty} s_n X_n = \frac{g(X)}{f^\star(X)} = \frac{X^7 + X + 1}{X^{10} + X^7 + X^4 + X^3 + X + 1} = \frac{1}{X^3 + 1}$$

Le polynôme caractéristique minimal du registre est donc

$$f_0(X) = X^3 + 1$$

ce qui signifie que la suite $(s_n)_{n\geq 0}$ peut être générée au moyen d'un LFSR de longueur 3.

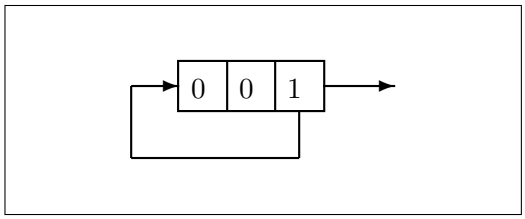


FIG. 7.3 –: *LFSR de longueur 3 produisant la même suite que le LFSR de la figure précédente*

En effet, le registre de la figure 7.3 produit la même suite que celui de la figure 7.2.

Imaginons que le LFSR de la figure 7.2 soit utilisé dans une application cryptographique. On voit alors qu'un attaquant peut entièrement déterminer la suite $(s_n)_{n\geq 0}$ s'il connaît uniquement les bits s_0, s_1, s_2, s_3, s_4 et s_5.

Le polynôme caractéristique minimal du registre joue également un grand rôle dans la détermination de la plus petite période de la suite engendrée.

Proposition 7.13 (Période d'une suite à récurrence linéaire) *Soit $(s_n)_{n\geq 0}$ une suite à récurrence linéaire de polynôme caractéristique minimal f_0 et dont l'état initial est non nul. Sa plus petite période est égale à l'ordre de f_0.*

preuve : Soit T la plus petite période de la suite $(s_n)_{n\geq 0}$ et n_0 la pré-période correspondante. Alors, $(s_n)_{n\geq 0}$ vérifie la relation de récurrence

$$\forall n \geq 0, \quad s_{n+n_0+T} = s_{n+n_0}$$

Le polynôme $f(X) = X^{n_0+T} - X^{n_0} = X^{n_0}(X^T - 1)$ est donc également un polynôme caractéristique de la suite. Le polynôme caractéristique minimal de la suite, f_0, divise alors f et s'écrit $f_0(X) = X^h g(X)$ où $h \leq n_0$ et $g(X)$ divise $X^T - 1$. Comme l'ordre de f_0 est égal à l'ordre de g, on en déduit que $ord(f_0) \leq T$. Or, d'après la proposition 7.7, T divise $ord(f_0)$. La plus petite période T de la suite est par conséquent égale à l'ordre de f_0. □

Pour une complexité linéaire donnée, on recherche surtout les suites dont la période est la plus grande possible. Ces suites, dites *suites ML* (de longueur maximale) sont donc obtenues à partir de registres dont les polynômes caractéristiques minimaux sont primitifs et l'état initial non nul.

Les racines du polynôme caractéristique minimal permettent également d'expliciter la forme du terme général de la suite engendrée, comme je l'ai fait à la proposition 7.8 dans le cas d'un polynôme caractéristique irréductible.

Proposition 7.14 (Terme général d'une suite à partir des racines de son polynôme caractéristique minimal) *Soit $(s_n)_{n\geq 0}$ une suite à récurrence linéaire de $\mathbb{F}_q$, $(\alpha_1, \ldots, \alpha_N)$ les racines de son polynôme caractéristique minimal f_0 dans son corps de décomposition F et $(m_1, \ldots, m_N)$ leurs multiplicités respectives. Alors il existe des éléments de F, $(\theta_{i,j})_{\substack{1 \leq i \leq N \\ 0 \leq j \leq m_i - 1}}$ tels que*

$$\forall n \geq 0, \quad s_n = \sum_{i=1}^{N} \sum_{j=0}^{m_i - 1} \theta_{i,j} \binom{n+j}{j} \alpha_i^n$$

preuve : D'après la proposition 7.10, la série génératrice de la suite $(s_n)_{n\geq 0}$ s'écrit

$$s(X) = \frac{g(X)}{f_0^\star(X)} \text{ avec } deg(g) < deg(f_0)$$

Or, il existe des coefficients $(\gamma_{i,j})_{\substack{1 \leq i \leq N \\ 0 \leq j \leq m_i - 1}}$ dans le corps de décomposition F de f_0 tels que

$$\frac{1}{f_0^\star(X)} = \frac{1}{\prod_{i=1}^{N}(1 - \alpha_i X)^{m_i}} = \sum_{i=1}^{N} \sum_{j=0}^{m_i - 1} \frac{\gamma_{i,j}}{(1 - \alpha_i X)^{j+1}}$$

En utilisant la formule de Taylor, on peut écrire

$$(1-\alpha_i X)^{-(j+1)} = \sum_{n\geq 0} \binom{n+j}{j} \alpha_i^n X^n$$

ce qui induit pour la série génératrice l'existence de coefficients $\theta_{i,j}$ tels que

$$s(X) = \sum_{n\geq 0} \sum_{i=1}^{N} \sum_{j=0}^{m_i-1} \theta_{i,j} \binom{n+j}{j} \alpha_i^n X^n$$

□

Un cas particulier de ce théorème est celui des LFSRs dont le polynôme caractéristique est un *polynôme homogène d'ordre t*, c'est-à-dire un polynôme qui se décompose en un produit de polynômes irréductibles distincts de mêmes degrés, ayant chacun un ordre t. Il est en effet possible de déterminer la période et la complexité linéaire d'une suite générée par un tel LFSR.

Corollaire 7.15 (Suite générée par un LFSR de polynôme caractéristique homogène) *Soit $(s_n)_{n\geq 0}$ une suite de $\mathbb{F}_q$ générée par un LFSR dont le polynôme caractéristique f est un polynôme homogène d'ordre t et de degré N, avec $f(X) = f_1(X)\dots f_k(X)$. Sa complexité linéaire est alors $\Lambda(s) = N$ et sa plus petite période est égale à t.*

De plus, si pour tout $1 \leq i \leq k$ on note α_i une racine de f_i dans le corps d'extension, il existe un unique k-uplet $(\theta_1, \dots, \theta_k)$ tel que

$$\forall n \geq 0, \quad s_n = Tr(\sum_{i=1}^{k} \theta_i \alpha_i^n)$$

preuve : Par la méthode de la série génératrice (proposition 7.10), il existe un polynôme $g \in \mathbb{F}_q[X]$ de degré au plus $N-1$ tel que

$$\sum_{n=0}^{\infty} s_n X^n = \frac{g(X)}{f^\star(X)}$$

Décomposer la fraction rationnelle sous la forme

$$\frac{g(X)}{f^\star(X)} = \sum_{i=1}^{k} \frac{g_i(X)}{f_i^\star(X)} \text{ avec } deg(g_i) < deg(f_i)$$

revient à exprimer la suite $(s_n)_{n\geq 0}$ comme une somme de suites de polynômes caractéristiques respectifs f_i. Les f_i étant irréductibles, on en déduit d'une part, à l'aide de la proposition 7.8, l'expression du terme général de $(s_n)_{n\geq 0}$ utilisant la fonction Trace, et d'autre part que g est premier avec f puisque chaque g_i est premier avec f_i. Il s'ensuit donc que f est le polynôme caractéristique minimal de la suite et, en conséquence, que $\Lambda(s) = N$. Comme $ord(f) = ppcm(ord(f_i)) = t$, la plus petite période de la suite $(s_n)_{n\geq 0}$ est égale à t. □

2.3 Combinaison de suites à récurrence linéaire

Toutefois, même lorsque le polynôme caractéristique du registre est choisi de façon appropriée (par exemple irréductible), la complexité linéaire de la suite produite reste généralement trop faible pour se mettre à l'abri d'une attaque par l'algorithme de Berlekamp-Massey. Afin de surmonter cet obstacle et d'augmenter la taille de l'espace des clefs, on utilise donc souvent k LFSRs q-aires en parallèle, dont les sorties sont combinées par une fonction $f : \mathbb{F}_q^k \to \mathbb{F}_q$, dite *fonction de combinaison* (ou d'assemblage).

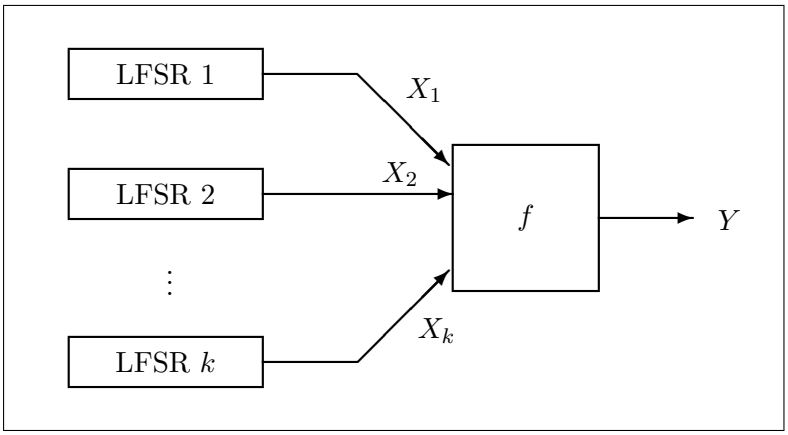


FIG. 7.4 –: *Combinaison de plusieurs LFSRs*

La suite ainsi obtenue étant toujours une suite à récurrence linéaire, il est indispensable d'estimer sa complexité linéaire et donc la taille effective de l'espace des clefs du générateur. Nous avons vu précédemment que toute fonction d'assemblage f sur $\mathbb{F}_q$ pouvait s'exprimer sous forme normale algébrique, c'est-à-dire au moyen d'un polynôme à coefficients dans $\mathbb{F}_q$. On peut en conséquence décomposer l'étude de la complexité linéaire résultant de la combinaison de plusieurs suites par la fonction f en trois étapes :

– la complexité linéaire de la somme de deux suites

– la complexité linéaire du produit de deux suites

– la complexité linéaire d'une suite élevée à une puissance $s > 0$.

Il est en effet évident que la multiplication d'une suite par un coefficient constant non nul ne modifie pas sa complexité linéaire. Les résultats présentés ici sont dus à différents auteurs mais ils sont pour la plupart résumés dans [Her86].

2.3.1 Somme de deux suites à récurrence linéaire

Proposition 7.16 *Soient $(u_n)_{n\geq 0}$ et $(v_n)_{n\geq 0}$ deux suites à récurrence linéaire de $\mathbb{F}_q$ de polynômes caractéristiques minimaux respectifs f_0 et g_0. Alors la complexité linéaire de leur somme est donnée par*

$$\Lambda(u+v) \leq \Lambda(u) + \Lambda(v)$$

avec égalité si et seulement si $pgcd(f_0, g_0) = 1$.
De plus, dans le cas d'égalité, la période de leur somme est égale au ppcm des périodes de $(u_n)_{n\geq 0}$ et de $(v_n)_{n\geq 0}$.

preuve : En additionnant les séries génératrices de ces suites dans $\mathbb{F}_q[[X]]$, on obtient

$$\sum_{n=0}^{\infty}(u_n + v_n)X^n = \frac{h_1(X)}{f_0(X)} + \frac{h_2(X)}{g_0(X)}$$

avec $pgcd(h_1, f_0) = 1$ et $pgcd(h_2, g_0) = 1$. En réduisant ces deux fractions au même dénominateur $p(X) = ppcm(f_0(X), g_0(X))$, on a

$$\sum_{n=0}^{\infty}(u_n + v_n)X^n = \frac{h(X)}{p(X)}$$

Le polynôme p est donc un polynôme caractéristique de la suite $(u_n + v_n)_{n\geq 0}$ et en conséquence un multiple de son polynôme caractéristique minimal. D'où $\Lambda(u+v) \leq \Lambda(u) + \Lambda(v)$.

Si f_0 et g_0 sont premiers entre eux, alors $p = f_0 g_0$ et $h(X) = h_1(X)g_0(X) + h_2(X)f_0(X)$, ce qui implique que h et p sont premiers entre eux, et donc que p est bien le polynôme caractéristique minimal de la suite $(u_n + v_N)_{n\geq 0}$. Dans ce cas, la période de la suite $(u_n + v_n)_{n\geq 0}$ est égale à l'ordre du polynôme $f_0 g_0$, c'est-à-dire à $ppcm(ord(f_0), ord(g_0))$. □

2.3.2 Produit de deux suites à récurrence linéaire

Le cas du produit de deux suites à récurrence linéaire a été exploré successivement par Selmer [Sel66], Zierler et Mills [ZM73], Herlestam [Her82, Her83, Her86], Rueppel et Staffelbach [RS87]et Golić [Gol89]. La plupart de ces auteurs se sont toutefois attachés à des cas particuliers, notamment ceux de LFSRs dont les polynômes minimaux sont soit irréductibles, soit sans racine multiple, puisqu'il est plus facile avec ces hypothèses de dériver des conditions suffisantes pour que le produit de deux suites atteigne une complexité linéaire maximale.

Il est en effet aisé de montrer dans le cas général que la complexité linéaire de la suite produit $(u_n v_n)_{n\geq 0}$ n'excède pas le produit des complexités linéaires de $(u_n)_{n\geq 0}$ et $(v_n)_{n\geq 0}$. Soit F un corps de décomposition commun des polynômes caractéristiques minimaux f_0 et g_0 des suites $(u_n)_{n\geq 0}$ et $(v_n)_{n\geq 0}$. Si $(\alpha_1, \ldots, \alpha_L)$ sont les racines avec les multiplicités $(m_1, \ldots, m_L)$ de f_0 dans le corps F, et

$(\beta_1, \ldots, \beta_M)$ les racines de g_0 de multiplicités $(n_1, \ldots, n_M)$, on peut en utilisant la proposition 7.14, exhiber des éléments $(A_{i,j})$ et $(B_{i,j})$ de F tels que

$$u_n = \sum_{i=1}^{L} \sum_{j=0}^{m_i-1} A_{i,j} \binom{n+j}{j} \alpha_i^n \text{ et } v_n = \sum_{i=1}^{M} \sum_{j=0}^{n_i-1} B_{i,j} \binom{n+j}{j} \beta_i^n$$

Comme le produit des coefficients binômiaux s'écrit aussi ([Rio79])

$$\text{pour } r \geq s, \quad \binom{n+r}{r}\binom{n+s}{s} = \sum_{t=r}^{r+s} (-1)^{r+s-t} \binom{t}{r}\binom{r}{t-s}\binom{n+t}{t}$$

on en déduit le terme général de la suite produit

$$u_n v_n = \sum_{i=1}^{L} \sum_{j=1}^{M} \sum_{t=0}^{m_i+n_j-2} \theta_{i,j,t} \binom{n+t}{t} (\alpha_i \beta_j)^n \tag{7.2}$$

Cette expression implique donc que la série génératrice de la suite $(u_n v_n)_{n\geq 0}$ est obtenue en développant une fraction rationnelle dont le dénominateur est le polynôme réciproque d'un polynôme h_0 ayant pour racines les $\alpha_i \beta_j$, chacun d'eux avec une multiplicité au plus $m_i n_j - 1$. Le degré du polynôme caractéristique minimal de la suite produit est donc inférieur ou égal à $\sum_{i=1}^{L} \sum_{j=1}^{M} m_i + n_j - 1 = \Lambda(u)M + L\Lambda(v) - LM$, ce qui induit

$$\Lambda(uv) \leq \Lambda(u)\Lambda(v)$$

La forme de $u_n v_n$ donnée par 7.2 donne plus précisément une condition nécessaire et suffisante pour que cette borne soit atteinte :

Proposition 7.17 [Her83, Her86] *Soient $(u_n)_{n\geq 0}$ et $(v_n)_{n\geq 0}$ deux suites à récurrence linéaire de $\mathbb{F}_q$, et soient $(\alpha_1, \ldots, \alpha_L)$ et $(\beta_1, \ldots, \beta_M)$ les racines de leurs polynômes caractéristiques minimaux f_0 et g_0 dans un corps de décomposition commun. La complexité linéaire de la suite produit $(u_n v_n)_{n\geq 0}$ vérifie $\Lambda(uv) = \Lambda(u)\Lambda(v)$ si et seulement si*

1. *au moins un des deux polynômes f_0 et g_0 n'a que des racines simples.*
2. *tous les produits $\alpha_i \beta_j$ pour $1 \leq i \leq L$ et $1 \leq j \leq M$ sont distincts.*

Le problème reste cependant de trouver des cas d'égalité plus explicites qui, en particulier, ne nécessitent pas le calcul des racines de f_0 et g_0. De nombreuses conditions plus ou moins restrictives sur les polynômes caractéristiques minimaux f_0 et g_0, permettant d'atteindre la complexité linéaire maximale, ont donc été développées. Il est par exemple connu, au moins depuis les travaux de Selmer [Sel66], que le produit de deux suites dont les polynômes caractéristiques minimaux sont irréductibles, sans racines multiples et de degrés premiers entre eux a une complexité linéaire maximale. En effet, les racines de f_0 sont de

la forme $(\alpha, \alpha^q, \ldots, \alpha^{q^{L-1}})$ dans $\mathbb{F}_{q^L}$, et celles de g_0 sont les $(\beta, \beta^q, \ldots, \beta^{q^{M-1}})$ dans $\mathbb{F}_{q^M}$. Le produit $\alpha\beta$ est donc racine d'un polynôme caractéristique de la suite produit. Or, tous les conjugués de $\alpha\beta$ dans $\mathbb{F}_{q^{LM}}$ sont distincts lorsque L et M sont premiers entre eux, ce qui implique que le degré du polynôme caractéristique minimal de la suite produit est égal à LM.

Je donne ici une autre condition, due à Golić [Gol89], pour que le produit de deux suites atteigne la complexité linéaire maximale. Elle s'inspire de l'idée introduite par Rueppel et Staffelbach [RS87] qui consiste à exprimer une condition sur les ordres des polynômes caractéristiques minimaux et non sur leurs degrés. Une telle condition est en effet moins contraignante car elle n'impose pas de combiner des LFSRs dont les longueurs sont premières entre elles. Je m'intéresse donc au produit de deux suites dont les polynômes caractéristiques minimaux sont d'ordres premiers entre eux. J'utiliserai à cette fin le lemme suivant :

Lemme 7.18 *Soient G_1 et G_2 deux sous-groupes d'un groupe cyclique G, d'ordres respectifs r_1 et r_2. Si r_1 et r_2 sont premiers entre eux, alors l'application*

$$\begin{array}{rccc} \pi : & G_1 \times G_2 & \rightarrow & G \\ & (x, y) & \mapsto & xy \end{array}$$

est injective.

preuve : L'ordre du groupe cyclique G, $|G|$, est par hypothèse un multiple de $r_1 r_2$, $|G| = d r_1 r_2$. Soit α un générateur de G. Alors α^{dr_2} (respectivement α^{dr_1}) est un générateur du sous-groupe G_1 (resp. G_2). Lorsque r_1 et r_2 sont premiers entre eux, le théorème de Bezout induit l'existence d'un couple d'entiers (u_1, u_2) tels que $\alpha^{du_1 r_1} \alpha^{du_2 r_2} = \alpha^d$. Ainsi $\pi(G_1 \times G_2)$ contient un groupe d'ordre $r_1 r_2 = |G_1 \times G_2|$. La fonction π est donc injective. □

Théorème 7.19 (Complexité linéaire du produit de deux suites) [Gol89] *Soient $(u_n)_{n\geq 0}$ et $(v_n)_{n\geq 0}$ deux suites à récurrence linéaires de $\mathbb{F}_q$ dont les polynômes caractéristiques minimaux sont d'ordres premiers entre eux. Alors la complexité linéaire de la suite produit $(u_n v_n)_{n\geq 0}$ est donnée par*

$$\Lambda(uv) = \Lambda(u)\Lambda(v)$$

et sa période est égale au produit des périodes de $(u_n)_{n\geq 0}$ et $(v_n)_{n\geq 0}$.

preuve : Décomposons les polynômes caractéristiques minimaux f_0 et g_0 de $(u_n)_{n\geq 0}$ et $(v_n)_{n\geq 0}$ en produit de facteurs irréductibles

$$f_0 = \prod_{i=1}^{L} f_i^{m_i} \text{ et } g_0 = \prod_{i=1}^{M} g_i^{n_i}$$

et notons $(e_1, \ldots, e_L)$ (resp. $(e'_1, \ldots, e'_M)$) les ordres des facteurs $(f_1, \ldots, f_L)$ (resp. $(g_1, \ldots, g_M)$). Alors l'ordre de f_0 s'exprime en fonction des ordres de ses facteurs irréductibles

$$ord(f_0) = p^s ppcm(e_1, \ldots, e_L) = p^s ord(\prod_{i=1}^{L} f_i)$$

où p est la caractéristique du corps $\mathbb{F}_q$ et $s = \min\{i \geq 0, p^i \geq \max(m_1, \ldots, m_L)\}$. De même

$$ord(g_0) = p^{s'} ppcm(e'_1, \ldots, e'_M) = p^{s'} ord(\prod_{i=1}^{M} g_i)$$

avec $s' = \min\{i \geq 0,\ p^i \geq \max(n_1, \ldots, n_M)\}$. Donc, les ordres de f_0 et g_0 sont premiers entre eux si et seulement si les ordres des polynômes $\hat{f} = \prod_{i=1}^{L} f_i$ et $\hat{g} = \prod_{i=1}^{M} g_i$ sont premiers entre eux et que au moins l'un des entiers s ou s' est nul. Cette deuxième condition équivaut au fait qu'au moins un des polynômes f_0 ou g_0 n'a que des racines simples.

Soit F un corps de décomposition commun de f_0 et g_0. Comme les ordres de $\hat{f}$ et $\hat{g}$ sont premiers entre eux, les sous-groupes multiplicatifs du groupe cyclique $F^\star$ engendrés respectivement par les racines de f_0 et g_0 sont premiers entre eux, ce qui implique, d'après le lemme précédent, que tous les produits $\alpha_i\beta_j$ de racines de f_0 et g_0 sont distincts. Ainsi la condition de la proposition 7.17 est vérifiée, et la complexité linéaire de la suite produit est par conséquent maximale.

L'ordre du polynôme dont les racines sont les $\alpha_i\beta_j$ est donc égal au produit des ordres de $\hat{f}$ et $\hat{g}$, ce qui implique d'autre part que la période de la suite produit est égale au produit des périodes des suites $(u_n)_{n\geq 0}$ et $(v_n)_{n\geq 0}$. □

On remarque que les hypothèses du théorème n'excluent pas le cas où les degrés des polynômes caractéristiques sont identiques.

Exemple 7.20: (Figure 7.5) Soient $(u_n)_{n\geq 0}$ et $(v_n)_{n\geq 0}$ deux suites binaires générées par des LFSRs de longueur 10 dont les polynômes caractéristiques minimaux respectifs sont

$$f_0(X) = X^{16} + X^{15} + X^{13} + X^8 + X^5 + X^3 + 1$$

$$g_0(X) = X^{16} + X^9 + X^8 + X^7 + X^5 + X^3 + 1 = (X^7 + X^3 + 1)(X^9 + X^5 + 1)$$

Le polynôme f_0 étant primitif, son ordre est égal à $2^{16} - 1 = 65534$. L'ordre de g_0 est, lui, égal à $ppcm(2^7 - 1, 2^9 - 1) = 64897$. Comme les ordres de f_0 et de g_0 sont premiers entre eux, la complexité linéaire de la suite produit $(u_n v_n)_{n\geq 0}$ est égale à 100 et sa période vaut $65534 * 64897 \simeq 2^{32}$.

Le lemme 7.18 se généralise au cas de k sous-groupes d'ordres premiers entre eux deux à deux. On en déduit donc que le produit de k suites dont les

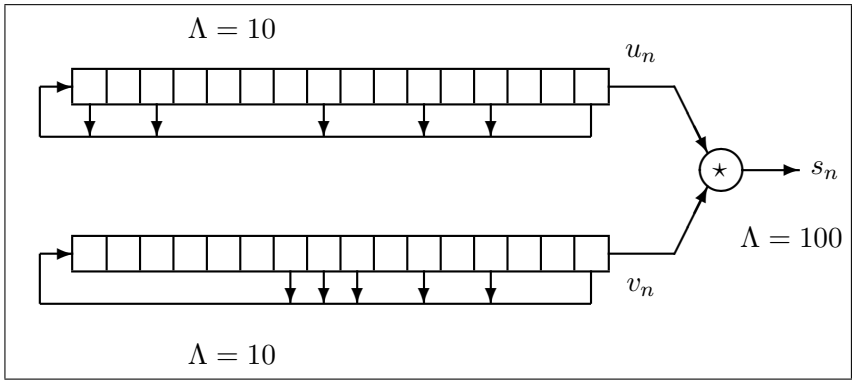


FIG. 7.5 –: *Exemple de produit de deux LFSRs de complexité linéaire optimale*

polynômes caractéristiques sont d'ordres *deux à deux* premiers entre eux atteint la complexité linéaire optimale.

Corollaire 7.21 (Complexité linéaire du produit de k suites) *Soient k suites à récurrence linéaire $(u_n^{(1)})_{n\geq 0}, \ldots, (u_n^{(k)})_{n\geq 0}$ de $\mathbb{F}_q$ dont les polynômes caractéristiques minimaux sont d'ordres deux à deux premiers entre eux. Alors, la complexité linéaire de la suite produit est*

$$\Lambda(u^{(1)} \ldots u^{(k)}) = \prod_{i=1}^{k} \Lambda(u^{(i)})$$

et sa période est égale au produit des périodes des suites $(u_n^{(i)})_{n\geq 0}$.

2.3.3 Puissance d'une suite à récurrence linéaire

J'étudie ici la suite à récurrence linéaire $(u_n)_{n\geq 0}$ de $\mathbb{F}_q$, pour un exposant positif $s < q$.

La complexité linéaire de la suite $(u_n^s)_{n\geq 0}$ a été déterminée par Herlestam [Her86] et Brynielsson [Bry86]. La démonstration utilise un corollaire du théorème de Lucas [Luc78] donnant la valeur des coefficients multinômiaux sur un corps de caractéristique p.

Lemme 7.22 (théorème de Lucas pour les coefficients multinômiaux) *Soit p un entier premier. A tout entier a dont la décomposition en base p s'écrit $a = \sum_{i=0}^{e} a_i p^i$ avec $0 \leq a_i < p$, on associe le polynôme $a(X) = \sum_{i=0}^{e} a_i X^i$. On a alors l'équivalence suivante*

$$\frac{s!}{i_1! \ldots i_N!} \not\equiv 0 \bmod (p) \Longleftrightarrow i_1(X) + \ldots + i_N(X) = s(X)$$

où $s = i_1 + i_2 + \ldots + i_N$.

preuve : On peut exprimer le coefficient multimômial sous forme d'un produit de coefficients binômiaux

$$\frac{s!}{i_1!\dots i_N!} = \binom{s}{i_1}\binom{s-i_1}{i_2}\dots\binom{s-i_1-\dots-i_{N-2}}{i_{N-1}}$$

Or, le théorème de Lucas donne l'expression des coefficients binômiaux modulo p : pour deux entiers a et b dont les décompositions en base p sont respectivement $a = \sum_{i=0}^{e} a_i p^i$ et $b = \sum_{i=0}^{e} b_i p^i$, on a

$$\binom{a}{b} \equiv \prod_{i=0}^{e} \binom{a_i}{b_i} \bmod p$$

On en déduit donc que

$$\frac{s!}{i_1!\dots i_N!} = \prod_{k=0}^{e} \binom{s_k}{(i_1)_k}\binom{(s-i_1)_k}{(i_2)_k}\dots\binom{(s-i_1-\dots-i_{N-2})_k}{(i_{N-1})_k}$$

Or $\binom{s_k}{(i_1)_k} \not\equiv 0 \bmod p$ si et seulement si $(i_1)_k \leq s_k$. Cette dernière condition induit que $(s-i_1)_k = s_k - (i_1)_k$. On montre alors par récurrence que le coefficient multinômial est non nul modulo p ssi

$$\forall\ 1 \leq k \leq e,\ \forall\ 0 \leq j \leq N,\ \ (i_j)_k \leq s_k - (i_0)_k - \dots - (i_{j-1})_k$$

ce qui équivaut également à dire que $\forall\ 1 \leq k \leq e,\ \ (i_1)_k + (i_2)_k + \dots + (i_N)_k = s_k$.
□

Théorème 7.23 (Complexité linéaire de la puissance d'une suite à récurrence linéaire) *Soit $\mathbb{F}_q$ le corps fini à q éléments de caractéristique p et $(u_n)_{n\geq 0}$ une suite à récurrence linéaire de $\mathbb{F}_q$ dont le polynôme caractéristique minimal est de degré N. Soit s un entier, $0 \leq s < q$ et $s = \sum_{i=0}^{e} s_i p^i$, $0 \leq s_i < p$, sa décomposition en base p. Alors la complexité linéaire de la suite puissance $(u_n^s)_{n\geq 0}$ vérifie*

$$\Lambda(u^s) \leq \prod_{i=0}^{e} \binom{N-1+s_i}{s_i}$$

avec égalité si $(u_n)_{n\geq 0}$ est une suite ML, c'est-à-dire si $ord(f) = q^N - 1$. Dans ce cas, la période de la suite puissance $(u_n^s)_{n\geq 0}$ est $q^N - 1$.

preuve : Soient $(\alpha_1, \dots, \alpha_N)$ les racines du polynôme f et $(m_1, \dots, m_N)$ leurs multiplicités. D'après la proposition 7.14, il existe des éléments $\theta_{i,j}$ dans le corps de décomposition de f tels que

$$\forall n \geq 0,\ \ u_n = \sum_{i=1}^{N} \left(\sum_{j=0}^{m_i-1} \theta_{i,j} \binom{n+j}{j} \right) \alpha_i^n = \sum_{i=1}^{N} A_{i,n} \alpha_i^n$$

Alors, le terme général de la suite puissance $(v_n)_{n\geq 0}$ est obtenu grâce à la formule du multinôme

$$v_n = u_n^s = \sum_{i_1+\ldots+i_N=s} \frac{s!}{i_1!\ldots i_N!} \left(\prod_{k=1}^{N} A_{k,n}^{i_k}\right)\left(\prod_{k=1}^{N} \alpha_k^{i_k}\right)^n \tag{7.3}$$

Les éléments $\prod_{k=1}^{N} \alpha_k^{i_k}$ distincts, pour lesquels les coefficients $\prod_{k=1}^{N} A_{k,n}^{i_k}$ et $\frac{s!}{i_1!\ldots i_N!}$ sont non nuls sur $\mathbb{F}_q$, sont donc racines d'un polynôme caractéristique de la suite $(v_n)_{n\geq 0}$. Or, avec les notations du lemme précédent, le coefficient multinômial est nul si et seulement si $i_1(X) + i_2(X) + \ldots + i_N(X) = s(X)$. Le nombre de N-uplets d'éléments positifs ou nuls dont la somme vaut s est égal au nombre de monômes en au plus N variables de degré total s, c'est-à-dire $\binom{N-1+s}{s}$. On en déduit donc que le degré du polynôme caractéristique p, et par conséquent la complexité linéaire $\Lambda(u^s)$ est inférieur à $\prod_{k=0}^{e} \binom{N-1+s_k}{s_k}$.

Dans le cas où la suite $(u_n)_{n\geq 0}$ est une suite ML, les racines de f sont de la forme $(\alpha, \alpha^q, \ldots, \alpha^{q^{N-1}})$ et le terme général de la suite s'écrit

$$u_n = \sum_{i=0}^{N-1} \theta^{q^i} \alpha^{q^i n}$$

Dans l'équation 7.3, tous les produits $\prod_{k=1}^{N} \alpha_k^{i_k} = \alpha^{\sum_{k=0}^{N-1} i_k q^k}$ sont donc distincts car $s < q$ et leurs coefficients $\prod_{k=0}^{N-1} \theta_k^{i_k}$ sont non nuls. De plus, le polynôme dont ces éléments sont les racines est un polynôme homogène d'ordre $q^N - 1$. On en déduit donc que la suite $(u_n^s)_{n\geq 0}$ atteint la complexité linéaire optimale.

□

On a en particulier dans le cas où $(u_n)_{n\geq 0}$ est une suite ML sur un corps de caractéristique 2,

$$\forall\, 0 \leq s < q-1, \;\; \Lambda(u^s) = \Lambda(u)^{w_H(s)}$$

où $w_H(s)$ désigne le poids de Hamming binaire de l'exposant s.

Exemple 7.24: (Combinaison de deux suites ML) Soient $(u_n)_{n\geq 0}$ et $(v_n)_{n\geq 0}$ deux suites de $\mathbb{F}_q$ avec $q = 2^8$, dont les polynômes caractéristiques f et g sont des polynômes primitifs de degré respectifs 7 et 8. Je considère alors la suite $(s_n)_{n\geq 0}$ obtenue en composant les deux suites précédentes par la fonction dont la forme algébrique normale est

$$f(x, y) = (x^{254} + y^{127} + 1)^r$$

La fonction f est donc 1-résiliente sur $\mathbb{F}_{2^8}$ si et seulement si $pgcd(r, q-1) = 1$. De plus, si $\binom{r}{3} \equiv 1 \bmod 2$, alors la forme algébrique normale de f contient un terme en $x^{254}y^{254}$ — pour $q = 2^8$, 256 exposants r conviennent.

D'après le théorème 7.23, la suite $(u_n^{254})_{n\geq 0}$ a pour complexité linéaire 7^7 et pour période q^7-1. Son polynôme caractéristique minimal n'a que des racines simples qui sont de la forme α^i où α est une racine de f, et donc un élément primitif de $\mathbb{F}_{q^7}$. De même la suite $(v_n^{254})_{n\geq 0}$ a pour complexité linéaire 8^7, pour période q^8-1, et les racines de son polynôme caractéristique minimal sont de la forme β^j où β est un élément primitif de $\mathbb{F}_{q^8}$. Les suites $(u_n^{254})_{n\geq 0}$ et $(v_n^{254})_{n\geq 0}$ vérifient donc les hypothèses de la proposition 7.17 ; la suite produit a donc une complexité linéaire $\Lambda = 8^7 7^7 \simeq 2^{40.65}$, et sa période vaut $(q^8-1)(q^7-1)/(q-1)^2 \simeq 2^{104}$ (cf. [Rue86, corollaire 5.10]). Les autres termes de la fonction f permettent d'augmenter encore ces deux valeurs.

Aussi la complexité linéaire de la suite $(s_n)_{n\geq 0}$ est-elle au moins $2^{40.65}$, ce qui signifie qu'une attaque par l'algorithme de Berlekamp-Massey requiert la connaissance d'au moins $2^{41.65}$ octets de la suite — ou du texte clair si cette suite est utilisée dans un chiffrement à flot. Sa période est, elle, égale à 2^{104}.

3 Non-linéarité d'une fonction sans corrélation

Nous venons de voir qu'une fonction de combinaison de LFSRs sur un corps fini $\mathbb{F}_q$, de caractéristique p, doit avoir à la fois un ordre de résilience élevé et un degré s en chacune de ses variables maximisant $w_p(s) = \sum_{i=0}^{e} s_i$ où $s = \sum_{i=0}^{e} s_i p^i$ est la décomposition de s en base p. Pour les fonction booléennes, ces conditions sont vérifiées lorsque le degré de la forme algébrique normale de la fonction et son ordre de résilience sont aussi élevés que possible. Malheureusement, il y a dans ce cas un compromis entre ces deux paramètres : un degré faible est en effet la contrepartie d'un ordre de résilience élevé. Ce compromis a été relevé par Siegenthaler [Sie84] qui a montré que, pour toute fonction booléenne $f : \mathbb{F}_2^n \to \mathbb{F}_2$, le degré d de la forme algébrique normale et l'ordre de non-corrélation t vérifient toujours $d+t \leq n$. J'exhibe ici une relation similaire pour les fonctions de $\mathbb{F}_q^n$ dans $\mathbb{F}_q^\ell$, qui découle en fait d'une propriété plus forte sur le degré de la forme algébrique normale en chacune des variables. Comme précédemment, j'identifie l'espace-vectoriel $\mathbb{F}_q^\ell$ au corps fini $\mathbb{F}_{q^\ell}$.

Théorème 7.25 *Soit f une fonction de $\mathbb{F}_q^n$ dans $\mathbb{F}_{q^\ell}$. Si f est sans corrélation d'ordre t (resp. t-résiliente) sur $\mathbb{F}_q$, la forme algébrique normale de f ne comporte aucun monôme de degré $q-1$ en plus de $n-t$ variables (resp. $n-t-1$ variables à condition que $q^\ell \neq 2$ ou $n \neq \ell + t$).*

preuve : Fixons j variables parmi $x_1, \ldots, x_n$; nous supposerons, sans restreindre la généralité du problème, qu'il s'agit des j dernières $x_{n-j+1} = x_{n-j+1}^\star$,

$\ldots, x_n = x_n^\star$. Par interpolation de Lagrange, on peut alors écrire la forme algébrique normale θ de la fonction f :

$$\theta(x_1,\ldots,x_{n-j},x^\star_{n-j+1},\ldots,x^\star_n)= \sum_{\alpha\in\mathbb{F}_q^{n-j}} -f(\alpha_1,\ldots,\alpha_{n-j},x^\star_{n-j+1},\ldots,x^\star_n)\prod_{i=1}^{n-j} L_{\alpha_i}(x_i) \tag{7.4}$$

Définissons alors les ensembles suivants

$$I_{1,\ldots,n-j}(v) = \{\alpha \in \mathbb{F}_q^{n-j}, f(\alpha_1,\ldots,\alpha_{n-j},x^\star_{n-j+1},\ldots,x^\star_n) = v\}$$

Si la fonction f est sans-corrélation d'ordre t sur $\mathbb{F}_q$, alors

$$\forall j \le t,\ \forall v \in \mathbb{F}_q^\ell,\ \ |I_{1,\ldots,n-j}(v)| = \frac{|f^{-1}(v)|}{q^j}$$

Aussi pour tout $j \le t$ et pour tout v de $\mathbb{F}_{q^\ell}$, a-t-on

$$|I_{1,\ldots,n-j}(v)| = q^{t-j}|I_{1,\ldots,n-t}(v)|$$

On en déduit par conséquent que, pour toutes les valeurs de v, et pour $j < t$, le cardinal des ensembles $I_{1,\ldots,n-j}(v)$ est nul dans $\mathbb{F}_{q^\ell}$ puisqu'il est multiple de q. Pour $j < t$, on peut alors écrire la relation 7.4 :

$$\theta(x_1,\ldots,x_{n-j},x^\star_{n-j+1},\ldots,x^\star_n) = -\sum_{v\in\mathbb{F}_{q^\ell}} v \sum_{\alpha\in I_{1,\ldots,n-j}(v)}\prod_{i=1}^{n-j} L_{\alpha_i}(x_i)$$

Comme chaque polynôme d'interpolation de Lagrange L_{α_i} est un polynôme unitaire de degré $q-1$, le coefficient du terme de degré $(q-1)(n-j)$ de $\sum_{\alpha\in I_{1,\ldots,n-j}(v)}\prod_{i=1}^{n-j} L_{\alpha_i}(x_i)$ est égal au cardinal de $I_{1,\ldots,n-j}(v)$ qui s'annule donc sur $\mathbb{F}_{q^\ell}$.

Ceci étant vrai pour tout choix de j variables parmi $x_1,\ldots,x_n$, avec $j < t$, cela implique que θ ne contient aucun produit de $n-t+1$ variables, ou plus, simultanément de degré $q-1$.

Dans le cas où f est, en outre, équilibrée, on peut déterminer de façon exacte le cardinal de $I_{1,\ldots,n-t}(v)$ puisque $|f^{-1}(v)| = q^{n-\ell}$ pour tout $v \in \mathbb{F}_{q^\ell}$. On a donc, en plus de la propriété précédente,

$$\forall v \in \mathbb{F}_{q^\ell},\ \ |I_{1,\ldots n-t}(v)| = q^{n-\ell-t}$$

$|I_{1,\ldots n-t}(v)|$ vaut donc zéro sur $\mathbb{F}_{q^\ell}$ lorsque $n-t-\ell > 0$. De plus, si $n = t+\ell$, alors $|I_{1,\ldots n-t}(v)| = 1$. Dans ce cas, le coefficient d'un monôme μ de θ tel que $deg_{x_i}\mu = q-1$ pour tous les $1 \le i \le n-t$ est alors égal à $-\sum_{v\in\mathbb{F}_{q^\ell}} v|I_{1,\ldots,n-t}(v)| = -\sum_{v\in\mathbb{F}_{q^\ell}} v = 0$ lorsque $q^\ell \neq 2$. Cela signifie donc que, quand $n \neq t+\ell$ ou $q^\ell \neq 2$, θ ne contient aucun produit de plus de $(n-t-1)$ variables simultanément de degré $q-1$. □

Remarque 7.26 *La démonstration précédente induit en fait une condition plus forte sur la forme algébrique normale de certains fonctions sans-corrélation d'ordre t, même si elles ne sont pas équilibrées : si la fonction* $f : \mathbb{F}_q^n \to \mathbb{F}_{q^\ell}$ *est sans-corrélation d'ordre* t *sur* $\mathbb{F}_q$ *et vérifie*

$$\forall y \in \mathbb{F}_{q^\ell}, \quad \frac{|f^{-1}(y)|}{q^t} \equiv 0 \bmod q$$

alors sa forme algébrique normale vérifie la propriété énoncée au théorème 7.25 pour les fonctions t-résilientes, c'est-à-dire qu'elle ne contient aucun monôme qui soit de degré $q-1$ *en plus de* $n-t-1$ *variables.*

On peut déduire de cette propriété une inégalité similaire à celle de Siegenthaler mettant en jeu le degré total de la fonction f :

Corollaire 7.27 *Soit* $f : \mathbb{F}_q^n \to \mathbb{F}_{q^\ell}$ *une fonction sans corrélation d'ordre* t *sur* $\mathbb{F}_q$*. Alors le degré total,* d*, de sa forme algébrique normale vérifie*

$$d + t \leq (q-1)n$$

Si, en plus, f *est équilibrée et* $n \neq \ell + t$ *ou* $q^\ell \neq 2$*, alors*

$$d + t \leq (q-1)n - 1$$

Il est en fait possible d'obtenir une condition plus restrictive sur l'ordre de non-corrélation d'une fonction f puisque l'on peut également faire intervenir le degré des formes algébriques normales de toutes les fonctions $p_2 \circ f \circ p_1$ avec $p_1 = (\pi_1, \ldots, \pi_n)$ et $p_2 = (\phi_1, \ldots, \phi_\ell)$ où les π_i et ϕ_i sont des permutations de $\mathbb{F}_q$. Il est en effet évident que toute fonction de la forme $p_2 \circ f \circ p_1$, avec f t-résiliente, est encore t-résiliente. Cette propriété apparaîtra par la suite comme un simple corollaire du théorème sur la composition de fonctions démontré au chapitre 9.

Remarque 7.28 *Soit* $f : \mathbb{F}_q^n \to \mathbb{F}_{q^\ell}$*. Alors son ordre de non-corrélation* t *sur* $\mathbb{F}_q$ *vérifie*

$$\delta + t \leq (q-1)n$$

où δ *est le plus grand degré des formes algébriques normales des fonctions* $p_2 \circ f \circ p_1$ *avec* $p_1 = (\pi_1, \ldots, \pi_n)$ *et* $p_2 = (\phi_1, \ldots, \phi_\ell)$*, quand les* π_i *et* ϕ_i *parcourent l'ensemble des permutations de* $\mathbb{F}_q$*. Si, en plus,* f *est équilibrée et* $n \neq \ell + t$ *ou* $q^\ell \neq 2$*, alors*

$$\delta + t \leq (q-1)n - 1$$

Exemple 7.29: Soit la fonction définie sur $\mathbb{F}_{16}$ par :

$$\begin{array}{rccc} f : & \mathbb{F}_{16} \times \mathbb{F}_{16} & \to & \mathbb{F}_{16} \\ & (x, y) & \mapsto & (x^{14} + y^7 + 1)^{11} \end{array}$$

Cette fonction est 1-résiliente sur $\mathbb{F}_{16}$: chaque exponentiation effectuée est une permutation du corps dans la mesure où les exposants utilisés sont premiers avec 15. En outre, sa forme normale algébrique est donnée par :

$$f(x,y) = x^{14}y^{14} + x^{14}y^{11} + x^{14}y^{10} + x^{13}y^{11} + x^{12}y^{11} + x^7y^{14} + x^{13}y^7 + x^6y^{14} + x^{13}y^3 + x^{14} + x^7y^7 + y^{14} + x^{13} + x^7y^6 + x^{12} + x^5y^7 + y^{11} + y^{10} + x^7 + y^7 + x^6 + y^6 + x^5 + x^4 + y^3 + y^2 + 1$$

On constate donc que, conformément au résultat du théorème 7.25, sa forme algébrique normale ne possède aucun terme contenant x^{15} ou y^{15}. De plus, cette fonction atteint la borne du corollaire 7.27 puisque la somme de son ordre de résilience et de son degré vaut 29.

On définit alors les fonctions sans corrélation d'ordre t et t-résilientes de *non-linéarité optimale* comme étant les fonctions dont la forme algébrique normale atteint les bornes du théorème 7.25 et du corollaire 7.27.

Définition 7.30 (fonction sans-corrélation de non-linéarité optimale) *Une fonction $f : \mathbb{F}_q^n \to \mathbb{F}_{q^\ell}$ sans corrélation d'ordre t (resp. t-résiliente) est de* non-linéarité optimale *si sa forme algébrique normale contient un monôme μ tel qu'il existe un ensemble d'indices $T \subset \{1,\ldots,n\}$ de cardinal $n-t$ (resp. $n-t-1$) vérifiant*

$$\forall i \in T,\ \ deg_{x_i}\mu = q-1 \text{ et } \forall j \notin T,\ \ deg_{x_j}\mu = a_j$$

où $w_p(a_j) = m(p-1)-1$.

Les fonction t-résilientes dont la non-linéarité est optimale, utilisées comme fonctions de combinaison de plusieurs LFSRs, permettent alors de construire des générateurs pseudo-aléatoires ayant une complexité linéaire très élevée. En effet, la suite-produit résultant de la combinaison des sorties de n LFSRs par une fonction dont la forme algébrique normale est un monôme μ ne contenant pas plus de $n-t$ variables simultanément de degré $q-1$, n'atteint la complexité linéaire maximale que si

$$\exists T \subset \{1,\ldots,n\}, |T| = n-t-1,\ \ \forall i \in T,\ \ deg_{x_i}\mu = q-1$$
$$\forall j \notin T,\ \ deg_{x_j}\mu = a_j \text{ avec } w_p(a_j) = m(p-1)-1$$

On voit par exemple que $a_j = q-2$ convient. Aussi toute fonction t-résiliente dont le forme algébrique normale contient un monôme de degré $q-1$ en $(n-t-1)$ variables et de degré $q-2$ en les autres variables est-elle une fonction t-résiliente de non-linéarité optimale.

4 Construction de fonctions résilientes de non-linéarité optimale

Je construis ici, pour certains corps finis $\mathbb{F}_q$, des fonctions $f : \mathbb{F}_q^n \to \mathbb{F}_q$, t-résilientes dont la non-linéarité est optimale. Je donne en particulier une famille

de fonctions $f : \mathbb{F}_{2^m}^n \to \mathbb{F}_{2^m}$ t-résilientes et de non-linéarité optimale pour toutes valeurs de n et t lorsque m est impair. Ces fonctions sont particulièrement bien adaptées à la combinaison de LFSRs puisque la complexité linéaire du générateur pseudo-aléatoire résultant de cet assemblage est maximale.

La construction que je propose est une construction par récurrence dans la mesure où une fonction t-résiliente à n variables est obtenue récursivement à partir d'une fonction t-résiliente à $(t+1)$ variables, dont la forme algébrique normale est donnée explicitement.

Je vais donc tout d'abord construire des fonctions à n variables et $(n-1)$-résilientes sur le corps fini $\mathbb{F}_q$, ou autrement dit des tableaux orthogonaux d'indice unité, de taille q^{n-1} et à n contraintes. J'utiliserai pour cela le lemme suivant :

Lemme 7.31 *Notons $\mathcal{A}$ l'algèbre $\mathbb{F}_q[x]/(x^q-x)$, avec $q > 2$. Dans cette algèbre, on a $deg(x^{i(q-2)}) < q-2$, pour toute valeur de i, $2 \leq i < q-1$. Et, si q est pair, on a $deg(x^{j\frac{q-2}{2}}) < q-2$ pour toute valeur de j, $3 \leq j \leq q-2$.*

preuve : Comme $i \leq q-2$, on peut dans un premier temps écrire la suite d'égalités suivantes :

$$x^{i(q-2)} = x^{q+q(i-1)-2i} = x^{q+(i-1)-2i} = x^{q-i-1}$$

La valeur de i étant supérieure à 2, on en déduit donc que le degré de ce monôme est au plus $q-3$.

Considérons maintenant le monôme $x^{j\frac{q-2}{2}}$ et $j = 2a+b$ avec $b \in \{0,1\}$. Dans le cas où $b = 0$, on peut appliquer le résultat précédent puisque $2 \leq a \leq q-2$. Dans le cas où b vaut 1, on a $2 \leq a < \frac{q-2}{2}$. On peut donc écrire comme précédemment les égalités suivantes :

$$x^{j\frac{q-2}{2}} = x^{a(q-2)+\frac{q-2}{2}} = x^{q-a-1+\frac{q-2}{2}} = x^{\frac{q-2}{2}-a}$$

Ceci implique donc que le degré du monôme $x^{j\frac{q-2}{2}}$ est égal à $\frac{q-2-2a}{2} < q-2$. □

Proposition 7.32 *Soit $q = 2^m$ avec m impair. Alors pour tout $n > 1$, il existe une fonction $f : \mathbb{F}_{2^m}^n \to \mathbb{F}_{2^m}$ $(n-1)$-résiliente dont la non-linéarité est optimale.*

preuve : Nous donnons une preuve constructive par récurrence sur le nombre de variables n.

– Dans le cas où $n = 2$, nous considérons la fonction f définie sur $\mathbb{F}_{2^m}$, pour $m \neq 1$ par

$$f(x,y) = (x^{q-2} + y^{\frac{q-2}{2}})^3$$

La condition m impair implique que $q-1$ et 3 sont premiers entre eux et, comme $pgcd(q-2, q-1) = 1$, que f est une fonction 1-résiliente puisque toutes les exponentiations utilisées sont des permutations du corps fini $\mathbb{F}_{2^m}$. Du reste, le coefficient du terme $(xy)^{q-2}$ de sa forme algébrique normale est égal à 1. La fonction f possède donc une non-linéarité optimale.

– Supposons qu'il existe une fonction $g : \mathbb{F}_{2^m}^n \to \mathbb{F}_2$, avec $m \neq 1$ qui soit $(n-1)$-résiliente et qui contienne le monôme $(x_1 \dots x_n)^{q-2}$ précédé d'un coefficient non nul. Considérons alors la fonction f à $n+1$ variables définie par

$$f(x_1, \dots, x_{n+1}) = (g(x_1, \dots, x_n) + x_{n+1}^{\frac{q-2}{2}})^3$$

Par le même raisonnement que précédemment, on montre que f est n-résiliente, et le coefficient du terme $(x_1 \dots x_{n+1})^{q-2}$ vaut également 1. La fonction f est donc de non-linéarité optimale.

– Lorsque $m = 1$, la proposition est également vérifiée : d'après le théorème 7.25, une fonction $f : \mathbb{F}_2^n \to \mathbb{F}_2$ $(n-1)$-résiliente est de non-linéarité optimale si elle contient un monôme de degré 1 en une seule de ses variables. La fonction $f(x_1, \dots, x_n) = x_1 + \dots + x_n$ vérifie alors ces conditions.

□

On peut également construire de telles fonctions sur un corps fini $\mathbb{F}_q$ de caractéristique $p > 3$ et $q \not\equiv 1 \bmod 3$, dans le cas où le nombre de variables n est impair :

Proposition 7.33 *Soit $q = p^m$ avec $p > 3$ et $q \not\equiv 1 \bmod 3$. Alors, pour tout $n > 1$ impair, il existe une fonction $f : \mathbb{F}_q^n \to \mathbb{F}_q$ $(n-1)$-résiliente dont la non-linéarité est optimale.*

preuve : Nous donnons une construction récursive similaire à celle de la proposition précédente.

– Pour $n = 3$, on considère la fonction f définie par

$$f(x, y, z) = (x^{q-2} + y^{q-2} + z^{q-2})^3$$

Comme $q \not\equiv 1 \bmod 3$, elle est 2-résiliente et le coefficient du terme $(xyz)^{q-2}$ est égal à 6 ; par conséquent, il ne s'annule pas pour p impair strictement supérieur à 3.

– Supposons qu'il existe une fonction $g : \mathbb{F}_q^{2r-1} \to \mathbb{F}_q$ $(2r-2)$-résiliente et de non-linéarité optimale, *i.e.* qu'elle contient le monôme $(x_1 \dots x_{2r-1})^{q-2}$ précédé d'un coefficient non nul. Considérons alors la fonction f à $2r+1$ variables définie par

$$f(x_1, \dots, x_{2r+1}) = (g(x_1, \dots, x_{2r-1}) + x_{2r}^{q-2} + x_{2r+1}^{q-2})^3$$

La fonction f est donc $2r$-résiliente ; le coefficient du terme $(x_1 \dots x_{2r+1})^{q-2}$ vaut également 6 fois le coefficient non nul.

□

Il est alors très aisé d'obtenir à partir de ces fonctions des fonctions à n variables, t-résilientes et de non-linéarité optimale grâce à la proposition suivante.

Proposition 7.34 *Soient f_1 et f_2 deux fonctions t-résilientes de $\mathbb{F}_q^n$ dans $\mathbb{F}_q$, de non-linéarité optimale et telles que $deg(f_1 - f_2) = deg(f_1)$. Alors, si $q \neq 2$ ou $t \neq n-1$, la fonction $g : \mathbb{F}_q^{n+1} \to \mathbb{F}_q$ définie par*

$$g(x_1, \ldots, x_{n+1}) = x_{n+1}^{q-1} f_1(x_1, \ldots, x_n) + (1 - x_{n+1}^{q-1}) f_2(x_1, \ldots, x_n)$$

est une fonction t-résiliente de non-linéarité optimale.

preuve : Par définition, nous avons:
$\forall x_{n+1} \neq 0, g(x_1,\ldots,x_{n+1}) = f_1(x_1,\ldots,x_n)$ et $g(x_1,\ldots,x_n,0) = f_2(x_1,\ldots,x_n)$.
Soit T un ensemble de i variables avec $i \leq t$. Les fonctions f_1 et f_2 étant toutes deux t-résilientes, on peut écrire, pour tout a dans $\mathbb{F}_q$, la succession d'égalités suivantes, lorsque $x_{n+1} \notin T$:

$$\begin{aligned}
&Pr[g(x_1, \ldots, x_{n+1}) = a/T = \alpha] = \\
&Pr[f_1(x) = a/T = \alpha] Pr[x_{n+1} \neq 0] + Pr[f_2(x) = a/T = \alpha] Pr[x_{n+1} = 0] = \\
&Pr[f_1(x) = a] Pr[x_{n+1} \neq 0] + Pr[f_2(x) = a] Pr[x_{n+1} = 0] = Pr[g(x) = a]
\end{aligned}$$

où x désigne le n- $(x_1, \ldots, x_n)$. Si l'on considère maintenant un ensemble T tel que $T = T' \cup \{x_{n+1}\}$, alors

$$\begin{aligned}
\forall \beta \neq 0, \quad Pr[g(x) = a/T' = \alpha', x_{n+1} = \beta] &= Pr[f_1(x_1, \ldots, x_n) = a/T' = \alpha] \\
&= Pr[f_1(x_1, \ldots, x_n) = a] = \frac{1}{q}
\end{aligned}$$

$$\begin{aligned}
\text{et} \quad Pr[g(x) = a/T' = \alpha', x_{n+1} = 0] &= Pr[f_2(x_1, \ldots, x_n) = a/T' = \alpha)] \\
&= Pr[f_2(x_1, \ldots, x_n) = a] = \frac{1}{q}
\end{aligned}$$

Aussi la fonction g vérifie-t-elle, pour tout sous-ensemble T d'au plus t variables

$$Pr[g(x) = a/T = \alpha] = Pr[g(x) = a]$$

En outre, sa forme algébrique normale est donnée par

$$g(x_1, \ldots, x_{n+1}) = x_{n+1}^{q-1}[f_1(x_1, \ldots, x_n) + f_2(x_1, \ldots, x_n)] + f_2(x_1, \ldots, x_n)$$

Comme f_1 et f_2 sont de non-linéarité optimale et que, par hypothèse, leurs termes de plus haut degré ne se compensent pas, la fonction g est égale de non-linéarité optimale.

□

A partir de ces 3 propositions, on peut alors construire une fonction à n variables, t-résiliente et de non-linéarité optimale dans les cas suivants :

Théorème 7.35 *Soit $q = p^m$ avec $q \not\equiv 1 \bmod 3$ et $p \neq 3$. Alors, pour tout $n > 1$, il existe une fonction $f : \mathbb{F}_q^n \to \mathbb{F}_q$ t-résiliente et de non-linéarité optimale pour tout $t < n$ si $p = 2$ et pour tout $t < n$ pair sinon.*

preuve : Les propositions 7.32 et 7.33 permettent de construire une fonction $g : \mathbb{F}_q^{t+1} \to \mathbb{F}_q$, t-résiliente et de non-linéarité optimale pour toute valeur de t quand $p = 2$ et pour tout t pair sinon. Il suffit ensuite d'appliquer la proposition 7.34 avec $f_1 = g$ et $f_2 = \alpha g$ où $\alpha \in \mathbb{F}_q \setminus \{0, 1\}$ pour obtenir une fonction à $t + 2$ variables, t-résiliente et de non-linéarité optimale. En itérant cette opération $n-t-1$ fois, on obtient une fonction à n variables, t-résiliente et de non-linéarité optimale.

Le cas booléen, $q = 2$, a été prouvé par Siegenthaler [Sie84]. □

Exemple 7.36: Je construis ici une fonction 2-résiliente à 4 variables sur $\mathbb{F}_8$. La proposition 7.32 me permet de construire deux fonctions g_1 et g_2 de non-linéarité optimale, qui sont respectivement 1-résiliente à 2 variables et 2-résiliente à 3 variables.

$g_1(x_1, x_2) = x_1^6 x_2^6 + x_1^5 x_2^3 + x_1^4 + x_2^2$

$g_2(x_1, x_2, x_3) = x_1^6 x_2^6 x_3^6 + x_1^5 x_2^3 x_3^6 + x_1^4 x_2^5 x_3^4 + x_1^5 x_2^7 + x_1^4 x_2^7 x_3 + x_1^2 x_2^6 x_3^4 + x_1^7 x_2^2 x_3 + x_1^4 x_3^6 + x_1^6 x_2^3 + x_1^5 x_2^3 x_3 + x_1^4 x_2^4 + x_1^2 x_2^4 x_3^2 + x_2^2 x_3^6 + x_2^4 x_3^3 + x_2^6 + x_1^4 x_3 + x_1^2 x_2 x_3 + x_1 x_2 x_3^2 + x_3^4 + x_2^2 x_3 + x_3^2$

J'applique ensuite la proposition 7.34 avec $f_1 = g_2$ et $f_2 = \alpha g_2$ où α est un élément de $\mathbb{F}_8 \setminus \{0, 1\}$. On obtient ainsi une fonction 2-résiliente à 4 variables, dont la non-linéarité est optimale.

$$
\begin{aligned}
f(x_1, x_2, x_3, x_4) &= (\alpha+1)x_1^6 x_2^6 x_3^6 x_4^7 + (\alpha+1)x_1^5 x_2^3 x_3^6 x_4^7 \\
&+(\alpha+1)x_1^4 x_2^5 x_3^4 x_4^7 + (\alpha+1)x_1^5 x_2^7 x_4^7 + (\alpha+1)x_1^4 x_2^7 x_3 x_4^7 \\
&+(\alpha+1)x_1^2 x_2^6 x_3^4 x_4^7 + \alpha x_1^6 x_2^6 x_3^6 + (\alpha+1)x_1^7 x_2^2 x_3 x_4^7 \\
&+(\alpha+1)x_1^4 x_3^6 x_4^7 + (\alpha+1)x_1^6 x_2^3 x_4^7 + (\alpha+1)x_1^5 x_2^3 x_3 x_4^7 \\
&+(\alpha+1)x_1^4 x_2^4 x_4^7 + (\alpha+1)x_1^2 x_2^4 x_3^2 x_4^7 + (\alpha+1)x_2^2 x_3^6 x_4^7 \\
&+\alpha x_1^5 x_2^3 x_3^6 + (\alpha+1)x_2^4 x_3^3 x_4^7 + \alpha x_1^4 x_2^5 x_3^4 + (\alpha+1)x_2^6 x_4^7 \\
&+\alpha x_1^5 x_2^7 + \alpha x_1^4 x_2^7 x_3 + (\alpha+1)x_1^4 x_3 x_4^7 + \alpha x_1^2 x_2^6 x_3^4 \\
&+(\alpha+1)x_1^2 x_2 x_3 x_4^7 + (\alpha+1)x_1 x_2 x_3^2 x_4^7 + (\alpha+1)x_3^4 x_4^7 \\
&+\alpha x_1^7 x_2^2 x_3 + \alpha x_1^4 x_3^6 + (\alpha+1)x_2^2 x_3 x_4^7 + \alpha x_1^6 x_2^3 + \alpha x_1^5 x_2^3 x_3 \\
&+(\alpha+1)x_3^2 x_4^7 + \alpha x_1^4 x_2^4 + \alpha x_1^2 x_2^4 x_3^2 + \alpha x_2^2 x_3^6 + \alpha x_2^4 x_3^3 \\
&+\alpha x_2^6 + \alpha x_1^4 x_3 + \alpha x_1^2 x_2 x_3 + \alpha x_1 x_2 x_3^2 + \alpha x_3^4 + \alpha x_2^2 x_3 + \alpha x_3^2
\end{aligned}
$$

Le degré de la forme algébrique normale de cette fonction est égal à 25 et vérifie donc bien $d + t = 7 * 4 - 1$.

5 Non-linéarité d'une fonction q-aire sans corrélation sur un sur-corps

Je m'intéresse ici aux propriétés de la forme algébrique normale d'une fonction f de $\mathbb{F}_{q^k}^n$ dans $\mathbb{F}_q$, sans corrélation d'ordre t sur $\mathbb{F}_{q^k}$.

Théorème 7.37 *Soit f une fonction de $\mathbb{F}_{q^k}^n$ dans $\mathbb{F}_q$ avec $k > 1$ Sa forme algébrique normale est alors un polynôme θ en kn variables dans l'algèbre $\mathcal{A} = \mathbb{F}_q[x_{i,j},\ 1 \leq i \leq n,\ 0 \leq j \leq k-1] / \left((x_{i,j}^q - x_{i,j})_{\substack{1 \leq i \leq n \\ 0 \leq j \leq k-1}} \right)$.*
Si f est sans corrélation d'ordre t sur $\mathbb{F}_{q^k}$ (resp. t-résiliente sur $\mathbb{F}_{q^k}$), alors θ ne comporte aucun monôme de degré $q-1$ en plus de $(kn-t)$ variables (resp. $(kn-t-1)$ variables).

preuve : Considérons tout d'abord f comme une fonction de $\mathbb{F}_{q^k}^n$ à valeurs dans $\mathbb{F}_{q^k}$. Sa forme algébrique normale est alors un polynôme μ de l'algèbre $\mathbb{F}_{q^k}[x_1, \ldots, x_n]/(x_1^{q^k} - x_1, \ldots, x_n^{q^k} - x_n)$. Soit α un élément primitif de $\mathbb{F}_{q^k}$; alors $\mathbb{F}_{q^k} = \mathbb{F}_q + \alpha\mathbb{F}_q + \ldots + \alpha^{k-1}\mathbb{F}_q$. On peut par conséquent représenter tout élément x_i de $\mathbb{F}_{q^k}$ par un polynôme en $(x_{i,0}, \ldots, x_{i,k-1})$ de l'algèbre $\mathbb{F}_{q^k}[x_{i,0}, \ldots, x_{i,n}]/(x_{i,0}^q - x_{i,0}, \ldots, x_{i,k-1}^q - x_{i,k-1})$. La fonction f peut donc s'écrire sous forme d'une fonction polynôme θ de l'algèbre $\mathbb{F}_{q^k}[x_{i,j},\ 1 \leq i \leq n,\ 0 \leq j \leq k-1]/(x_{i,j}^q - x_{i,j})$, quotientée par tous idéaux $(x_{i,j}^q - x_{i,j})$ pour $1 \leq i \leq n$ et $0 \leq j \leq k-1$. Comme la fonction f est à valeurs dans le corps $\mathbb{F}_q$, on a $\theta^q(x) = \theta(x)$ pour tous les x de $\mathbb{F}_{q^k}^n$. Grâce au lemme 7.2, on en déduit que $\theta^q = \theta$, et donc que tous les coefficients de θ sont dans $\mathbb{F}_q$.

Exprimons maintenant x_i^s sous forme d'un polynôme en $x_{i,0}, \ldots, x_{i,k-1}$, pour un entier $s \leq q^k - 1$, dont la décomposition en base q est $s = \sum_{j=0}^{k-1} s_j q^j$. On obtient donc

$$\begin{aligned} x_i^s &= \prod_{j=0}^{k-1} (x_{i,0} + \alpha x_{i,1} + \ldots + \alpha^{k-1} x_{i,k-1})^{s_j q^j} \\ &= \prod_{j=0}^{k-1} (x_{i,0} + \alpha^{q^j} x_{i,1} + \ldots + \alpha^{(k-1)q^j} x_{i,k-1})^{s_j} \end{aligned}$$

Le polynôme représentant x_i^s contient donc un monôme de degré $q-1$ en r variables uniquement si la décomposition de s en base q contient r coefficients s_i égaux à $q-1$. Ainsi, seul le polynôme représentant $x_i^{q^k-1}$ peut contenir un monôme de degré $q-1$ en tous les $x_{i,j}$, et pour $s < q^k - 1$, le polynôme représentant x_i^s contient au mieux un monôme de degré $q-1$ en $k-1$ variables.

Or, d'après le théorème 7.25, le polynôme μ ne contient aucun monôme de degré $q-1$ en plus de $n-t$ variables. Donc le polynôme θ ne contient aucun monôme de degré $q-1$ en plus de $k(n-t) + (k-1)t$ variables, c'est-à-dire $kn - t$ variables.

Si f est en plus une fonction équilibrée, on a $|f^{-1}(v)| = q^{nk-1}$ pour tout $v \in \mathbb{F}_q$. Comme $k > 1$ et $n > t$, $\frac{|f^{-1}(v)|}{q^{kt}} \equiv 0 \bmod q$. La remarque 7.26 implique alors, par un raisonnement similaire au précédent, que θ ne contient aucun monôme de degré $q-1$ en plus de $k(n-t-1)+(k-1)(t+1)$ variables, c'est-à-dire $kn-t-1$ variables. □

Remarque 7.38 *Comme pour le théorème 7.25, la démonstration implique en réalité une condition moins restrictive pour obtenir la propriété énoncée pour les fonctions t-résilientes. En effet, si la fonction $f : \mathbb{F}_{q^k}^n \to \mathbb{F}_q$ est sans corrélation d'ordre t sur $\mathbb{F}_{q^k}$ et vérifie*

$$\forall v \in \mathbb{F}_q, \quad \frac{|f^{-1}(v)|}{q^{kt}} \equiv 0 \bmod q$$

alors sa forme algébrique normale satisfait la propriété énoncée au théorème 7.37 pour les fonctions t-résilientes, c'est-à-dire qu'elle ne contient aucun monôme qui soit de degré $q-1$ en plus de $(kn-t-1)$ variables.

Ce résultat induit également une inégalité similaire à celle de Siegenthaler sur le degré total de la forme algébrique normale des fonctions de $\mathbb{F}_{q^k}^n$ dans $\mathbb{F}_q$, sans corrélation d'ordre t et t-résilientes sur $\mathbb{F}_{q^k}$.

Corollaire 7.39 *Soit $f : \mathbb{F}_{q^k}^n \to \mathbb{F}_q$ une fonction sans corrélation d'ordre t sur $\mathbb{F}_{q^k}$. Alors le degré total d de sa forme algébrique normale vérifie*

$$d + t \leq (q-1)kn$$

Si, en plus, f est équilibrée, et $k > 1$, alors

$$d + t \leq (q-1)kn - 1$$

Exemple 7.40: Considérons la fonction définie sur $\mathbb{F}_8$

$$\begin{array}{rccc} \Phi\colon & \mathbb{F}_8 \times \mathbb{F}_8 & \to & \mathbb{F}_8 \\ & (x,y) & \mapsto & (x^3+y^3)^3 \end{array}$$

Soit α une racine de X^3+X+1. A chaque élément x de $\mathbb{F}_8$ on associe le polynôme $x_0+\alpha x_1+\alpha^2 x_2$. On peut alors appréhender la fonction Φ comme une fonction de $\mathbb{F}_2^6$ dans $\mathbb{F}_2^3$ dont les composantes booléennes f_0, f_1 et f_2 sont définies par $\Phi = f_0 + \alpha f_1 + \alpha f_2$. Les fonctions f_i sont par conséquent des fonctions booléennes à 6 variables et 1-résilientes sur $\mathbb{F}_8$. D'après le théorème précédent, elles ne peuvent donc pas contenir de produit de plus de $3 \times 2 - 1 - 1 = 4$ variables. En explicitant leurs formes algébriques normales respectives, on voit donc que chacune des ces fonctions est de non-linéarité optimale.

$f_0(x_0;x_1;x_2;y_0;y_1;y_2) = x_0+y_0+x_1y_2+x_2y_1+x_0x_1y_1+x_1y_0y_1+x_2y_0y_1+x_0x_1y_2+x_2y_0y_2+x_0x_2y_2+x_0x_2y_0y_1+x_0x_1y_0y_2$

$f_1(x_0;x_1;x_2;y_0;y_1;y_2) = x_2 + y_2 + x_0y_1 + x_1y_0 + x_0y_2 + x_2y_0 + x_0x_2y_1 + x_1y_0y_2 + x_2y_1y_2 + x_1x_2y_2 + x_0x_2y_2 + x_2y_0y_2 + x_1y_1y_2 + x_1x_2y_1 + x_0y_0y_2 + x_0x_2y_0 + x_0x_2y_1y_2 + x_1x_2y_0y_2$

$f_2(x_0;x_1;x_2;y_0;y_1;y_2) = x_1 + y_1 + x_2 + y_2 + x_2y_1 + x_1y_2 + x_0y_1 + x_1y_0 + x_0x_1y_0 + x_0y_0y_1 + x_2y_0y_1 + x_0x_1y_2 + x_0x_2y_0 + x_0y_0y_2 + x_0x_2y_1 + x_1y_0y_2 + x_0x_1y_1 + x_1y_0y_1 + x_1x_2y_1 + x_1y_1y_2 + x_0x_2y_2 + x_2y_0y_2 + x_1x_2y_0y_1 + x_0x_1y_1y_2 + x_0x_2y_1y_2 + x_1x_2y_0y_2$

Chapitre 8

Autres applications des fonctions sans corrélation en cryptographie

La construction de générateurs pseudo-aléatoires par combinaison de plusieurs registres à décalage à rétroaction linéaire constitue historiquement la première utilisation des fonctions sans corrélation en cryptographie. Toutefois, depuis les travaux de Siegenthaler, bien d'autres applications virent le jour et l'on constate maintenant que la théorie des fonctions sans corrélation et résiliente est omniprésente en cryptographie.

Ces propriétés sont d'une part utilisées pour construire des générateurs pseudo-aléatoires dans l'optique d'un chiffrement à flot. D'un point de vue plus général que celui que j'ai développé jusqu'à présent, il est en effet souhaitable de concevoir des générateurs pseudo-aléatoires dont les propriétés cryptographiques permettent de garantir la sécurité du chiffrement à flot correspondant. Ceci est en général impossible puisque la suite qui constitue la clef du système ne peut être complètement aléatoire. Cependant, Ueli Maurer et Jim Massey [MM90, MM91] ont défini une classe de générateurs pseudo-aléatoires, fondés sur les fonctions sans corrélation, qui induisent une parfaite sécurité du chiffrement à flot — ce chiffrement est alors *provably-secure* — sous certaines conditions ; il s'agit de la notion de *générateur aléatoire localement parfait*.

D'autre part, il est maintenant connu, depuis les travaux de Claus Schnorr et Serge Vaudenay [SV95], que l'utilisation de fonctions sans corrélation est primordiale dans les primitives cryptographiques dites *conventionnelles*. On regroupe sous ce terme toutes les primitives cryptographiques composées d'un réseau dont chaque sommet est une petite boîte de calcul, tels certains systèmes de chiffrement à clef secrète comme le DES ou la plupart des fonctions de hachage (MD4, MD5, FFT-Hashing, ...). Pendant longtemps, le principal critère utilisé pour construire de telles primitives était simplement que le réseau soit suffisamment inextricable et les boîtes suffisamment irrégulières pour que l'on ne parvienne pas à avoir une vue d'ensemble de la fonction. Quelques

travaux récents induisent cependant une théorisation du problème : il apparaît notamment que, parmi les boîtes du réseau, doivent figurer des boîtes réalisant une *confusion* parfaite afin de dissimuler toute structure (algébrique ou statistique) des données, ainsi que des boîtes réalisant une *diffusion* parfaite afin qu'une modification même minime de l'entrée de la fonction soit répercutée dans la sortie. L'utilisation de ces dernières boîtes permet entre autres de se prémunir contre certaines attaques par collision. La propriété de confusion parfaite est assurée par les *fonctions courbes*, qui sont aussi éloignées que possibles des fonctions affines [Rot76, Car94] ; celle de diffusion parfaite est, elle, obtenue grâce à des fonctions introduites par Schnorr et Vaudenay sous le nom de *multipermutations*, qui sont très liées aux fonctions sans corrélation.

Je verrai que l'on peut donc appliquer certains des résultats précédents sur les fonctions sans corrélation aux multipermutations, dans la mesure où l'on peut associer à toute multipermutation π de $\mathbb{F}_{2^m}^r$ dans $\mathbb{F}_{2^m}^n$ une fonction f_π sans corrélation d'ordre r sur $\mathbb{F}_{2^m}$. J'étudierai en particulier l'ordre de non-corrélation, t, de cette fonction f_π sur $\mathbb{F}_2$ puisqu'une valeur élevée de t permet d'éviter certaines attaques par collision si l'on considère les entrées de la boîte comme des chaînes binaires et non plus comme des éléments de $\mathbb{F}_{2^m}$. Je donnerai en particulier une borne supérieure sur cet ordre t en fonction du degré des composantes binaires de la multipermutation, vues comme des fonctions booléennes.

1 Générateurs aléatoires localement parfaits

1.1 Définition et motivations cryptographiques

Aucun système de chiffrement n'est parfaitement sûr en théorie puisque l'on peut toujours, par exemple, procéder à une recherche exhaustive de la clef utilisée. Toutefois, la possibilité d'une telle attaque importe rarement puisqu'elle n'est généralement pas réalisable, si l'espace des clefs est suffisamment grand. Il est en revanche primordial de se prémunir contre toute attaque réalisable dans la pratique. Le critère de "faisabilité" communément envisagé est le temps de calcul, mais on peut également considérer l'espace-mémoire requis, le nombre de digits du texte clair auxquels l'attaquant peut accéder lors d'une attaque à clair-connu, ... Il est alors intéressant de pouvoir garantir qu'un système de chiffrement résiste à toute attaque "réalisable", en spécifiant le critère de faisabilité considéré ; on dira alors qu'il est *provably-secure*.

Le premier chiffrement à flot provably-secure a été introduit par Schnorr [Sch88] : il utilise un générateur pseudo-aléatoire dont la sécurité n'est pas fondée, à la différence de la plupart des générateurs, sur certaines hypothèses non démontrées, telle la difficulté de certains problèmes mathématiques ..., mais est au contraire prouvée si l'on suppose que l'attaquant n'a accès qu'à un nombre limité de digits du texte clair. Maurer et Massey [MM90, Mau90, MM91]ont ensuite généralisé ce travail en définissant une classe de générateurs pseudo-aléatoires, les *générateurs aléatoires localement parfaits*, conduisant à un chif-

frement à flot provably-secure sous une condition du même type. Ce critère de faisabilité est évidemment bien plus restrictif qu'une restriction sur la puissance de calcul disponible, et souvent moins réaliste. Cette propriété est malgré tout digne d'intérêt puisqu'il n'existe aucun chiffrement à flot dont on puisse garantir la sécurité en supposant que la puissance de calcul de l'ennemi est limitée.

Je généralise ici la définition de générateur aléatoire localement parfait donnée par Maurer et Massey dans le cas binaire, au cas d'un alphabet quelconque à q éléments.

Définition 8.1 (générateur aléatoire localement parfait) [MM90] *Soit $\mathcal{F}$ un alphabet à q éléments. Une fonction $f : \mathcal{F}^k \to \mathcal{F}^n$, où $k < n$, est un* (k, n)-générateur aléatoire localement parfait *d'ordre t sur $\mathcal{F}$ si, lorsque les k digits de l'entrée sont k variables aléatoires indépendantes uniformément distribuées, alors tout ensemble de t-digits de la sortie est un ensemble de t variables aléatoires indépendantes uniformément distribuées.*

Cela signifie donc que la connaissance de t digits de la suite produite par un tel générateur ne suffit pas pour déduire une quelconque information sur n'importe quel autre digit de la sortie. Le chiffrement à flot utilisant un générateur aléatoire localement parfait d'ordre t est, par conséquent, provably-secure si l'on suppose qu'un attaquant ne peut pas accéder à plus de t bits du texte clair lors d'une attaque à clair connu.

1.2 Lien avec les fonctions sans corrélation

Comme l'ont montré Maurer et Massey, la propriété de générateur aléatoire localement parfait peut s'exprimer en termes de tableaux orthogonaux : sur un alphabet $\mathcal{F}$ à q éléments, une fonction f de $\mathcal{F}^k$ dans $\mathcal{F}^n$, avec $k < n$, est un (k, n)-générateur aléatoire localement parfait d'ordre t sur $\mathcal{F}$ si et seulement si le tableau dont les lignes sont les vecteurs $f(x),\ x \in \mathcal{F}^k$ est un tableau orthogonal de taille q^k, à n contraintes et de force t sur $\mathcal{F}$.

Les résultats du chapitre 6 concernant les tableaux orthogonaux peuvent donc être appliqués aux générateurs aléatoires localement parfaits. On peut par exemple construire un (k, n)-générateur aléatoire localement parfait d'ordre t à l'aide d'un code de distance duale $t + 1$. La section 6 et les tables 6.2, 6.3, 6.4 induisent également des bornes sur l'ordre maximal que peut atteindre un (k, n)-générateur aléatoire localement parfait sur un alphabet $\mathcal{F}$.

Je résume finalement ces propriétés à travers la proposition suivante, qui donne plusieurs définitions équivalentes de la propriété de générateur aléatoire localement parfait :

Proposition 8.2 *Soit $\mathcal{F}$ un alphabet à q éléments. Les assertions suivantes sont équivalentes :*

1. *La fonction $f : \mathcal{F}^k \to \mathcal{F}^n$, avec $k < n$, est un (k, n)-générateur aléatoire localement parfait d'ordre t sur $\mathcal{F}$.*

2. *Le tableau dont les lignes sont les vecteurs* $(f(x))_{x\in\mathcal{F}^k}$ *est un tableau orthogonal de taille* q^k*, à* n *contraintes et de force* t *sur* $\mathcal{F}$*.*

3. *La fonction* $F : \mathcal{F}^n \to \mathbb{F}_2$ *définie par* $F(x) = 1$ *ssi* $x \in f(\mathcal{F}^k)$ *est une fonction sans corrélation d'ordre* t *sur* $\mathcal{F}$*.*

4. *La fonction* f *est la fonction d'encodage d'un code de longueur* n*, de taille* q^k *et de distance duale* $t+1$ *sur* $\mathcal{F}$*, à condition que* $\mathcal{F}$ *soit muni d'une structure de groupe abélien.*

2 Multipermutations

2.1 Définition et motivations cryptographiques

Je m'intéresse ici à la sécurité des primitives cryptographiques conventionnelles telles que les a définies Serge Vaudenay [Vau95a], *i.e.* décrites par un graphe orienté dont les sommets et les feuilles sont respectivement les entrées et les sorties de la fonction et dont les noeuds sont des boîtes de calcul. On trouve dans cette catégorie de nombreux systèmes de chiffrement par blocs (DES, IDEA, SAFER) et fonctions de hachage (MD4, MD5, FFT-Hashing).

Depuis les travaux de Shannon [Sha49], on distingue deux types de boîtes suivant la façon dont elle affecte les données :

– **les boîtes de confusion** qui servent à faire disparaître toute structure particulière, algébrique ou statistique, des données. Ainsi, la boîte de la figure 8.1 est une boîte de confusion dans la mesure où elle réalise une permutation de l'alphabet qui permet de dissimuler toute structure linéaire — il s'agit d'une des boîtes de confusion du système de chiffrement à clef secrète SAFER inventé par Jim Massey [Mas94].

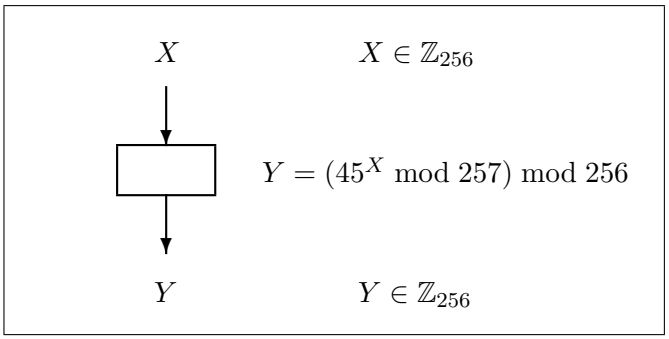


FIG. 8.1 –: *Exemple de boîte de confusion*

– **les boîtes de diffusion** qui fusionnent plusieurs données tout en propageant toute modification, même minime, des entrées dans leurs sorties. La

boîte de la figure 8.2 est donc, elle, une boîte de diffusion. On utilise de telles boîtes dans les systèmes de chiffrement, notamment afin de propager l'information contenue dans le texte clair et dans la clef, et également dans les fonctions de hachage puisqu'elles permettent d'éviter certaines attaques par collision.

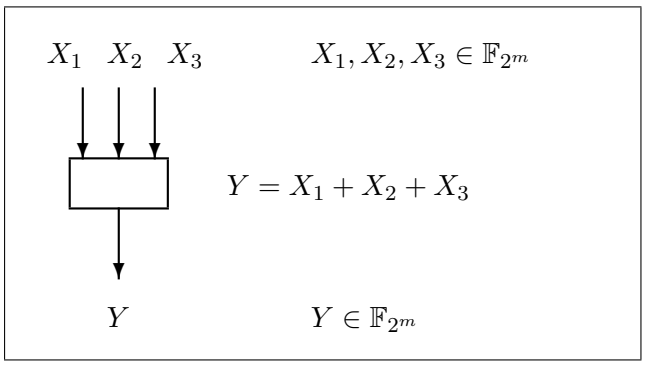


FIG. 8.2 –: *Exemple de boîte de diffusion*

Des multiples tentatives de cryptanalyse du DES ont notamment émergé certains critères de sécurité sur les différentes composantes d'une primitive cryptographiques conventionnelles. Je m'intéresserai par la suite plus particulièrement aux propriétés requises pour construire les boîtes de diffusion.

Critères sur le graphe :

Certaines conditions sur le graphe, sans tenir compte des boîtes de calcul, ont été mises en évidence par Serge Vaudenay [Vau95a] suite à la technique de *cryptanalyse par boîtes noires* [SV95].

Critères sur les boîtes de confusion :

De multiples critères de non-linéarité (strict avalanche criterium, critère de propagation, ...) ont été développés notamment à partir de l'analyse des boîtes-S du DES. Une des conditions les plus fortes est que les composantes des boîtes de confusion opérant sur des données binaires soient des fonctions $f : \mathbb{F}_2^n \to \mathbb{F}_2$ telles que, pour tout vecteur $a \in \mathbb{F}_2^n$ $f(x) = f(x + a)$ pour exactement la moitié des valeurs de $x \in \mathbb{F}_2^n$. Une telle fonction est dite *parfaitement non-linéaire* puisqu'une fonction linéaire vérifie au contraire $f(x) = f(x + a)$ pour toutes les valeurs de x ou pour aucune d'entre elles [MS90]. Les boîtes correspondant sont alors celles qui résistent le mieux à la cryptanalyse différentielle. Ces fonctions correspondent exactement aux fonctions *courbes* introduites par Rothaus [Rot76], et se caractérisent notamment par une propriété sur leur transformée de Fourier. Elles n'existent que dans le cas où le nombre de variables n est pair.

Critères sur les boîtes de diffusion :

L'idée intuitive développée par Maurer et Massey [MM90], selon laquelle la propriété de générateur aléatoire localement parfait est proche de celle de diffusion, a été formalisée par Schnorr et Vaudenay à travers la notion de multipermutations [SV95], généralisée ensuite dans [Vau95b].

Définition 8.3 (multipermutation) [SV95, Vau95b]
Une (r, n)-multipermutation *sur un alphabet fini* $\mathcal{F}$ *est une fonction* π *de* $\mathcal{F}^r$ *dans* $\mathcal{F}^n$ *telle que deux* $(r+n)$*-uplets différents de la forme* $(x, f(x))$ *coïncident sur au plus* $n+1$ *positions différentes.*

Comme le fait remarquer Serge Vaudenay, une multipermutation à une entrée et n sorties est une collection de n permutations de l'alphabet $\mathcal{F}$. Une multipermutation à deux entrées et n sorties est un ensemble de n carrés latins deux-à-deux orthogonaux sur $\mathcal{F}$.

Cette définition implique donc qu'une boîte contenant une telle fonction assure une diffusion parfaite. En effet, la modification de t valeurs d'entrée suscite une modification d'au moins $n-t+1$ valeurs de sorties. Son utilisation au sein du réseau d'une primitive cryptographique permet donc d'éviter bon nombre d'attaques, notamment celles qui consistent à trouver un couple de valeurs d'entrées proches dont les images par la boîte sont également très proches. Ainsi, Serge Vaudenay a développé en 1994 une cryptanalyse partielle de la fonction de hachage MD4 en exploitant le fait que certaines de ses boîtes ne sont pas des multipermutations [Vau95b] ; il a pu alors exhiber aisément des collisions sur les 2 premiers tours de la fonction — qui conduisent à des pseudo-collisions pour la fonction.

2.2 Exemple : FFT-Hashing

Je montre ici sur un exemple l'importance de l'utilisation des multipermutations dans les primitives cryptographiques.

La famille de fonctions de hachage FFT-Hashing est issue d'une idée de Claus Schnorr, présentée à Crypto'91. Baritaud, Gilbert et Girault ont toutefois construit des collisions pour cette fonction [BGG93], ce qui a conduit Schnorr à présenter une seconde version [Sch93], cassée elle par Serge Vaudenay [Vau93]. Cependant, cette construction a été reprise dans un travail commun de Schnorr et Vaudenay [SV95, SV94] auquel je fais ici référence.

Ces fonctions reposent sur une famille de fonctions de compression, notées $g_{k,s}$, agissant sur des mots de $\mathbb{F}_{2^m}$, dont le graphe de calcul est celui de la transformée de Fourier rapide. Les paramètres k et s indiquent que la fonction $g_{k,s}$ possède 2^k entrées et $s+1$ couches. Il s'agit donc d'une fonction de $\mathbb{F}_{2^m}^{2^k}$ dans $\mathbb{F}_{2^m}^{2^{k-1}}$; elle comporte exactement $2^{k-1}(s+1)$ boîtes de calcul — 2^{k-1} boîtes par couche —, notées $B_{i,j}$ avec $0 \leq i < 2^{k-1}$ et $0 \leq j \leq s$. Chacune de ces boîtes possède 2 entrées et 2 sorties à valeurs dans $\mathbb{F}_{2^m}$.

On associe alors à cette fonction de compression une fonction de hachage, $h_{k,s}$ de la manière suivante : le message à hacher, M, est découpé en n blocs de 2^{m+k-1} bits. A partir d'une valeur initiale $H_0 \in \mathbb{F}_{2^m}^{k-1}$, on itère la fonction de compression

$$\forall\ 1 \leq i \leq N,\ \ H_i = g_{k,s}(H_{i-1}, M_i)$$

Le haché du message est alors constitué de la dernière valeur de hachage, H_n.

La figure 8.3 reprend de [SV95] la description de la fonction de compression $g_{k,s}$ à l'itération i.

Entrées :

$H_{i-1} = (h_0, h_1, \ldots, h_{2^{k-1}-1})$ où chaque h_i est dans $\mathbb{F}_{2^m}$
$M_i = (m_0, m_1, \ldots, m_{2^{k-1}-1})$ où chaque m_i est dans $\mathbb{F}_{2^m}$

Algorithme :

- $E^0 = (h_0, m_0, h_1, m_1, \ldots, h_{2^{k-1}-1}, m_{2^{k-1}-1}) = (e_0^0, e_1^0, \ldots, e_{2^k-1}^0)$
- Pour j variant de 0 à s, pour tout i entre 0 et $2^k - 1$ dont la j-ième composante binaire est nulle,
  $$(e_i^{j+1}, e_{i+2^j}^{j+1}) = B_{i,j}(e_i^j, e_{i+2^j}^j)$$
  où + désigne le XOR binaire.
- $E^{s+1} = (e_0^{s+1}, e_1^{s+1}, \ldots, e_{2^k-1}^{s+1})$

Sorties :

$H_i = (e_0^{s+1}, e_2^{s+1}, e_4^{s+1}, \ldots, e_{2^k-2}^{s+1})$

FIG. 8.3 –: *Description de la fonction de compression $g_{k,s}$*

Ainsi les figures 8.4 et 8.5 décrivent respectivement les fonctions $g_{2,1}$ et $g_{3,3}$. Le schéma de $g_{4,4}$ est, lui, explicité dans [Vau95a, page 68].

Les critères classiques de sécurité d'une fonction de hachage imposent que la fonction de compression $g_{k,s}$ vérifie les deux conditions suivantes :

– il doit être difficile d'inverser $g_{k,s}$, c'est-à-dire de trouver une valeur de m telle $h' = g_{k,s}(h, m)$, pour h et h' fixés.

– il doit être difficile de trouver une collision pour $g_{k,s}$, c'est-à-dire de trouver deux valeurs m et m' telles que $g_{k,s}(h, m) = g_{k,s}(h', m')$, pour h et h' fixés.

Je vais montrer sur un exemple simple, même s'il n'est pas réellement pertinent d'un point de vue cryptographique, qu'il est beaucoup plus facile d'inverser la fonction $g_{2,1}$ lorsque les boîtes $B_{i,j}$ ne sont pas des multipermutations. Je suppose ici que toutes les boîtes de calcul du réseau sont identiques, et je me place

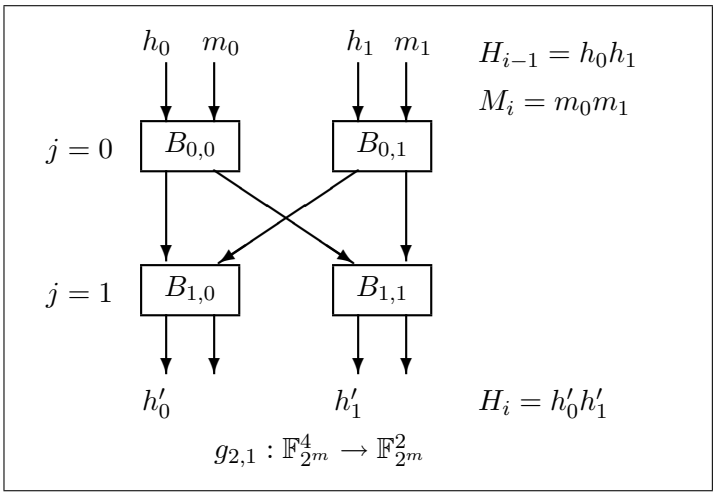


FIG. 8.4 –: *Fonction de compression* $g_{2,1}$

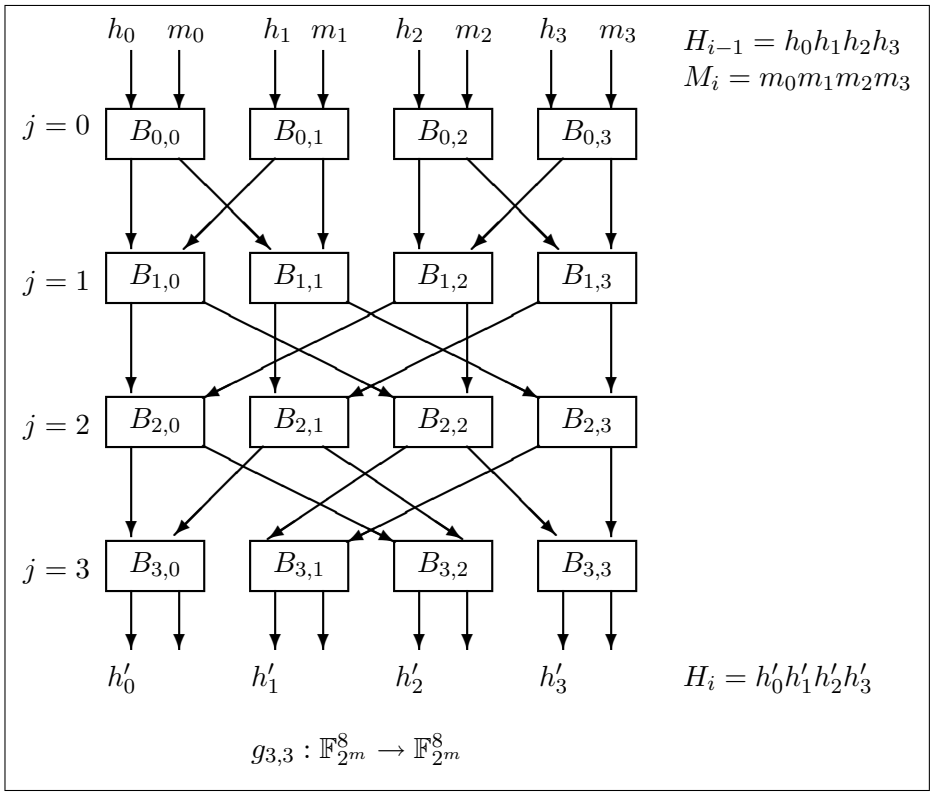


FIG. 8.5 –: *Fonction de compression* $g_{3,3}$

dans le cas où $m = 2$, c'est-à-dire que toutes les boîtes (à 2 entrées et 2 sorties) opèrent sur des éléments de $\mathbb{F}_4$. Afin d'alléger les notations, je désignerai les éléments de $\mathbb{F}_4 = \{0, 1, \alpha, \alpha^2\}$, où α est une racine de $X^2 + X + 1$, par l'écriture décimale de leur représentation polynômiale $\{0, 1, 2, 3\}$. Le problème considéré ici est de trouver un bloc de message $m = (m_0, m_1) \in \mathbb{F}_4^2$ tel que $g_{2,1}(h, m) = h$ pour $h = (3, 2)$.

Cas où les boîtes ne sont pas des multipermutations

Supposons dans un premier temps que les boîtes du réseau sont données par la table de vérité suivante, où a et b sont respectivement les entrées de gauche et de droite de la boîte, et x et y ses sorties de gauche et de droite :

a	b	x	y
0	0	0	1
0	1	1	2
0	2	2	0
0	3	3	3
1	0	1	3
1	1	1	3
1	2	0	2
1	3	3	0
2	0	3	2
2	1	0	0
2	2	1	1
2	3	2	3
3	0	3	3
3	1	1	0
3	2	0	3
3	3	1	3

Les différentes étapes de la résolution apparaissent à la figure 8.6 :

1. L'examen de la table de vérité permet de déduire du fait que l'entrée gauche de $B_{0,0}$ est 3, que sa sortie de droite vaut 0 ou 3.

2. En analysant la boîte $B_{1,1}$, sachant que sa sortie de gauche est égale à 2 et que son entrée de gauche est 0 ou 3, on constate à partir de la table de vérité que l'entrée de gauche est nécessairement 0, et que son entrée de droite vaut 2.

3. Pour la boîte $B_{0,1}$ dont on connaît l'entrée de gauche, égale à 2 et la sortie de droite égale à 2, l'entrée de droite, m_1 et la sortie de gauche sont donc entièrement déterminées et égales respectivement à 0 et 3.

4. Je connais alors l'entrée de gauche (3) et la sortie de droite (0) de la boîte $B_{0,0}$. J'en déduis donc que que son entrée de droite, m_0 vaut 1 et sa sortie de gauche vaut 1.

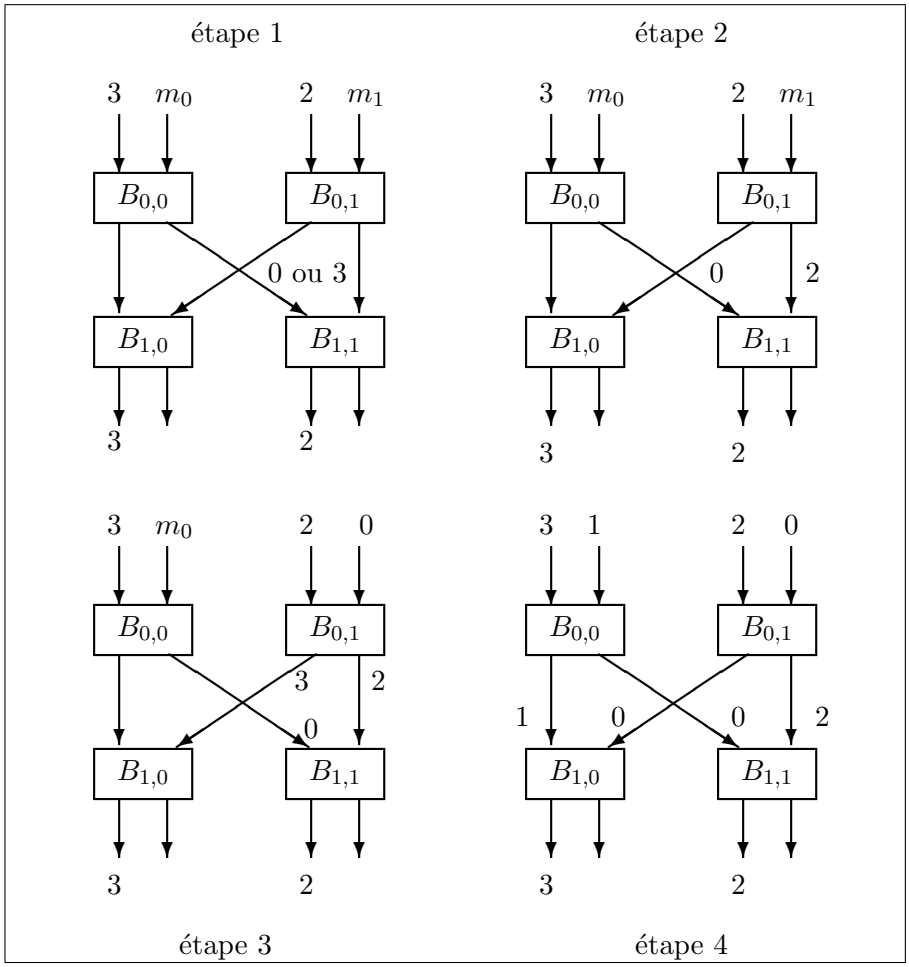


FIG. 8.6 –: *Exemple d'inversion de la fonction $g_{2,1}$ quand les boîtes ne sont pas des multipermutations*

5. On vérifie finalement sur la boîte $B_{1,0}$ que cette solution convient.

Aussi le message $m = (1, 0)$ est-il solution du problème d'inversion considéré. On voit que cette solution a été trouvée sans aucun calcul.

Cas où les boîtes sont des multipermutations

Supposons maintenant que les boîtes $B_{i,j}$ sont données par la table de vérité suivante, qui est celle d'une (2,2)-multipermutation de $\mathbb{F}_4$, puisqu'elle représente

deux carrés latins orthogonaux.

a	b	x	y
0	0	0	0
0	1	1	1
0	2	2	2
0	3	3	3
1	0	1	2
1	1	0	3
1	2	3	0
1	3	2	1
2	0	2	3
2	1	3	2
2	2	0	1
2	3	1	0
3	0	3	1
3	1	2	0
3	2	1	3
3	3	0	2

Essayons alors de résoudre le même problème d'inversion que précédemment. Je vais donc tenter, pour la boîte $B_{0,0}$, de déduire de la connaissance de son entrée de gauche (3), des renseignements sur ses autres entrées/sorties. En examinant la table de vérité, on constate alors que chacune des autres entrées/sorties peut prendre toutes les valeurs possibles de $\mathbb{F}_4$, ce qui ne conduit à aucune information supplémentaires. Le même phénomène se produit pour n'importe laquelle des boîtes du réseau, et ce quelque soit le problème d'inversion considéré. Cela signifie donc que pour trouver une solution à ce problème, il faut au minimum tester toutes les possibilités pour une autre entrée de la fonction, par exemple m_0, ce qui augmente évidemment la complexité d'une telle attaque.

On mesure donc l'importance de l'utilisation des multipermutations dans les boîtes de diffusion des primitives cryptographiques.

2.3 Lien avec les fonctions sans corrélation

La notion de multipermutation peut, elle aussi, s'exprimer en termes de tableaux orthogonaux. On peut alors donner plusieurs définitions équivalentes de cette propriété — l'équivalence entre les énoncés 1, 2 et 3 est mentionnée dans [Vau95a].

Proposition 8.4 (caractérisations des multipermutations) *Soit $\mathcal{F}$ un alphabet à q éléments. Les assertions suivantes sont équivalentes :*

1. *La fonction $\pi : \mathcal{F}^r \to \mathcal{F}^n$ est une (r, n) multipermutation sur $\mathcal{F}$.*
2. *Le tableau dont les lignes sont les vecteurs $(x, \pi(x))_{x\in\mathcal{F}^r}$ est un tableau orthogonal de taille q^r, à $r + n$ contraintes et de force r sur $\mathcal{F}$.*

3. *Le code dont les mots sont les $(r+n)$-uplets $(x, \pi(x))_{x \in \mathcal{F}^r}$ est un code MDS de longueur $r+n$ et de taille q^r.*

4. *La fonction g_π de $\mathcal{F}^r$ dans $\mathcal{F}^{r+n}$, qui à tout x associe $(x, \pi(x))$, est un $(r, r+n)$-générateur aléatoire localement parfait d'ordre r sur $\mathcal{F}$.*

5. *La fonction f_π de $\mathcal{F}^{r+n}$ dans $\mathbb{F}_2$ telle que $|f^{-1}(1)| = q^r$, définie par $f_\pi(x, y) = 1$ ssi $y = \pi(x)$ est sans corrélation d'ordre r sur $\mathcal{F}$.*

preuve : La fonction π est une (r, n)-multipermutation si et seulement si, par définition, deux $(n+r)$-uplets de la forme $(x, \pi(x))$ ne coïncident pas sur r positions. Comme il y a exactement q^r tels vecteurs, cette propriété équivaut à dire que, étant donné un ensemble quelconque R de r positions, tout r-uplet apparaît exactement une fois comme la projection d'un vecteur $(x, \pi(x))$ sur R, ce qui est exprimé par l'assertion 2.

La borne de Singleton impose que la distance minimale d du code dont les mots sont les vecteurs $(x, \pi(x))$ est au plus $n+1$. La fonction π est donc une multipermutation si et seulement si la borne de Singleton est atteinte pour ce code.

Les deux dernières propriétés se déduisent immédiatement des caractérisations des générateurs aléatoires localement parfait et des fonctions sans corrélation en termes de tableaux orthogonaux. □

La caractérisation des multipermutations par les codes MDS impose notamment la distribution des distances des vecteurs $(x, \pi(x))$. Cette propriété, bien connue pour les codes MDS linéaires sur un corps fini, a été généralisée par Ph. Delsarte à tout code défini sur un groupe abélien fini. Aussi, lorsque π est une (r, n)-multipermutation sur un groupe abélien à q éléments, la distribution moyenne des distances, $(A_0, \ldots, A_{n+r})$, entre les vecteurs $(x, \pi(x))$ est-elle donnée par

$$\forall\, 0 \leq i \leq r-1, \quad A_{n+r-i} = \sum_{j=i}^{r-1} (-1)^{j-i} \binom{j}{i} \binom{n+r}{j} (q^{r-i} - 1)$$

2.4 Ordre de non-corrélation binaire d'une multipermutation

A la suite de Schnorr et Vaudenay, il est raisonnable d'imposer que les boîtes de diffusion d'une primitive cryptographique contiennent des multipermutations. Je m'intéresserai ici plus particulièrement au cas où les entrées et sorties de ces boîtes sont des éléments d'un corps de caractéristique 2 puisqu'il s'agit de la situation rencontrée généralement dans la pratique. L'équivalence développée plus haut entre une (r, n)-multipermutation de $\mathbb{F}_{2^m}$ et une fonction f_π à valeurs dans $\mathbb{F}_2$ et sans corrélation d'ordre r sur $\mathbb{F}_{2^m}$ conduit immédiatement à s'interroger sur le comportement d'une telle multipermutation au niveau binaire, ou autrement dit sur l'ordre de non-corrélation de la fonction f_π sur $\mathbb{F}_2$.

En effet, imposer que les boîtes de diffusion d'une primitive soient des multipermutations permet de minorer la complexité de diverses attaques de la fonction : dans l'exemple développé précédemment (figure 8.6), on peut par exemple déduire qu'il est au minimum nécessaire de passer en revue toutes les 2^m valeurs possibles de m_0 pour inverser la fonction $g_{2,1}$. Mais il est naturel de se demander si la complexité d'une telle attaque ne peut pas être réduite lorsque l'on considère les entrées et sorties de la primitive comme des chaînes binaires et non plus comme des éléments de $\mathbb{F}_{2^m}$. Il semble alors possible de déduire de précieuses informations en n'examinant que quelques bits des entrées/sorties des boîtes sans être contraint d'essayer toutes les valeurs possibles de $\mathbb{F}_{2^m}$.

Aussi vais-je maintenant illustrer cette idée par un exemple — une fois encore non significatif d'un point de vue cryptographique mais uniquement destiné à une meilleure appréhension du problème. Je vais en effet montrer qu'il est possible par cette méthode de réduire la complexité de la recherche de collisions pour la fonction de compression $g_{2,1}$.

2.4.1 Exemple de recherche de collisions pour une fonction de compression de FFT-Hashing

Trouver une collision pour cette fonction revient, pour des valeurs H_1 et H_2 fixées dans $\mathbb{F}_{2^m}$, à exhiber un couple de messages (M_1, M_2) et (M'_1, M'_2) solution du réseau de la figure 8.7.

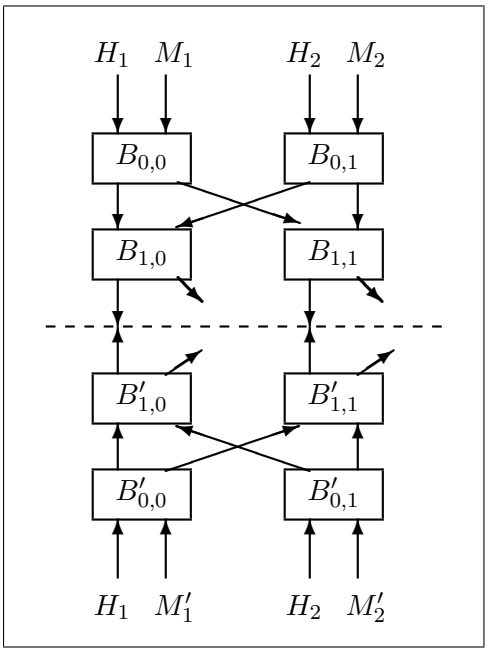


FIG. 8.7 –: *Réseau de collisions de la fonction de compression* $g_{2,1}$

J'utiliserai ici la terminologie développée par Schnorr et Vaudenay [SV95]

relative à ce qu'ils ont appelé la *cryptanalyse par boîtes noires*. Je dirai donc qu'une boîte à r entrées et n sorties est *résolue* dès lors que toutes ses entrées et sorties sont déterminées. Par définition, lorsque la boîte considérée est une (r, n)-multipermutation, sa résolution nécessite la connaissance d'exactement r valeurs de ses entrées/sorties. Ainsi résoudre une telle boîte quand seules $t < r$ valeurs des entrées/sorties sont connues requiert l'examen de toutes les possibilités de $r-t$ autres entrées/sorties, *i.e.* l'examen d'un espace de taille q^{r-t} où q est le cardinal de l'alphabet considéré. La *complexité de résolution* de cette boîte est alors q^{r-t}. De même la *complexité de résolution d'un réseau* comme celui de la figure 8.7, *i.e.* la complexité de la résolution de toutes ses boîtes, correspond à la taille maximale de l'espace examiné au cours de la cryptanalyse.

Les entrées et sorties des boîtes sont assimilées à des éléments de $\mathbb{F}_{2^m}$

Reprenons tout d'abord la recherche de collisions pour $g_{2,1}$ telle qu'elle est envisagée dans [SV95]. Le cheminement de cette cryptanalyse est détaillé par la figure 8.8. Dans cette figure comme dans toutes celles qui vont suivre, les boîtes résolues sont marquées d'une étoile ($*B_{0,0}*$). En outre, les différentes entrées/sorties de la primitive dont toutes les valeurs doivent être examinées sont soulignées afin de rendre visible la complexité de l'attaque à chaque étape de la résolution.

1. Toutes les boîtes de réseau étant des multipermutations, il est nécessaire pour obtenir une quelconque information d'essayer toutes les valeurs possibles pour une des inconnues parmi M_1, M_2, M'_1, M'_2. Si l'on passe en revue les 2^m valeurs que peut prendre par exemple M_1, on peut alors en déduire les deux sorties de la boîte $B_{0,0}$.

2. Pour pouvoir progresser dans la cryptanalyse et résoudre une nouvelle boîte, il faut encore examiner toutes les valeurs possibles pour une autre des inconnues, par exemple M_2. On résout alors successivement les boîtes $B_{0,1}$, $B_{1,0}$ et $B_{1,1}$.

3. De nouveau, seule une des entrées/sorties est connue pour chacune des boîtes restant à résoudre. Il est donc encore nécessaire de passer en revue toutes les valeurs d'une nouvelle inconnue, par exemple M'_1. Ceci permet maintenant de résoudre toutes les boîtes de la partie inférieure du réseau (successivement $B'_{0,0}$, $B'_{1,0}$, $B'_{1,1}$ et finalement $B'_{0,1}$) et par conséquent de trouver des collisions.

En résumé, la propriété de multipermutation requise pour chaque boîte du réseau implique que la complexité d'une telle recherche de collisions est $(2^m)^3$.

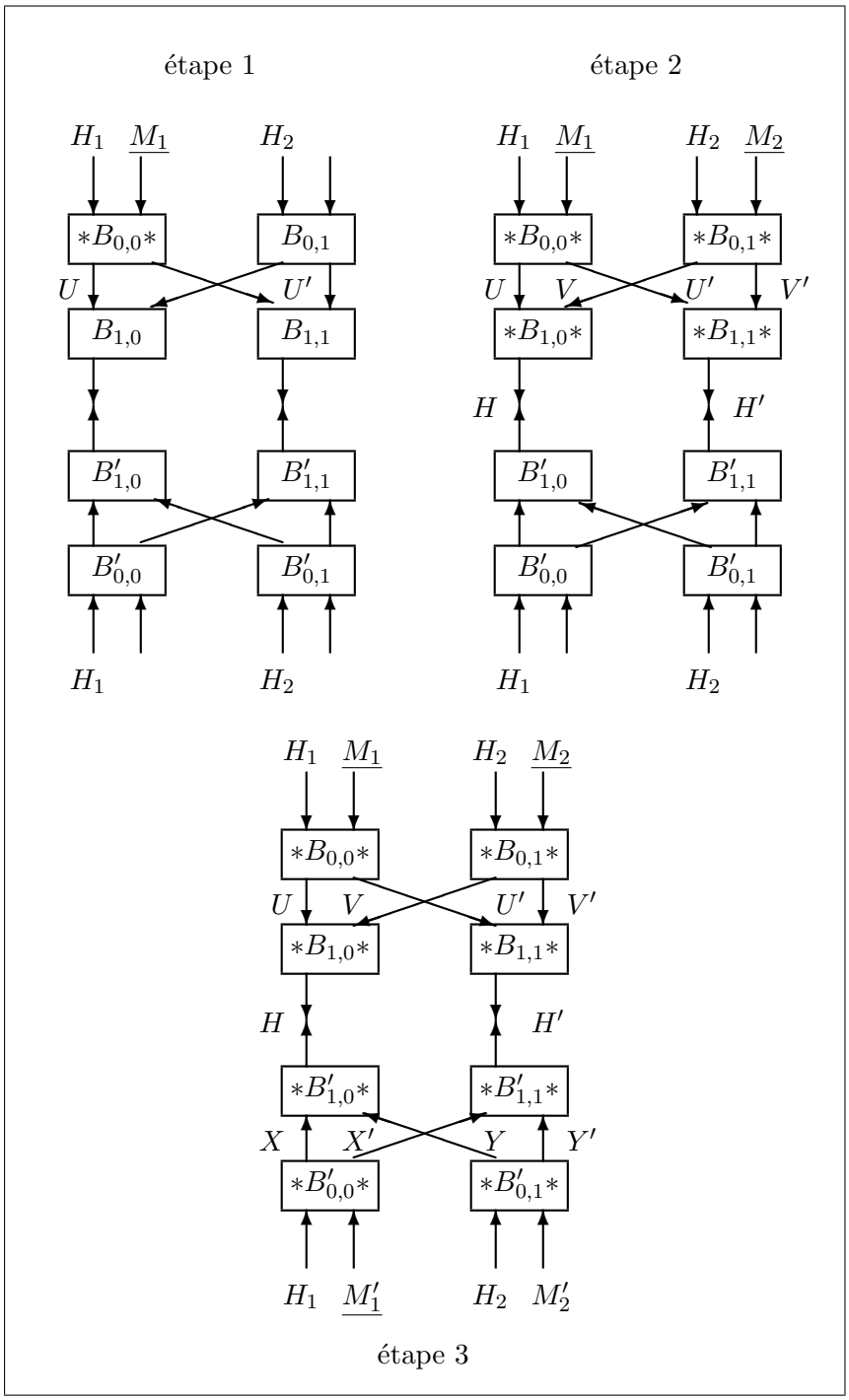


FIG. 8.8 –: *Recherche de collisions pour $g_{2,1}$ quand les entrées et sorties sont assimilées à des éléments de $\mathbf{F}_{2^m}$*

Les entrées et sorties des boîtes sont assimilées à des éléments de $\mathbf{F}_2^m$

Je vais maintenant montrer que l'analyse de ce réseau de collisions au niveau du bit, et non plus au niveau de $\mathbb{F}_{2^m}$, peut dans certains cas être développée en une complexité moindre. Pour cela, je m'intéresse au cas où les différentes boîtes sont définies sur $\mathbb{F}_4$ et spécifiées de la manière suivante. Toutes les boîtes de la partie gauche du réseau — $B_{0,0}$, $B_{1,0}$ et leurs symétriques — sont définies par la fonction

$$\begin{array}{rccc} \pi_1 : & \mathbb{F}_2^4 & \rightarrow & \mathbb{F}_2^4 \\ & (x_1, x_0, y_1, y_0) & \mapsto & (x_0 + y_0, x_1 + y_1, x_1 + x_0 + y_1 + 1, x_1 + y_0) \end{array}$$

et les boîtes de la partie droite — $B_{0,1}$, $B_{1,1}$ et leurs symétriques — contiennent la fonction

$$\begin{array}{rccc} \pi_2 : & \mathbb{F}_2^4 & \rightarrow & \mathbb{F}_2^4 \\ & (x_1, x_0, y_1, y_0) & \mapsto & (x_1 + y_1, x_0 + y_0, x_1 + x_0 + y_1 + 1, x_1 + y_0) \end{array}$$

Ces fonctions sont bien des (2,2)-multipermutations de $\mathbb{F}_4$ puisqu'elles correspondent toutes deux à un couple de carrés latins orthogonaux, comme le montrent les tableaux suivants.

$x\backslash y$	0	1	α	α^2
0	$(0,1)$	$(\alpha,0)$	$(1,\alpha^2)$	(α^2,α)
1	(α,α^2)	$(0,\alpha)$	$(\alpha^2,1)$	$(1,0)$
α	$(1,\alpha)$	(α^2,α^2)	$(0,0)$	$(\alpha,1)$
α^2	$(\alpha^2,0)$	$(1,1)$	(α,α)	$(0,\alpha^2)$

Fonction π_1

$x\backslash y$	0	1	α	α^2
0	$(0,1)$	$(1,0)$	(α,α^2)	(α^2,α)
1	$(1,\alpha^2)$	$(0,\alpha)$	$(\alpha^2,1)$	$(\alpha,0)$
α	(α,α)	(α^2,α^2)	$(0,0)$	$(1,1)$
α^2	$(\alpha^2,0)$	$(\alpha,1)$	$(1,\alpha)$	$(0,\alpha^2)$

Fonction π_2

La figure 8.9 permet de visualiser les étapes successives de la recherche de collisions suivantes :

1. L'examen de toutes les valeurs possibles pour M_1 et pour le bit de poids faible de M_2 me permet de résoudre la boîte $B_{0,0}$ et de trouver le bit de poids faible des deux sorties de $B_{0,1}$. De manière similaire, la connaissance complète de l'entrée de gauche et du bit de poids faible de celle de droite

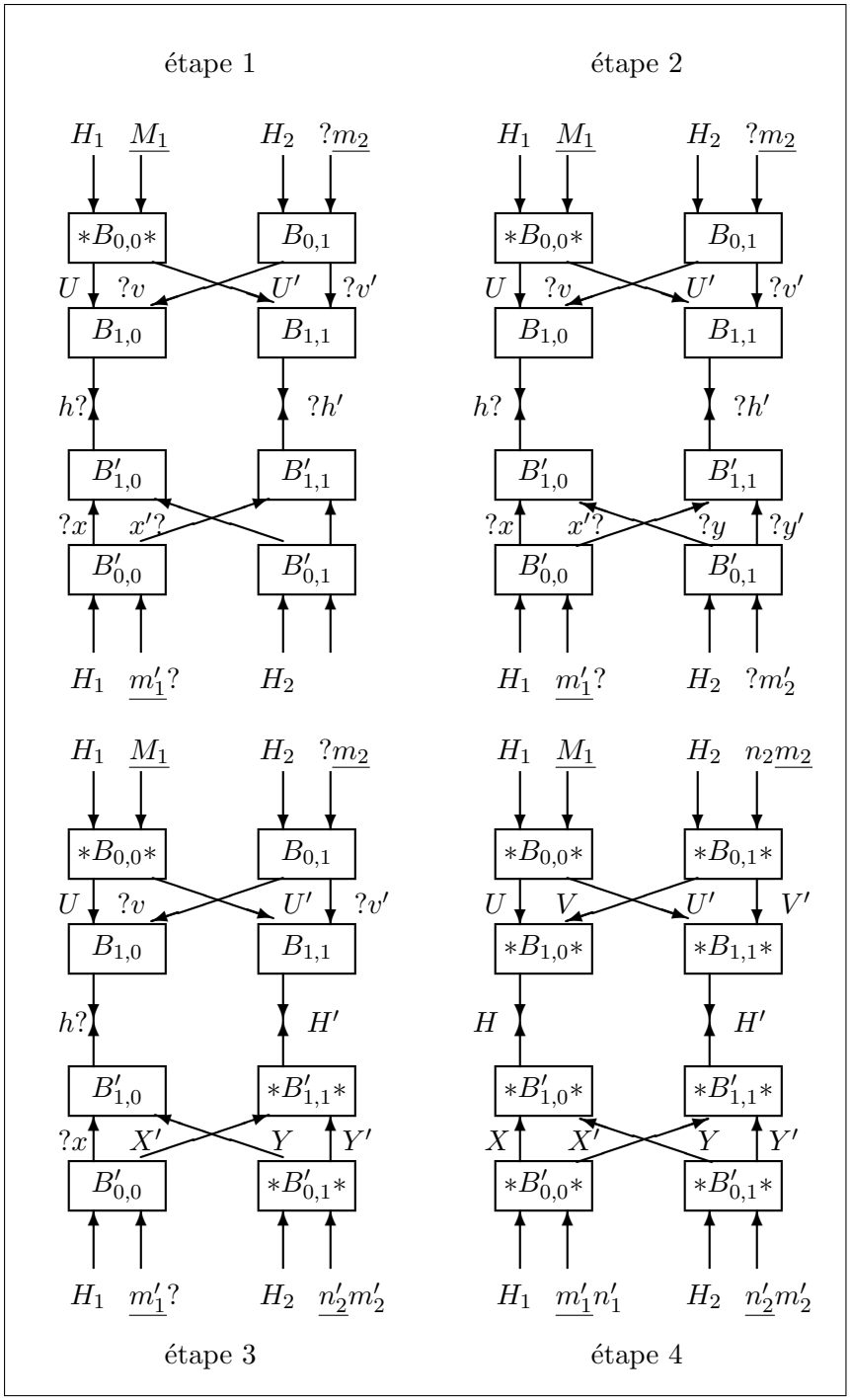


FIG. 8.9 –: *Recherche de collisions pour $g_{2,1}$ quand les entrées et sorties sont assimilées à des éléments de* $\mathbf{F}_2^m$

de la boîte $B_{1,0}$ (resp. $B_{1,1}$) détermine le bit de poids fort (resp. de poids faible) de sa sortie de gauche.

Je considère alors toutes les valeurs possibles pour le bit de poids fort de M'_1 et obtiens ainsi le bit de poids faible de la sortie de gauche et le bit de poids fort de la sortie de droite de la boîte $B'_{0,0}$. A l'issue de cette étape, la complexité de la résolution s'élève à 2^4.

2. Le bit de poids faible de l'entrée de gauche et celui de poids fort de la sortie de gauche de la boîte $B'_{1,0}$ sont désormais connus. La structure particulière de cette boîte permet alors d'en déduire le bit de poids faible de son entrée de droite, ce qui conduit également à la détermination du bit de poids faible de M'_2 et de la sortie de droite de la boîte $B'_{0,1}$.

3. L'examen de toutes les valeurs possibles du bit de poids fort de M'_2 permet alors de résoudre complètement la boîte $B'_{0,1}$. A ce stade, la complexité de la résolution est 2^5. La boîte $B'_{1,1}$ peut ensuite être complètement résolue à partir de la connaissance de son entrée de droite, du bit de poids fort de son entrée de gauche et du bit de poids faible de sa sortie de gauche, en utilisant la structure de la fonction π_2.

4. On peut maintenant résoudre successivement les boîtes $B_{1,1}$, $B_{0,1}$, $B_{1,0}$, $B'_{1,0}$ et $B'_{0,0}$, puisqu'elles possèdent deux entrées/sorties déterminées.

Pour résumer, nous venons de développer une cryptanalyse appréhendant les entrées et sorties du réseau comme des chaînes binaires, dont la complexité s'élève à 2^5. Cette attaque est donc plus efficace que celle exposée par Schnorr et Vaudenay qui, elle, assimilait les entrées et sorties de la primitive à des éléments de $\mathbb{F}_{2^m}$.

Un des points-clef qui assure le succès de la précédente cryptanalyse se situe au tout début de la seconde étape. A ce moment de l'analyse, la connaissance de seulement deux bits parmi toutes les entrées et sorties de la boîte $B'_{1,0}$ (le bit de poids faible de l'entrée de gauche et le bit de poids fort de la sortie de gauche) permet de déterminer le bit de poids de faible de son entrée de droite. On a donc intuitivement la sensation que la diffusion opérée par la multipermutation π_1 n'est pas meilleure d'un point de vue binaire que pour les éléments de $\mathbb{F}_{2^m}$. Cela peut s'exprimer plus formellement par le fait que l'ordre de non-corrélation sur $\mathbb{F}_2$, t, de la fonction f_{π_1} associée à cette multipermutation n'excède pas son ordre de non-corrélation sur $\mathbb{F}_{2^m}$, r.

Afin de se prémunir contre toutes les attaques évoquées par Cl. Schnorr et S. Vaudenay, même lorsqu'elles se placent au niveau du bit, il est donc indispensable que les multipermutations utilisées dans les boîtes de diffusion d'une primitive cryptographique aient un "bon"ordre de non-corrélation sur $\mathbb{F}_2$.

2.4.2 Bornes sur l'ordre de non-corrélation binaire d'une multipermutation

Déterminer l'ordre de non-corrélation binaire d'une multipermutation définie sur $\mathbb{F}_{2^m}$ est cependant loin d'être un problème aisé comme en témoignent les nombreuses tentatives infructueuses pour calculer de manière générale la distance minimale de l'image binaire d'un code de Reed-Solomon défini sur $\mathbb{F}_{2^m}$. Les inégalités que j'ai démontrées au chapitre précédent vont malgré tout me permettre de minorer et majorer cet ordre de non-corrélation binaire en fonction du degré de la forme algébrique normale de des composantes binaires de la multipermutation.

Je commencerai tout d'abord par donner un encadrement du degré de la forme algébrique normale de la fonction sans corrélation f_π associée canoniquement à une multipermutation (*cf.* proposition 8.4).

Proposition 8.5 *A toute (r,n)-multipermutation π sur $\mathbb{F}_{2^m}$ est associée une fonction $f_\pi : \mathbb{F}_2^{m(r+n)} \to \mathbb{F}_2$ sans corrélation d'ordre r sur $\mathbb{F}_2$, et dont la forme algébrique normale est de degré d avec*

$$mn - 1 + \max_{\substack{1 \le i \le n \\ 0 \le j \le m-1}} deg(\pi_{i,j}) \le d \le m(r+n) - r$$

où $\pi = (\pi_1, \ldots, \pi_n)$ est identifiée à une fonction de $\mathbb{F}_2^{mr}$ dans $\mathbb{F}_2^{mn}$ et la fonction booléenne $\pi_{i,j}$ désigne la j-ième composante binaire de π_i.

preuve : La partie droite de l'encadrement provient simplement du théorème 7.37 qui donne une inégalité sur le degré d'une fonction q-aire sans corrélation sur $\mathbb{F}_{q^m}$.

Sa partie gauche se déduit, elle, de l'expression de la fonction f_π. En effet, si l'on considère maintenant la fonction π comme un ensemble de mn fonctions booléennes définies par

$$\begin{array}{rccc} \pi_{i,j} : & \mathbb{F}_2^{mr} & \to & \mathbb{F}_2 \\ & (x_{1,0}, \ldots, x_{r,m-1}) & \mapsto & \pi_{i,j}(x_{1,0}, \ldots, x_{r,m-1}) \end{array}$$

on a par définition $f_\pi(x_{1,0}, \ldots, x_{r+n,m-1}) = 1$ si et seulement si, pour toutes les valeurs de $1 \le i \le n$ et $0 \le j < m$, $x_{r+i,j} = \pi_{i,j}(x_{1,0}, \ldots, x_{r,m-1})$. On en déduit alors la forme algébrique normale de la fonction f_π :

$$f_\pi(x_{1,0}, \ldots, x_{r+n,m-1}) = \prod_{i=1}^{n} \prod_{j=0}^{m-1} [\pi_{i,j}(x_{1,0}, \ldots, x_{r,m-1}) - x_{r+i,j} - 1]$$

Cette factorisation de la forme algébrique normale implique en outre qu'elle contient qu'elle contient tous les monômes de type $\pi_{k,\ell}(x) \prod_{(i,j) \ne (k,\ell)} x_{r+i,j}$. Aussi son degré est-il supérieur ou égal à $mn - 1 + \max_{i,j} deg \pi_{i,j}$. □

Cette minoration du degré de f_π me permet maintenant, à partir de l'inégalité de Siegenthaler, d'encadrer l'ordre de non-corrélation binaire t de la

fonction.

Théorème 8.6 *A toute (r,n)-multipermutation π sur $\mathbb{F}_{2^m}$ est associée une fonction $f_\pi : \mathbb{F}_2^{m(r+n)} \to \mathbb{F}_2$ sans corrélation d'ordre r sur $\mathbb{F}_{2^m}$ et dont l'ordre de non-corrélation sur $\mathbb{F}_2$, t, vérifie*

$$r \leq t \leq nr - \max_{\substack{1 \leq i \leq n \\ 0 \leq j \leq m-1}} deg(\pi_{i,j})$$

preuve : L'ordre de non-corrélation binaire est trivialement supérieur ou égal à r. La seconde partie de l'encadrement provient directement de la proposition précédente et de l'inégalité de Siegenthaler (théorème 7.25) associée à la remarque 7.26. En effet, on a

$$\frac{|f_\pi^{-1}(1)|}{2^t} = 2^{mr-t} \equiv 0 \bmod 2 \quad \text{et} \quad \frac{|f_\pi^{-1}(0)|}{2^t} = 2^{mr-t}(2^{mn}-1) \equiv 0 \bmod 2$$

puisque la borne de Bush (proposition 6.29) assure que $t < mr$ car il n'existe aucun tableau orthogonal binaire de taille 2^{mr}, à $m(r+n)$ contraintes et de force mr quand $mr > 1$. □

Ce théorème montre donc qu'il n'est pas souhaitable de choisir, pour les boîtes de diffusion d'une primitive cryptographique, des multipermutations sur $\mathbb{F}_{2^m}$ dont les composantes binaires, assimilées à des fonctions booléennes, sont de degré élevé.

En outre, la remarque 7.28 permet d'améliorer le théorème précédent puisque l'on peut borner de manière similaire l'ordre de non-corrélation binaire de la fonction f_π à partir de degré maximal des composantes binaires de toutes les multipermutations $p_2 \circ \pi \circ p_1$ avec $p_1 = (\phi_1, \ldots, \phi_r)$ et $p_2 = (\psi_1, \ldots, \psi_n)$, où les ϕ_i et ψ_i sont des permutations de $\mathbb{F}_{2^m}$.

Corollaire 8.7 *A toute (r,n)-multipermutation π sur $\mathbb{F}_{2^m}$ est associée une fonction $f_\pi : \mathbb{F}_2^{m(r+n)} \to \mathbb{F}_2$ sans corrélation d'ordre r sur $\mathbb{F}_{2^m}$ et dont l'ordre de non-corrélation sur $\mathbb{F}_2$, t, vérifie*

$$r \leq t \leq nr - \delta$$

où δ est le plus entier parmi $\{\max_{i,j} deg\ (p_2 \circ \pi \circ p_1)_{i,j}\}$ avec $p_1 = (\phi_1, \ldots, \phi_r)$ et $p_2 = (\psi_1, \ldots, \psi_n)$ quand les ϕ_i et ψ_i parcourent l'ensemble des permutations de $\mathcal{F}_{2^m}$.

> **Exemple 8.8:** L'exemple que j'ai choisi pour illustrer l'importance de ce théorème est tout simplement l'une des multipermutations suggérées par Schnorr et Vaudenay [SV95]. Il s'agit de la fonction
>
> $$\begin{array}{rccc} \pi : & \mathbb{F}_8^2 & \to & \mathbb{F}_8^2 \\ & (x,y) & \mapsto & (x+y, x+y+R(y)+y\wedge\alpha) \end{array}$$

où α un élément primitif de $\mathbb{F}_8$, R désigne la rotation circulaire vers la droite, $+$ et $\wedge$ sont respectivement le XOR et le ET bit-à-bit.

Cette fonction est une (2,2)-multipermutation - une démonstration détaillée de cette propriété figure dans [SV95, théorème 4] -.

Je considère alors la fonction π' définie par

$$\pi'(x, y) = [\pi(x^3, y^3)]^3$$

Puisque l'exponentiation par 3 est une permutation de $\mathbb{F}_8$, π' est également une (2,2)-multipermutation. En outre, la fonction booléenne à 6 variables $\pi'_{1,0}$ correspondant au bit de poids faible de sa première composante est donnée par

$$\begin{array}{rccc} \pi'_{1,0} : & \mathbb{F}_2^6 & \rightarrow & \mathbb{F}_2 \\ & (x, y) & \mapsto & (x^3 + y^3)^3_{|0} \end{array}$$

où $Z_{|0}$ désigne de bit de poids faible de la décomposition de z sur $\mathbb{F}_2^3$. La forme algébrique normale de cette fonction booléenne, calculée à l'exemple 5, est

$f_0(x_0, x_1, x_2, y_0, y_1, y_2) = x_0 + y_0 + x_1y_2 + x_2y_1 + x_0x_1y_1 + x_1y_0y_1 + x_2y_0y_1 + x_0x_1y_2 + x_2y_0y_2 + x_0x_2y_2 + x_0x_2y_0y_1 + x_0x_1y_0y_2$.

Elle est donc de degré 4. Le corollaire précédent me permet en conséquence d'encadrer l'ordre de non-corrélation binaire t de la fonction f_π associée à la multipermutation π par $2 \leq t \leq 6 - 4$. Aussi la multipermutation π assure-t-elle d'un point de vue binaire, la diffusion la plus mauvaise possible.

Je peux alors conclure de cette étude que l'utilisation de la multipermutation suggérée par [SV95] dans les boîtes de diffusion d'une primitive cryptographique est à proscrire.

Chapitre 9

Construction de nouvelles fonctions résilientes par composition

Il apparaît à la lumière des deux chapitres précédents que les fonctions sans corrélation et les fonctions résilientes sont des objets primordiaux en cryptographie, tant pour les techniques de chiffrement à flot que pour les primitives conventionnelles. Un des problèmes essentiels reste néanmoins de construire de telles fonctions, surtout si l'on désire un nombre de variables et un ordre de non-corrélation élevés. La construction de fonctions résilientes à partir de codes correcteurs d'erreurs, évoquée au chapitre 6, est certes suffisamment générale mais elle conduit principalement à des fonctions résilientes linéaires dans la mesure où les familles de codes non-linéaires étudiées à ce jour dans la littérature sont relativement peu nombreuses. Or, l'utilisation de fonctions linéaires dans les boîtes de diffusion des primitives cryptographiques est naturellement à éviter — sauf si celles-ci sont suivies de boîtes de confusion parfaite. Les cryptographes, en particulier ceux qui conçoivent des primitives, sont donc souvent en quête de fonctions sans corrélation dont la structure soit un peu originale, c'est-à-dire différente de la table d'un groupe ou d'une fonction linéaire. Il s'avère en outre indispensable de disposer d'un grand nombre de ces fonctions dès lors que l'on souhaite les utiliser comme clefs d'un système.

En m'appuyant sur la constatation qu'il est *a contrario* aisé de produire des fonctions sans corrélation d'ordre faible ou ayant un nombre de variables restreint, je propose ici une nouvelle méthode de construction permettant d'obtenir des fonctions ayant à la fois un grand nombre de variables et un ordre de non-corrélation élevé, en composant des fonctions sans corrélation plus simples. Après avoir donné plusieurs exemples illustrant cette construction et avoir explicité sa structure dans le cas linéaire, je montrerai qu'elle est particulièrement adéquate pour fournir des fonctions de combinaison de registres à décalage à rétroaction linéaire.

1 Construction par composition

Dans tout ce chapitre, l'alphabet $\mathcal{F}$ à q éléments est muni d'une structure de groupe abélien.

Définition 9.1 (composition de fonctions) *Soit $(g_i)_{1\leq i\leq k}$ une famille de k fonctions*

$$g_i : \mathcal{F}^n \to \mathcal{F}^d \text{ avec } d \leq n$$

l'ensemble d'arrivée $\mathcal{F}^d$ de chacune de ces fonctions étant noté $\mathcal{A}$. On lui associe la fonction g de la manière suivante

$$\begin{array}{rccc} g : & \mathcal{F}^{nk} & \to & \mathcal{A}^k \\ & (x_1,\ldots,x_k) & \mapsto & (g_1(x_1),\ldots,g_k(x_k)) \end{array}$$

Soit maintenant une fonction h

$$h : \mathcal{A}^k \to \mathcal{F}^\ell \text{ avec } \ell \leq kd$$

La fonction composée $f = h \circ g$ *est alors définie par*

$$\begin{array}{rccc} f : & \mathcal{F}^{nk} & \to & \mathcal{F}^\ell \\ & (x_1,\ldots,x_k) & \mapsto & h(g_1(x_1),\ldots,g_k(x_k)) \end{array}$$

Les propriétés de non-corrélation des fonctions g_i et h se répercutent alors sur la fonction composée comme le traduisent les deux propositions suivantes.

Proposition 9.2 *Si chaque fonction g_i est équilibrée et si h est sans corrélation d'ordre r sur $\mathcal{A}$, alors $h \circ g$ est sans corrélation d'ordre r sur $\mathcal{F}^n$.*

preuve : Soit v dans $\mathcal{F}^\ell$, $R = \{i_1,\ldots,i_r\}$ un ensemble de r indices parmi $\{1,\ldots,k\}$, et $\bar{R} = \{j_1,\ldots,j_{k-r}\}$ son complémentaire. La fonction h étant sans corrélation d'ordre r sur $\mathcal{A}$, $h^{-1}(v)$ est un tableau orthogonal à k contraintes, de force r et d'indice λ_v sur $\mathcal{A}$. Etant donné un vecteur $(a_{i_1},\ldots,a_{i_r})$ de $\mathcal{A}^r$, il y a donc exactement λ_v éléments $z = (z_1,\ldots,z_k) \in \mathcal{A}^k$ de $h^{-1}(v)$ dont la restriction sur R est égale à a.

Notons g_R la fonction qui, à $(x_{i_1},\ldots,x_{i_r})$, associe $(g_{i_1}(x_{i_1}),\ldots,g_{i_r}(x_{i_r}))$ et $g_{\bar{R}}$ celle qui, à $(x_{j_1},\ldots,x_{j_{k-r}})$, associe $(g_{j_1}(x_{j_1}),\ldots,g_{j_{k-r}}(x_{j_{k-r}}))$. Comme chaque g_i est par hypothèse équilibrée, $|g_i^{-1}(a_i)| = q^{n-d}$. On en déduit que, pour tout $b = (b_{j_1},\ldots,b_{j_{k-r}})$, $g_{\bar{R}}^{-1}(b)$ est un sous-ensemble de $\mathcal{F}^{n(k-r)}$ de taille $q^{(n-d)(k-r)}$. De même $|g_R^{-1}(a)| = q^{(n-d)r}$ et les $g_R^{-1}(a)$ partitionnent $(\mathcal{F}^n)^r$ lorsque a parcourt $\mathcal{A}^r$.

Aussi chaque r-uplet de $(\mathcal{F}^n)^r$ apparaît-il comme la projection sur R d'exactement $\lambda_v q^{(n-d)(k-r)}$ éléments de $(h\circ g)^{-1}(v)$. Cela signifie donc que $(h\circ g)^{-1}(v)$ est un tableau orthogonal à k contraintes, de force r et d'indice $\lambda_v q^{(n-d)(k-r)}$ sur $\mathcal{F}^n$. □

Proposition 9.3 *Si la fonction composée $f = h\circ g$ est sans corrélation d'ordre r sur $\mathcal{F}^n$ et si chaque fonction g_i est sans corrélation d'ordre t sur $\mathcal{F}$, alors f est sans corrélation d'ordre t' sur $\mathcal{F}$ avec $t' = (t+1)(r+1) - 1$.*

preuve : Soit u un élément de $(\mathcal{F}^n)^k$. On note $W_H(u)$ son poids de Hamming dans $(\mathcal{F}^n)^k$ — le nombre de ses composantes non nulles dans $\mathcal{F}^n$ —, et $w_H(u)$ son poids de Hamming dans $\mathcal{F}^{nk}$ — le nombre de ses composantes non nulles dans $\mathcal{F}$.

La fonction f est sans corrélation d'ordre r sur $\mathcal{F}^n$ si et seulement si, pour tout $v \in \mathcal{F}^\ell$, $f^{-1}(v)$ est un tableau orthogonal de force r sur $\mathcal{F}^n$. Par le théorème 6.10, on a donc

$$\forall u \in (\mathcal{F}^n)^k,\ 1 \leq W_H(u) \leq r, \sum_{x \in f^{-1}(v), x \in (\mathcal{F}^n)^k} < x, u >= 0$$

Si l'on choisit maintenant u de sorte que $W_H(u) > r$ et $w_H(u) < (t+1)(r+1)$, alors il existe au moins un indice $i \in \{1, \ldots, k\}$ tel que $1 \leq w_H(u_i) \leq t$. Grâce au lemme 6.6, on obtient

$$\begin{aligned} \sum_{x \in f^{-1}(v), x \in (\mathcal{F}^n)^k} < x, u > &= \sum_{y \in h^{-1}(v)} \sum_{x \in g^{-1}(y)} < x, u > \\ &= \sum_{y \in h^{-1}(v)} \prod_{i=1}^{k} \sum_{x_i \in g_i^{-1}(y_i)} < x_i, u_i > \end{aligned}$$

Comme chaque g_i est sans corrélation d'ordre t sur $\mathcal{F}$, au moins un des facteurs $\sum_{x_i \in g_i^{-1}(y_i)} < x_i, u_i >$ s'annule. On en déduit par conséquent que

$$\forall u \in \mathcal{F}^{nk},\ 1 \leq w_H(u) \leq t', \sum_{x \in f^{-1}(v), x \in \mathcal{F}^{nk}} < x, u >= 0$$

□

Le théorème suivant résulte donc immédiatement de ces deux propositions.

Théorème 9.4 (ordre de non-corrélation d'une fonction composée) *Si chaque fonction g_i est t-résiliente sur $\mathcal{F}$ et si h est sans corrélation d'ordre r (resp. r-résiliente) sur $\mathcal{A}$, alors la fonction composée $h \circ g$ est sans corrélation d'ordre t' (resp. t'-résiliente) sur $\mathcal{F}$, avec $t' = (t+1)(r+1) - 1$.*

Ainsi, à partir des tableaux orthogonaux (q^{n-d}, n, q, t) constitués des vecteurs de $g_i^{-1}(x)$ et des tableaux orthogonaux $(q^{kd-\ell}, k, q^d, r)$ formés par les éléments de $h^{-1}(y)$, nous avons construit q^ℓ tableaux orthogonaux $(q^{kn-l}, kn, q, (t+1)(r+1)-1)$ d'indice $\lambda_f = q^{kn-\ell-tr-t-r}$.

2 Exemples et corollaires

L'exemple qui suit illustre ce théorème pour des petites valeurs des paramètres ; il permet de visualiser ce résultat d'un point de vue combinatoire. Je

construis en effet ici des tableaux orthogonaux binaires de taille 32, à 6 contraintes et de force 3.

Exemple 9.5: Je considère deux fonctions g_1 et g_2 identiques définies par :

$$g_1 = g_2 : \begin{array}{ccc} \mathbb{F}_2^3 & \to & \mathbb{F}_2^2 \\ (x_1; x_2; x_3) & \mapsto & (x_1 + x_2; x_1 + x_3) \end{array}$$

Cette fonction est 1-résiliente sur $\mathbb{F}_2$.
Je compose maintenant ces deux fonctions avec la fonction $h : (\mathbb{F}_2^2)^2 \to \mathbb{F}_2$ donnée par la transposée de sa table de vérité T_h :

$${}^tT_h = \left[\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \begin{array}{c} x_1 \\ x_2 \\ \\ x_3 \\ x_4 \end{array}$$

Sa table de vérité étant un tableau orthogonal de force 1 (et d'indice 2) sur $\mathbb{F}_2^2$, la fonction h est une fonction 1-résiliente sur $\mathbb{F}_2^2$. En tant que fonction booléenne, il s'agit d'une fonction non-linéaire puisque sa forme algébrique normale est $h(x_1, x_2, x_3, x_4) = x_1 + x_4 + x_2x_3 + x_2x_4$.

La fonction composée f est donc une fonction booléenne à 6 variables qui est, d'après le théorème, 3-résiliente sur $\mathbb{F}_2$. On peut ainsi vérifier que sa table de vérité T_f est un tableau orthogonal binaire de taille 32, à 6 contraintes, de force 3 et d'indice 4 :

$${}^tT_f = \left[\begin{array}{cccccccccccccccccccccccccccccccc} 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1 \\ 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0 \\ 0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1&0&1&0&0&1&1&0&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \\ 0&1&0&1&0&1&0&1&0&1&1&0&0&1&1&0&0&1&0&1&0&1&0&1&1&0&0&1&1&0 \\ 1&0&0&1&1&0&0&1&0&1&0&1&0&1&0&1&0&1&1&0&0&1&1&0&0&1&0&1&0&1&0&1 \end{array}\right] \begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array}$$

La fonction f s'écrit sous forme algébrique normale
$f(x_1, x_2, x_3, x_4, x_5, x_6) = x_1 + x_2 + x_4 + x_6 + x_1x_5 + x_1x_6 + x_3x_5 + x_3x_6$.
Son degré est donc optimal au sens de l'inégalité de Siegenthaler.

Voici maintenant un exemple d'utilisation de cette construction qui fournit une fonction résiliente dont les paramètres sont suffisamment élevés pour qu'elle puisse être utilisée dans des applications cryptographiques. Je construis en effet une fonction booléenne 5-résiliente à 12 variables en composant des fonctions binaires linéaires — *i.e.* des fonctions syndromes — à l'aide d'une fonction non-linéaire sur $\mathbb{F}_2^3$. Il est donc très facile d'obtenir une fonction non-linéaire 5-résiliente pour combiner 12 LFSRs binaires à l'aide de cette méthode alors qu'une construction directe paraît problématique pour ces paramètres.

Exemple 9.6: Soient g_1 et g_2 deux fonctions linéaires identiques :

$$g_1 = g_2 : \begin{array}{ccc} \mathbb{F}_2^6 & \to & \mathbb{F}_2^3 \\ x & \mapsto & x^tH \end{array}$$

où H est une matrice de parité du code linéaire [6,3], $\mathcal{C}$:

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

Le code dual $\mathcal{C}^{\perp}$ a une distance minimale égale à 3, ce qui implique que la fonction g_1 est 2-résiliente sur $\mathbb{F}_2$.

Soit maintenant α une racine du polynôme $X^3 + X + 1$. J'identifie tout élément du corps $\mathbb{F}_8$ à un triplet de bits suivant la décomposition $\mathbb{F}_8 = \alpha^2\mathbb{F}_2 + \alpha\mathbb{F}_2 + \mathbb{F}_2$ et je définis la fonction h par :

$$\begin{array}{rccc} h\colon & \mathbb{F}_8 \times \mathbb{F}_8 & \to & \mathbb{F}_2 \\ & (x, y) & \mapsto & (x^3 + y^3)^3{}_{|0} \end{array}$$

où $z_{|0}$ désigne le bit de poids faible de la décomposition de z sur $\mathbb{F}_2^3$. Par construction, la fonction h est une fonction 1-résiliente sur $\mathbb{F}_8$. La fonction composée f est donc une fonction booléenne à 12 variables, 5-résiliente et de degré 4. Sa forme algébrique normale est donnée par

$f(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}) =$

$x_0x_1x_6x_8 + x_0x_1x_6x_{10} + x_0x_1x_6x_{11} + x_0x_1x_8x_9 + x_0x_1x_8x_{10} + x_0x_1x_9x_{10} +$
$x_0x_1x_9x_{11} + x_0x_1x_{10}x_{11} + x_0x_2x_6x_7 + x_0x_2x_6x_9 + x_0x_2x_6x_{11} + x_0x_2x_7x_9 +$
$x_0x_2x_7x_{10} + x_0x_2x_9x_{10} + x_0x_2x_9x_{11} + x_0x_2x_{10}x_{11} + x_0x_3x_6x_8 + x_0x_3x_6x_{10} +$
$x_0x_3x_6x_{11} + x_0x_3x_8x_9 + x_0x_3x_8x_{10} + x_0x_3x_9x_{10} + x_0x_3x_9x_{11} + x_0x_3x_{10}x_{11} +$
$x_0x_4x_6x_7 + x_0x_4x_6x_9 + x_0x_4x_6x_{11} + x_0x_4x_7x_9 + x_0x_4x_7x_{10} + x_0x_4x_9x_{10} +$
$x_0x_4x_9x_{11} + x_0x_4x_{10}x_{11} + x_0x_5x_6x_7 + x_0x_5x_6x_8 + x_0x_5x_6x_9 + x_0x_5x_6x_{10} +$
$x_0x_5x_7x_9 + x_0x_5x_7x_{10} + x_0x_5x_8x_9 + x_0x_5x_8x_{10} + x_1x_3x_6x_8 + x_1x_3x_6x_{10} +$
$x_1x_3x_6x_{11} + x_1x_3x_8x_9 + x_1x_3x_8x_{10} + x_1x_3x_9x_{10} + x_1x_3x_9x_{11} + x_1x_3x_{10}x_{11} +$
$x_1x_4x_6x_8 + x_1x_4x_6x_{10} + x_1x_4x_6x_{11} + x_1x_4x_8x_9 + x_1x_4x_8x_{10} + x_1x_4x_9x_{10} +$
$x_1x_4x_9x_{11} + x_1x_4x_{10}x_{11} + x_2x_3x_6x_7 + x_2x_3x_6x_9 + x_2x_3x_6x_{11} + x_2x_3x_7x_9 +$
$x_2x_3x_7x_{10} + x_2x_3x_9x_{10} + x_2x_3x_9x_{11} + x_2x_3x_{10}x_{11} + x_2x_4x_6x_7 + x_2x_4x_6x_9 +$
$x_2x_4x_6x_{11} + x_2x_4x_7x_9 + x_2x_4x_7x_{10} + x_2x_4x_9x_{10} + x_2x_4x_9x_{11} + x_2x_4x_{10}x_{11} +$
$x_3x_4x_6x_7 + x_3x_4x_6x_8 + x_3x_4x_6x_9 + x_3x_4x_6x_{10} + x_3x_4x_7x_9 + x_3x_4x_7x_{10} +$
$x_3x_4x_8x_9 + x_3x_4x_8x_{10} + x_3x_5x_6x_7 + x_3x_5x_6x_8 + x_3x_5x_6x_9 + x_3x_5x_6x_{10} +$
$x_3x_5x_7x_9 + x_3x_5x_7x_{10} + x_3x_5x_8x_9 + x_3x_5x_8x_{10} + x_4x_5x_6x_7 + x_4x_5x_6x_8 +$
$x_4x_5x_6x_9 + x_4x_5x_6x_{10} + x_4x_5x_7x_9 + x_4x_5x_7x_{10} + x_4x_5x_8x_9 + x_4x_5x_8x_{10} +$
$x_0x_1x_7 + x_0x_1x_8 + x_0x_1x_9 + x_0x_2x_8 + x_0x_2x_9 + x_0x_2x_{10} + x_0x_2x_{11} + x_0x_3x_7 +$
$x_0x_3x_8 + x_0x_3x_9 + x_0x_4x_8 + x_0x_4x_9 + x_0x_4x_{10} + x_0x_4x_{11} + x_0x_5x_7 + x_0x_5x_{10} +$
$x_0x_5x_{11} + x_1x_3x_7 + x_1x_3x_8 + x_1x_3x_9 + x_1x_4x_7 + x_1x_4x_8 + x_1x_4x_9 + x_1x_6x_7 +$
$x_1x_6x_9 + x_1x_6x_{11} + x_1x_7x_9 + x_1x_7x_{10} + x_1x_9x_{10} + x_1x_9x_{11} + x_1x_{10}x_{11} +$
$x_2x_3x_8 + x_2x_3x_9 + x_2x_3x_{10} + x_2x_3x_{11} + x_2x_4x_8 + x_2x_4x_9 + x_2x_4x_{10} +$
$x_2x_4x_{11} + x_2x_6x_7 + x_2x_6x_8 + x_2x_6x_9 + x_2x_6x_{10} + x_2x_7x_9 + x_2x_7x_{10} +$
$x_2x_8x_9 + x_2x_8x_{10} + x_3x_4x_7 + x_3x_4x_{10} + x_3x_4x_{11} + x_3x_5x_7 + x_3x_5x_{10} +$
$x_3x_5x_{11} + x_3x_6x_7 + x_3x_6x_8 + x_3x_6x_9 + x_3x_6x_{10} + x_3x_7x_9 + x_3x_7x_{10} +$
$x_3x_8x_9 + x_3x_8x_{10} + x_4x_5x_7 + x_4x_5x_{10} + x_4x_5x_{11} + x_4x_6x_8 + x_4x_6x_{10} +$

$$x_4x_6x_{11} + x_4x_8x_9 + x_4x_8x_{10} + x_4x_9x_{10} + x_4x_9x_{11} + x_4x_{10}x_{11} + x_5x_6x_8 + x_5x_6x_{10} + x_5x_6x_{11} + x_5x_8x_9 + x_5x_8x_{10} + x_5x_9x_{10} + x_5x_9x_{11} + x_5x_{10}x_{11} + x_1x_8 + x_1x_9 + x_1x_{10} + x_1x_{11} + x_2x_7 + x_2x_{10} + x_2x_{11} + x_3x_7 + x_3x_{10} + x_3x_{11} + x_4x_7 + x_4x_8 + x_4x_9 + x_5x_7 + x_5x_8 + x_5x_9 + x_0 + x_3 + x_4 + x_6 + x_9 + x_{10}$$

A Eurocrypt'95, Zhang et Zheng présentèrent un article dont les principaux résultats concernaient l'ordre de résilience d'une fonction binaire obtenue soit par addition de fonctions t-résilientes [ZZ95, section 3], soit en permutant les sorties d'une fonction t-résiliente [ZZ95, section 4]. Ces deux théorèmes peuvent être considérés comme des corollaires immédiats du résultat que je viens de démontrer. Ils sont également immédiatement généralisables aux fonctions définies sur un alphabet fini quelconque $\mathcal{F}$.

Corollaire 9.7 (addition de fonctions résilientes) *Soit $(g_i)_{1\leq i\leq k}$ une famille de fonctions de $\mathcal{F}^n$ dans $\mathcal{F}^d$, t-résilientes sur $\mathcal{F}$, et $h : (\mathcal{F}^d)^k \to \mathcal{F}^d$ l'addition sur $\mathcal{F}^d$. Alors la fonction composée*

$$\begin{array}{rccc} f: & \mathcal{F}^{nk} & \to & \mathcal{F}^d \\ & (x_1,\ldots,x_k) & \mapsto & g_1(x_1)+\ldots+g_k(x_k) \end{array}$$

est t'-résiliente sur $\mathcal{F}$ avec $t' = (t+1)k - 1$.

Corollaire 9.8 (permutation des sorties d'une fonction résiliente) *Soit $g : \mathcal{F}^n \to \mathcal{F}^d$ une fonction t-résiliente sur $\mathcal{F}$ et h une permutation de $\mathcal{F}^d$. Alors $h \circ g$ est encore une fonction t-résiliente sur $\mathcal{F}$.*

Enfin, un autre intérêt de cette construction est qu'elle permet de construire des tableaux orthogonaux ayant à la fois un grand nombre de variables et une force élevée — proche des bornes théoriques définies au chapitre 6.

Exemple 9.9: Je considère deux fonctions identiques g_1 et g_2 obtenues à partir des translatés du code de Preparata $\mathcal{P}(5)$ (cf. théorème 6.26). Le code $\mathcal{P}(5)$ étant un code binaire non-linéaire systématique de longueur 64, de taille 2^{52} et de distance duale 28, la fonction

$$g_1 = g_2 : \quad \mathbb{F}_2^{64} \quad \to \quad \mathbb{F}_2^{12}$$

est 27-résiliente sur $\mathbb{F}_2$. Soient maintenant π_1, π_2 et π_3 des permutations de l'alphabet $\mathbb{F}_2^{12}$. La fonction

$$\begin{array}{rccc} h: & (\mathbb{F}_2^{12})^2 & \to & \mathbb{F}_2^{12} \\ & (x_1,x_2) & \mapsto & \pi_3(\pi_1(x_1)+\pi_2(x_2)) \end{array}$$

est une fonction 1-résiliente sur $\mathbb{F}_2^{12}$.

La fonction composée

$$f : \mathbb{F}_2^{128} \to \mathbb{F}_2^{12}$$

est donc une fonction 55-résiliente sur $\mathbb{F}_2$. Les tableaux de la forme $f^{-1}(v)$ pour $v \in \mathbb{F}_2^{12}$ sont par conséquent des tableaux orthogonaux de taille 2^{116}, à 64 contraintes et de force 55. Ils atteignent alors la force maximale que l'on peut obtenir pour un tableau orthogonal construit à partir d'un code linéaire connu [BV93].

3 Composition de fonctions linéaires et dual d'un code concaténé

Je m'intéresse ici plus particulièrement aux fonctions résultant de la composition de fonctions g_i linéaires par une fonction h linéaire — la linéarité est ici définie au sens usuel (cf. théorème 6.25). Une telle fonction $f = h \circ g$ est trivialement linéaire ; elle s'identifie par conséquent à une fonction syndrome. Il est donc légitime de s'interroger sur la structure du code qui lui est associé.

Définition 9.10 (Code concaténé) [For66] *Soit $\mathcal{B}$ un code linéaire $[n_b, k_b]$ sur $\mathbb{F}_q$, $\mathcal{E}$ un code linéaire $[n_e, k_e]$ sur $\mathbb{F}_{q^{k_b}}$ et θ un isomorphisme d'espaces vectoriels*

$$\begin{array}{rccc} \theta : & \mathbb{F}_{q^{k_b}} & \to & \mathcal{B} \\ & x & \mapsto & \theta(x) \end{array}$$

Le code concaténé admettant $\mathcal{B}$ comme code interne et $\mathcal{E}$ comme code externe est le code $\mathcal{B}\square_\theta\mathcal{E}$ constitué des mots de $\mathcal{E}$ dans lesquels les symboles de $\mathbb{F}_{q^{k_b}}$ sont remplacés par des mots de $\mathcal{B}$ à l'aide de l'isomorphisme θ

$$\mathcal{B}\square_\theta\mathcal{E} = \{(\theta(x_1), \ldots, \theta(x_{n_e})),\ \text{où } (x_1, \ldots, x_{n_e}) \in \mathcal{E}\}$$

Je reprends ici les notations utilisées par Nicolas Sendrier [Sen91]. On peut définir de la même façon des codes concaténés obtenus à partir de codes internes $(\mathcal{B}_i)_{1 \le i \le n_e}$ différents ; les codes de Justesen [MS77, page 307], par exemple, sont un cas particulier de cette construction — ils ont pour code externe un code de Reed-Solomon.

Proposition 9.11 (Paramètres d'un code concaténé) *Soient $(\mathcal{B}_i)_{1 \le i \le n_e}$ une famille de codes linéaires $[n_b, k_b, d_b]$ sur $\mathbb{F}_q$, $\mathcal{E}$ un code linéaire $[n_e, k_e, d_e]$ sur $\mathbb{F}_{q^{k_b}}$ et $(\theta_i)_{1 \le i \le n_e}$ une famille d'isomorphismes d'espaces vectoriels*

$$\theta_i : \mathbb{F}_{q^{k_b}} \to \mathcal{B}_i$$

On définit l'isomorphisme $\mathbb{F}_q$-linéaire Θ :

$$\begin{array}{rccc} \Theta : & \mathbb{F}_{q^{k_b}}^{n_e} & \to & \mathcal{B}_1 \times \ldots \times \mathcal{B}_{n_e} \\ & x = (x_1, \ldots, x_n) & \mapsto & (\theta_1(x_1), \ldots, \theta_{n_e}(x_{n_e})) \end{array}$$

Alors le code concaténé $(\mathcal{B}_i)\square_\Theta\mathcal{E}$ est un code linéaire de longueur $n_b n_e$, de dimension $k_b k_e$ et de distance minimale $d_b d_e$.

Le code associé à une fonction résiliente obtenue par composition de fonctions linéaires peut alors s'exprimer en termes de code concaténé :

Théorème 9.12 (Composition de fonctions résilientes linéaires) *Soit une famille de k fonctions linéaires t-résilientes $(g_i)_{1\leq i\leq k}$*

$$\begin{array}{rccc} g_i: & \mathbb{F}_q^n & \to & \mathbb{F}_q^d \\ & x_i & \mapsto & x_i^t H_i \end{array}$$

où H_i est une matrice de parité systématique d'un code linéaire $\mathcal{C}_i$ de longueur n, de dimension $n-d$ et de distance duale $t+1$ sur $\mathbb{F}_q$.
Soit ψ un isomorphisme de $\mathbb{F}_{q^d}$ dans $\mathbb{F}_q^d$ et, pour tout entier $j>0$, on note Ψ_j l'isomorphisme associé :

$$\begin{array}{rccc} \Psi_j: & (\mathbb{F}_{q^d})^j & \to & \mathbb{F}_q^{dj} \\ & (x_1,\ldots,x_j) & \mapsto & (\psi(x_1),\ldots,\psi(x_j)) \end{array}$$

Soit h une fonction r-résiliente linéaire sur $\mathbb{F}_{q^d}$ définie par

$$\begin{array}{rccc} h: & (\mathbb{F}_q^d)^k & \to & \mathbb{F}_q^{d\ell} \\ & x & \mapsto & \Psi_\ell(\Psi_k^{-1}(x)^t H') \end{array}$$

où H' est une matrice de parité d'un code linéaire $\mathcal{E}$ de longueur k, de dimension $k-\ell$ et de distance duale $r+1$ sur $\mathbb{F}_{q^d}$.

Alors, la fonction composée $f=h\circ g$ est une fonction $((r+1)(t+1)-1)$-résiliente linéaire qui s'écrit

$$\begin{array}{rccc} f: & \mathbb{F}_q^{nk} & \to & \mathbb{F}_q^{d\ell} \\ & x & \mapsto & x^t M \end{array}$$

où M est une matrice de parité du code linéaire $\mathcal{C}=((\mathcal{C}_i^\perp)\square_\Theta\mathcal{E}^\perp)^\perp$ de longueur kn, de dimension $kn-d\ell$ et de distance duale $(r+1)(t+1)$, avec l'isomorphisme d'espaces vectoriels $\Theta=(\theta_1,\ldots,\theta_k)$ défini par :

$$\begin{array}{rccc} \theta_i: & \mathbb{F}_{q^d} & \to & \mathcal{C}_i^\perp \\ & x & \mapsto & \psi(x)H_i \end{array}$$

preuve : La fonction $f=h\circ g$ étant linéaire, il suufit de montrer que $f^{-1}(0)=\mathcal{C}$. Soit $x=(x_1,\ldots,x_k)\in(\mathbb{F}_q^n)^k$ un mot du code $\mathcal{C}$ et v son image par la fonction g Considérons le vecteur $\bar{v}\in(\mathbb{F}_q^n)^k$ défini à partir de v par $\bar{v}_i=(v_i,0,\ldots,0)$ pour $1\leq i\leq k$. Les matrices H_i étant sous forme systématique, on a $g(\bar{v})=v=g(x)$. Comme les fonctions g_i sont des fonctions syndromes relativement aux codes $\mathcal{C}_i$, on en déduit donc que

$$\forall\ 1\leq i\leq k,\ \ x_i\in\bar{v}_i+\mathcal{C}_i$$

ce qui implique que

$$\forall u \in \mathcal{E}^{\perp},\ \ \bar{v}.\Theta(u) = x.\Theta(u) = 0$$

Or, les $(n-d)$ dernières composantes des vecteurs $\bar{v}_i$ sont nulles et, les matrices H_i étant systématiques, les d premières composantes de $\theta_i(u_i)$ correspondent au vecteur $\psi(u_i)$. On obtient par conséquent pour tout u de $\mathcal{E}^{\perp}$

$$\begin{aligned} \bar{v}.\Theta(u) &= \sum_{i=1}^{k} v_i.\psi(u_i) \\ &= \Psi_k^{-1}(v).u = 0 \end{aligned}$$

ce qui signifie que $\Psi_k^{-1}(v)$ est un mot du code $\mathcal{E}$, et donc que $f(x) = 0$, par définition de la fonction h. Par égalité des dimensions des $\mathbb{F}_q$-espaces vectoriels $\mathcal{C}$ et $Ker(f)$, on montre donc que les éléments de $f^{-1}(0)$ sont exactement les vecteurs du code $\mathcal{C}$ et que la fonction f calcule le syndrome d'un mot relativement à ce code. □

Ce théorème me permet donc non seulement de décrire les fonctions résilientes résultant de la composition de fonctions linéaires, mais aussi d'expliciter la forme du dual d'un code concaténé.

Corollaire 9.13 (Dual d'un code concaténé) *Soit $(\mathcal{B}_i)_{1\leq k\leq n_e}$ une famille de codes linéaires $[n_b, k_b]$ sur $\mathbb{F}_q$, $\mathcal{E}$ un code linéaire $[n_e, k_e]$ sur $\mathbb{F}_{q^{k_b}}$ et $\Theta = (\theta_1, \ldots, \theta_{n_e})$ un isomorphisme $\mathbb{F}_q$-linéaire de $\mathbb{F}_{q^{k_b}}^{n_e}$ dans $\mathcal{B}_1 \times \ldots \times \mathcal{B}_{n_e}$ défini par*

$$\begin{array}{rccc} \theta_i\colon & \mathbb{F}_{q^{k_b}} & \rightarrow & \mathcal{B}_i \\ & x_i & \mapsto & \psi(x_i)G_i \end{array}$$

où G_i est une matrice génératrice systématique de $\mathcal{B}_i$ et ψ un isomorphisme de $\mathbb{F}_{q^{k_b}}$ dans $\mathbb{F}_q^{k_b}$.

Alors le dual du code concaténé $(\mathcal{C}_i)\square_{\Theta}\mathcal{E}$ est constitué des mots de $\mathcal{E}^{\perp}$ dont chaque composante $y_i \in \mathbb{F}_{q^{k_b}}$ est remplacée par un mot de $\mathbb{F}_q^{n_b}$ ayant pour syndrome y_i relativement au code $\mathcal{C}_i^{\perp}$.

$$((\mathcal{C}_i)\square_{\Theta}\mathcal{E})^{\perp} = \{(x_1, \ldots, x_{n_e}) \text{ où } (\psi^{-1}(x_1^t G_1), \ldots, \psi^{-1}(x_{n_e}^t G_{n_e})) \in \mathcal{E}\ \perp\}$$

4 Application à la combinaison de LFSRs

Les fonctions résilientes obtenues par composition sont particulièrement adéquates pour combiner des LFSRs (Figure 9.1).

En effet, elles permettent d'une part de réduire le nombre d'opérations nécessaires pour calculer la sortie du générateur à partir des sorties des différents registres, puisque toutes les fonctions g_i sont évaluées en parallèle.

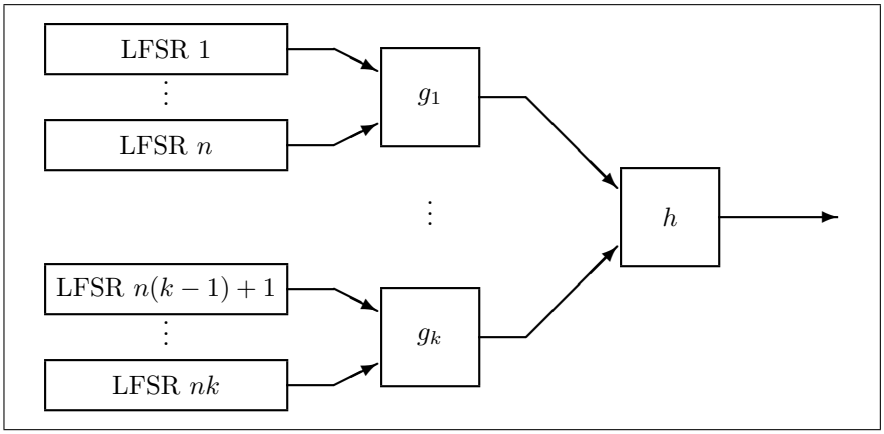


FIG. 9.1 –: *Combinaison de LFSRs à l'aide d'une fonction composée*

D'autre part, l'utilisation de cette construction réduit en général considérablement la quantité de données requise pour décrire la fonction de combinaison, ce qui constitue un paramètre fondamental lorsque celle-ci n'est pas câblée mais calculée en logiciel — on choisit souvent cette option pour pouvoir changer régulièrement la fonction de combinaison ou même l'utiliser comme une clef secrète du système. En effet, la manière la moins coûteuse pour décrire une fonction équilibrée à nk variables sur un alphabet à q éléments consiste à écrire la suite de ses valeurs, *i.e.* q^{nk} digits q-aires. Or, si l'on utilise la construction par composition décrite au début de ce chapitre (définition 9.1), il suffit de transmettre les valeurs des fonctions g_i et h, ce qui représente $kdq^n + q^{kd}$ digits. Ainsi, à l'exemple 2, j'ai construit une fonction booléenne permettant de combiner 12 LFSRs dont la description ne nécessite que 56 octets (32 si on prend $g_1 = g_2$), au lieu de 512 octets dans le cas général.

Conclusion de l'étude sur les fonctions résilientes

Cette étude a donc mis en évidence le rôle fondamental joué par les fonctions sans corrélation et les fonctions résilientes dans différents domaines de la cryptographie, en particulier pour la génération de suites pseudo-aléatoires et la conception de primitives cryptographiques conventionnelles.

Constatant que les propriétés de non-corrélation et de résilience étaient purement combinatoires et non algébriques, je les ai appréhendées dans un cadre plus vaste que celui que l'on considère habituellement : celui des fonctions définies sur un alphabet fini quelconque. J'ai alors caractérisé ces propriétés, initialement exprimées de manière probabiliste, à la fois en termes combinatoires (au moyen de la structure de tableau orthogonal), en termes de transformée de Fourier et en termes matriciels. Ces multiples points de vue apportent à ces objets une grande richesse car on peut les appréhender sous des angles tout à fait différents et complémentaires : l'approche combinatoire induit des bornes sur l'ordre de résilience admissible par une fonction, l'approche par la transformée de Fourier m'a permis, au chapitre 9, de construire de nouvelles fonctions sans corrélation...

Je me suis ensuite plus particulièrement intéressée aux fonctions définies sur un corps fini car on peut les représenter sous forme polynômiale. Cette autre approche est essentielle notamment si l'on utilise des fonctions résilientes pour combiner des registres à décalage à rétroaction linéaire car leur ordre de non-linéarité, donné par le degré de ce polynôme, conditionne à la fois la complexité linéaire et la période de la suite pseudo-aléatoire ainsi produite. J'ai alors montré qu'il existait un compromis entre l'ordre de non-linéarité et l'ordre de non-corrélation de toute fonction définie sur $\mathbb{F}_q$, généralisant ainsi l'inégalité établie par Siegenthaler pour les fonctions booléennes.

J'ai exploité ce résultat fondamental en construisant une famille de fonctions résilientes q-aires de non-linéarité optimale que j'ai utilisée pour assembler des registres à décalage à rétroaction linéaire q-aires. J'ai ainsi conçu un nouvel algorithme de chiffrement à flot à très haut débit — 180 Mbit/s sur une carte et plus de 1 Gbit/s sur une puce — qui résiste aux attaques par corrélation de Siegenthaler et à l'attaque par l'algorithme de Berlekamp-Massey. L'intérêt de l'utilisation des fonctions résilientes q-aires apparaît ici pleinement car un tel débit n'était pas accessible par les méthodes de classiques de combinaison de

registres à décalage binaires.

Une seconde application de l'inégalité régissant l'ordre de non-linéarité d'une fonction sans corrélation q-aire concerne les primitives cryptographiques conventionnelles : j'ai mis en évidence un nouveau critère que doivent respecter les fonctions utilisées dans les boîtes de diffusion de ces primitives afin d'éviter certaines attaques, notamment des attaques par collisions dans le cas des fonctions de hachage. Ce résultat constitue une avancée dans la tentative de théorisation des primitives cryptographiques conventionnelles entamée notamment par Serge Vaudenay, et il peut avoir de nombreuses retombées tant pour la cryptanalyse que pour la conception de nouveaux systèmes de chiffrement symétriques et de nouvelles fonctions de hachage.

Perspectives

Les deux thèmes que j'ai abordés dans ce mémoire pouvaient paraître à première vue aussi éloignés que possible au sein de la cryptologie : le premier entrait dans le cadre de la cryptanalyse et le second dans celui de la cryptographie proprement dite, c'est-à-dire la conception de nouveaux systèmes ; le premier concernait les systèmes à clef publique et le second les systèmes à clef secrète. Leurs traitements et les concepts mathématiques que j'ai utilisés dans ces deux parties sont pourtant très proches, puisque ces sujets se rattachent directement à des problèmes de combinatoire, de théorie des codes et de corps finis. Cette parenté paraissait évidente pour les cryptosystèmes à mots de poids faible ; elle était sous-jacente pour les algorithmes de chiffrement à flot depuis les travaux de Siegenthaler et de Camion, Carlet, Charpin et Sendrier sur les fonctions booléennes ; mais qui aurait spontanément imaginé que la sécurité des primitives cryptographiques conventionnelles était, elle aussi, liée à des propriétés de tableaux orthogonaux et de codes MDS ? Tout ceci sans mentionner le partage du secret [MS81, Mas93a], les codes d'authentification [vT95] et encore bien d'autres axes de recherche en cryptographie qui finalement rejoignent des structures connues de mathématiques discrètes [Mas93b].

Alors qu'elles semblaient n'avoir aucun rapport, les deux questions que je soulève maintenant — *Quelle alternative au RSA ?*, *Quels critères de sécurité pour les primitives conventionnelles ?* — peuvent pourtant être appréhendées de manière identique : les mots d'un code de Goppa et les fonctions résilientes q-aires ne se représentent-ils pas tous deux par des polynômes sur un corps fini ?

Quelle alternative au RSA ?

Le système de chiffrement de McEliece est actuellement la seule véritable alternative au RSA dans la mesure où il s'agit du seul système à clef publique qui ne repose pas sur un problème de théorie des nombres. L'attaque que j'ai exposée sous forme d'un algorithme général de décodage compromet la sécurité de ce système si l'on emploie les paramètres proposés à l'origine par McEliece, mais il ne s'agit pas d'une attaque structurelle du système dans le sens où il suffit d'augmenter légèrement la taille des paramètres pour s'en prémunir — notons que les clefs publiques deviendraient dans ce cas gigantesques...

La seule fragilité potentielle du système de McEliece me semble donc résider dans l'emploi des codes de Goppa comme clefs secrètes. Il est *a priori* moins

ardu de décoder un code de Goppa permuté que de décoder un code aléatoire ; il paraît donc imaginable que l'algorithme que j'ai conçu puisse exploiter, même sous forme d'une heuristique, certaines particularités de cette famille de codes. Mais il faudrait pour cela être capable de la caractériser et de la différentier des codes aléatoires.

J'ai d'autre part rappelé que quantité de familles de codes classiques (codes concaténés, codes de Reed-Solomon généralisés,...) ne pouvaient pas être utilisées dans le système de McEliece car il était possible de retrouver la structure de ces codes à partir de la donnée d'une matrice génératrice d'un code équivalent. Toutes les tentatives dans ce sens sur les codes de Goppa ont jusqu'à présent échoué mais je pense qu'il s'agit de la seule possibilité d'attaque structurelle sur le système de McEliece — qui, de surcroît, permettrait de retrouver la clef secrète au lieu de ne décrypter qu'un seul message. Vu leur nombre, il n'est évidemment pas question de caractériser chacun des codes de Goppa irréductibles de longueur 1024 et de degré 50, mais plutôt d'essayer de déterminer des invariants du code qui puissent apporter certaines informations sur le polynôme de Goppa du code secret.

Dans l'état actuel des connaissances, je dirais donc que la sécurité du système de McEliece avec ses paramètres originaux est insuffisante mais, qu'en l'absence d'attaque structurelle, cet algorithme à clef publique reste une véritable alternative au RSA, une fois ses paramètres modifiés.

Quant au système d'identification de Stern — je ne m'intéresserai qu'à celui-ci puisque j'ai donné une cryptanalyse de celui de Girault et montré que les paramètres choisis par Véron étaient moins fiables — il me semble, lui, hors de portée de toute attaque structurelle. Le fait qu'il utilise, contrairement au système de McEliece, un code aléatoire comme clef secrète ne permet en effet pas de l'attaquer par un problème moins général que celui que j'ai étudié. Seule une faille dans le protocole interactif, telle celle que j'ai mise en évidence dans le schéma de Girault, serait à mon avis susceptible de remettre en cause sa sécurité.

Quels critères de sécurité pour les primitives conventionnelles ?

Le monde de la cryptographie à clef secrète semble des plus hétéroclites : qu'ont donc en commun un générateur pseudo-aléatoire à base de registres à décalage à rétroaction linéaire et une fonction de hachage comme MD4 ? Pourtant, le fait qu'ils reposent sur les mêmes structures mathématiques n'est pas le fruit du hasard. Pour résister à la cryptanalyse, tout cryptosystème, quel qu'il soit, doit à la fois contenir des fonctions de diffusion et de confusion. J'ai montré ici que le premier de ces concepts se ramenait à la propriété de non-corrélation, alors que le second est lié à la notion de non-linéarité [MS90, Car94].

Il paraît alors naturel d'étudier les fonctions employées dans les systèmes conventionnels à travers la structure des codes de Reed-Muller généralisés. Une

voie de recherche consisterait par exemple à affiner la notion d'ordre de non-linéarité que j'ai utilisée dans ce travail en étudiant le *spectre de non-linéarité* des fonctions sans corrélation, c'est-à-dire leur distance à tous les codes de Reed-Muller généralisés d'ordre inférieur au degré de leur forme algébrique normale. Une fonction peut en effet posséder un ordre de non-linéarité très élevé mais être en même temps très proche d'une fonction de faible degré. Une telle situation induit alors des faiblesses cryptographiques telles celles décelées par Knudsen et Robshaw sur le DES [KR96]. Cette vision tend également à unifier les approches jusqu'ici très différentes des fonctions sans corrélation et des fonctions courbes.

Enfin, j'ai montré au chapitre 6 que les propriétés de non-corrélation et de résilience étaient purement combinatoires et qu'elles ne nécessitaient par conséquent aucune structure algébrique particulière sur les ensembles considérés. Cela signifie qu'il est possible, comme l'a fait Jim Massey dans l'algorithme SAFER, de passer d'une structure algébrique à une autre tout en respectant les critères de diffusion, ce qui pourrait accroître fortement la non-linéarité d'un système.

Annexe A

Quelques notions de théorie des codes

Je rappelle brièvement dans cette annexe quelques définitions et résultats fondamentaux de théorie des codes ; la plupart de ces énoncés sont extraits des ouvrages de référence de MacWilliams et Sloane [MS77] et de van Lint [vL82].

1 Concepts de base

Soit $\mathcal{F}$ un alphabet à q éléments. Pour tout entier $n > 0$, je note $\mathcal{F}^n$ l'ensembles des mots de longueur n à composantes dans $\mathcal{F}$.

Définition A.1 *Un* code $\mathcal{C}$ *sur l'alphabet* $\mathcal{F}$ *de* longueur n *et de* taille M *est un ensemble de* M *mots distincts de* $\mathcal{F}^n$*.*

On munit alors $\mathcal{F}^n$ d'une métrique, la métrique de Hamming : la *distance de Hamming* entre deux mots $x = (x_0, \ldots, x_{n-1})$ et $y = (y_0, \ldots, y_{n-1})$ de $\mathcal{F}^n$ est

$$d(x, y) = |\{i,\ 0 \leq i \leq n-1,\ x_i \neq y_i\}|$$

Lorsque $\mathcal{F}$ est un groupe et 0 son élément neutre, on appelle *poids de Hamming* d'un mot x de $\mathcal{F}^n$ le nombre de ses coordonnées non nulles :

$$w(x) = d(x, 0)$$

Définition A.2 (Distance minimale d'un code)

– *La* distance minimale d *d'un code est la plus petite distance entre deux mots de code, c'est-à-dire*

$$d = \min_{\substack{x, y \in \mathcal{C} \\ x \neq y}} d(x, y)$$

– *Lorsque $\mathcal{F}$ est un groupe, le* poids minimal *du code est, de façon similaire, l'entier*

$$\min_{\substack{x \in \mathcal{C} \\ x \neq 0}} w(x)$$

– *La* distribution des poids *d'un code de longueur n est la suite $(A_0, \ldots, A_n)$ où A_i est le nombre de mots de code de poids i.*

La distance minimale est un paramètre fondamental puisqu'elle détermine à la fois la capacité de détection et la capacité de correction d'un code. En effet, un code de distance minimale d permet de détecter $d-1$ erreurs et d'en corriger $t = \lfloor \frac{d-1}{2} \rfloor$.

Pour un code de longueur et de taille données, elle ne peut toutefois pas prendre n'importe quelle valeur : elle est limitée par la *borne de Singleton.*

Proposition A.3 (Borne de Singleton) *Soit $\mathcal{C}$ un code de longueur n, de taille M et de distance minimale d sur un alphabet $\mathcal{F}$ à q éléments. Alors*

$$M \leq q^{n-d+1}$$

Un code atteignant cette borne est appelé code MDS *(de l'anglais "Maximum Separable Distance").*

2 Codes linéaires

Les codes les plus étudiés sont les codes linéaires définis sur un corps fini $\mathbb{F}_q$ qui, grâce à leur structure algébrique, sont très faciles à décrire, à encoder et à décoder.

Définition A.4 (Code linéaire) *Un code $\mathcal{C}$ de longueur n sur le corps fini $\mathbb{F}_q$ est* linéaire *s'il constitue un sous-espace vectoriel de $\mathbb{F}_q^n$. On parle alors de sa* dimension*, k, en tant qu'espace vectoriel et on note $[n, k, d]$ ses paramètres.*

La borne de Singleton implique que la distance minimale d'un code linéaire de longueur n et de dimension k vérifie $d \leq n - k + 1$.

Par définition, il est possible de représenter un code linéaire de dimension k par k vecteurs de base. On écrit généralement ces vecteurs sous forme d'une matrice à k lignes et n colonnes, appelée *matrice génératrice* du code. Une matrice génératrice est dite *systématique* si elle s'écrit $(I_k|Z)$.

De manière similaire, une *matrice de parité* d'un code linéaire $[n, k]$ est une matrice H à $(n - k)$ lignes et n colonnes vérifiant

$$\mathcal{C} = \mathrm{Ker} H = \{x,\ x\ {}^tH = 0\}$$

Proposition A.5 *Soit $\mathcal{C}$ un code linéaire de longueur n et de dimension k, et $G = (I_k|Z)$ une matrice génératrice systématique de $\mathcal{C}$. Alors $H = (-{}^tZ|I_{n-k})$ est une matrice de parité de $\mathcal{C}$.*

Ces outils classiques d'algèbre linéaire permettent également d'associer à tout code linéaire son dual :

Définition A.6 (Dual d'un code linéaire) *Le* dual $\mathcal{C}^\perp$ *d'un code linéaire* $\mathcal{C}$ *de longueur* n *et de dimension* k *sur* $\mathbb{F}_q$ *est le code linéaire de longueur* n *et de dimension* $(n-k)$ *orthogonal de* $\mathcal{C}$ *dans* $\mathbb{F}_q^n$ *pour le produit scalaire usuel,* $x.y = \sum_{i=0}^{n-1} x_i y_i$.

Toute matrice de parité de $\mathcal{C}$ est donc une matrice génératrice de son dual.

On utilise également la notion de matrice de parité pour détecter et corriger des erreurs. Elle permet en effet, grâce à l'application syndrome, de définir une partition des mots de l'espace.

Définition A.7 (Syndrome) *Soit* $\mathcal{C}$ *un code linéaire de longueur* n *et de dimension* k *sur* $\mathbb{F}_q$ *et* H *une matrice de parité de* $\mathcal{C}$. *Le* syndrome *d'un vecteur* x *de* $\mathbb{F}_q^n$ *est le vecteur* s *de* $\mathbb{F}_q^{n-k}$ *défini par*

$$s = x\ {}^tH$$

L'application qui, à tout mot de l'espace, associe son syndrome est donc $\mathbb{F}_q$-linéaire. Elle induit alors une relation d'équivalence sur l'espace $\mathbb{F}_q^n$ définie par

$$\begin{aligned} x\mathcal{R}y &\iff x\ {}^tH = y\ {}^tH \\ &\iff (x-y) \in \mathcal{C} \end{aligned}$$

La classe d'équivalence d'un mot x de $\mathbb{F}_q^n$ est appelée *coset de* x (ou *translaté*) et est notée $(x + \mathcal{C})$. Elle correspond à tous les mots de l'espace de syndrome $x\ {}^tH$.

Ainsi décoder un mot x dont au plus t coordonnées sont erronées revient à rechercher le mot de plus petit poids dans le coset de x.

3 Groupe d'automorphismes - Codes cycliques

Deux codes $\mathcal{C}_1$ et $\mathcal{C}_2$ de longueur n et de taille M sont dits *équivalents* (par permutation) si l'un est l'image de l'autre par une permutation des coordonnées, c'est-à-dire s'il existe une permutation π de $\{0, \ldots, n-1\}$ telle que

$$(x_0, \ldots, x_{n-1}) \in \mathcal{C}_1 \iff (x_{\pi(0)}, \ldots, x_{\pi(n-1)}) \in \mathcal{C}_2$$

Toute permutation étant une isométrie pour la métrique de Hamming, deux codes équivalents ont en particulier la même distribution des poids.

Parmi toutes les permutations possibles des coordonnées, on distingue celles qui laissent le code invariant :

Définition A.8 (Groupe des permutations d'un code) *Le* groupe des permutations *d'un code* $\mathcal{C}$ *de longueur* n *sur un alphabet fini* $\mathcal{F}$ *est l'ensemble des permutations de* $\{0, \ldots, n-1\}$ *qui laissent* $\mathcal{C}$ *globalement invariant.*

Plus généralement, on appelle *groupe d'automorphismes* d'un code de longueur n défini sur le corps $\mathbb{F}_q$ l'ensemble des applications monômiales qui laissent le code globalement invariant, où π est une application monômiale [MS77, page] de $\mathbb{F}_q^n$ s'il existe une permutation σ de $\{0, \ldots, n-1\}$ et n éléments non nuls $a_0, \ldots, a_{n-1}$ de $\mathbb{F}_q$ tels que

$$\pi(x_0, \ldots, x_{n-1}) = (a_0 x_{\pi(0)}, \ldots, a_{n-1} x_{\pi(n-1)})$$

Le groupe d'automorphismes d'un code binaire s'identifie donc à son groupe de permutations.

Certaines classes de codes sont définies par une propriété de leur groupe d'automorphismes ou de leur groupe de permutations. Ainsi les *codes cycliques* de longueur n sont les codes linéaires dont le groupe de permutations contient le groupe cyclique d'ordre n. Lorsque l'on a introduit un ordre sur les coordonnées des mots de $\mathcal{F}^n$, cette propriété peut être définie par :

Définition A.9 (Code cyclique) *Un code linéaire $\mathcal{C}$ de longueur n sur $\mathbb{F}_q$ est* cyclique *si et seulement si*

$$(x_0, \ldots, x_{n-2}, x_{n-1}) \in \mathcal{C} \Longleftrightarrow (x_{n-1}, x_0, \ldots, x_{n-2}) \in \mathcal{C}$$

Si l'on identifie chaque mot $c = (c_0, \ldots, c_{n-1})$ de $\mathbb{F}_q^n$ au polynôme $c(X) = \sum_{i=0}^{n-1} c_i X^i$ de l'algèbre $\mathcal{R}_n = \mathbb{F}_q^n/(X^n - 1)$, transformer c en son shifté revient à multiplier $c(X)$ par X. Un code cyclique de longueur n sur $\mathbb{F}_q$ est donc un idéal de $\mathcal{R}_n$. Comme l'algèbre $\mathcal{R}_n$ est une algèbre principale, tout code cyclique est engendré par son polynôme unitaire de plus bas degré, appelé *polynôme générateur* du code. La proposition suivante résume donc les propriétés fondamentales de ce polynôme.

Proposition A.10 (Polynôme générateur d'un code cyclique) *Soit $\mathcal{C}$ un code cyclique de longueur n sur le corps fini $\mathbb{F}_q$, non réduit à $\{0\}$. Il existe alors un polynôme g de $\mathcal{C}$ vérifiant les propriétés suivantes :*

(i) g est l'unique polynôme unitaire de degré minimal de $\mathcal{C}$.

(ii) $\mathcal{C}$ est l'idéal de $\mathcal{R}_n$ engendré par g.

(iii) Tout mot $c(X)$ de $\mathcal{C}$ s'écrit de façon unique $c(X) = f(X)g(X)$.

(iv) $g(X)$ divise $X^n - 1$.

(v) g est un produit de polynômes minimaux sur $\mathbb{F}_q$. Il s'écrit donc

$$g(X) = \prod_{i \in I} (X - \alpha^i)$$

où α est une racine primitive n-ième de l'unité dans $\mathbb{F}_q$ et I est une réunion de classes cyclotomiques modulo n.

On appelle *zéros d'un code cyclique* les racines de son polynôme générateur et *ensemble de définition* l'ensemble I défini précédemment.

4 Construction de nouveaux codes à partir de codes connus

Quantité de méthodes permettent d'obtenir de nouveaux codes à partir de codes connus. Je n'en présente ici que trois.

Code étendu

Etendre un code consiste à lui ajouter un symbole de parité.

Définition A.11 (Code étendu) *Soit $\mathcal{C}$ un code de longueur n et de taille M sur un groupe fini $\mathcal{F}$. Le* code étendu *correspondant, $\hat{\mathcal{C}}$ est le code de longueur $n+1$ et de taille M défini par*

$$\hat{\mathcal{C}} = \{(c_0, \ldots, c_{n-1}, c_n) \text{ où } (c_0, \ldots, c_{n-1}) \in \mathcal{C} \text{ et } c_n = -\sum_{i=0}^{n-1} c_i\}$$

Code poinçonné

Poinçonner un code est l'opération inverse de la précédente. Il s'agit simplement de détruire une ou plusieurs coordonnées.

Définition A.12 (Code poinçonné) *Soit $\mathcal{C}$ un code de longueur n et de taille M sur un alphabet fini $\mathcal{F}$. Le* code poinçonné *en position i correspondant, $\mathcal{C}^\star$, est le code de longueur $n-1$ et de taille M défini par*

$$\mathcal{C}^\star = \{(c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{n-1}) \text{ où } (c_0, \ldots, c_{i-1}, c_i, c_{i+1}, \ldots, c_{n-1}) \in \mathcal{C}\}$$

Code raccourci

L'opération consistant à raccourcir un code diffère de la précédente dans le sens où l'on ne conserve que les mots dont la i-ème coordonnée était nulle. Contrairement au code poinçonné, un code raccourci est de taille inférieure à celle du code de départ.

Définition A.13 (Code raccourci) *Soit $\mathcal{C}$ un code de longueur n sur un groupe fini $\mathcal{F}$. Le* code raccourci *en position i correspondant, $\mathcal{C}^+$, est le code de longueur $n-1$ défini par*

$$\mathcal{C}^+ = \{(c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{n-1}) \text{ où } (c_0, \ldots, c_{i-1}, 0, c_{i+1}, c_{n-1}) \in \mathcal{C}\}$$

Bibliographie

[ACS92] Augot (D.), Charpin (P.) et Sendrier (N.). – Studying the locator polynomial of minimum weight codewords of BCH codes. *IEEE Transactions on Information Theory*, vol. IT-38, n° 3, 1992, pp. 960–973.

[AM87] Adams (C.M.) et Meijer (H.). – Security-related comments regarding McEliece's public-key cryptosystem. *In : Advances in Cryptology - CRYPTO'87*, éd. par Pomerance (C.), Lecture Notes in Computer Science, n° 293. pp. 224–228. – Springer-Verlag, 1987.

[AS94] Augot (D.) et Sendrier (N.). – Idempotents and the BCH bound. *IEEE Transactions on Information Theory*, vol. IT-40, n° 1, 1994, pp. 204–207.

[Aug93] Augot (D.). – *Etude algébrique des mots de poids minimum des codes cycliques, méthodes d'algèbre linéaire sur les corps finis.* – Thèse de doctorat, Université Paris 6, 1993.

[BB95] Bland (J.A.) et Baylis (D.J.). – *Minimum distance as a combinatorial search problem.* – Rapport de recherche 7/95, Nottingham Trent University, septembre 1995.

[BBR88] Bennett (C.H.), Brassard (G.) et Robert (J.-M.). – Privacy amplification by public discussion. *SIAM J. Computing*, vol. 17, n° 2, avril 1988, pp. 210–229.

[Ber68] Berlekamp (E.R.). – *Algebraic Coding Theory.* – McGraw-Hill, 1968.

[BGG93] Baritaud (T.), Gilbert (H.) et Girault (M.). – FFT-Hashing is not collision-free. *In : Advances in Cryptology - EUROCRYPT'92*, éd. par Rueppel (R.A.), Lecture Notes in Computer Science, n° 658. pp. 35–44. – Springer-Verlag, 1993.

[BGS94] Bierbrauer (J.), Gopalakrishnan (K.) et Stinson (D.R.). – Bounds for resilient functions and orthogonal arrays. *In : Advances in Cryptology - CRYPTO'94*, éd. par Desmedt (Y.G.), Lecture Notes in Computer Science, n° 839, pp. 247–256. – 1994.

[BMvT78] Berlekamp (E.R.), McEliece (R.J.) et van Tilborg (H.C.A.). – On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, vol. IT-24, n° 3, 1978, pp. 384–386.

[BN90] Bruck (J.) et Naor (M.). – The hardness of decoding linear codes with preprocessing. *IEEE Transactions on Information Theory*, vol. 36, n° 2, 1990, pp. 381–385.

[BO92] Brickell (E.F.) et Odlyzko (A.M.). – Cryptanalysis: A survey of recent results. *In : Comtemporary Cryptology - the Science of Information Integrity*, éd. par Simmons (G.J.). pp. 501–540. – IEEE Press, 1992.

[BRC60] Bose (R.C.) et Ray-Chaudhuri (D.K.). – On a class of error correcting binary group codes. *Information and Control*, n° 3, 1960, pp. 68–79.

[Bry86] Brynielsson (L.). – On the linear complexity of combined shift register sequences. *In : Advances in Cryptology - EUROCRYPT '85*, éd. par Pichler (F.), Lecture Notes in Computer Science, n° 219. pp. 156–160. – Springer-Verlag, 1986.

[Bry89] Brynielsson (L.). – A short proof of the Xiao-Massey lemma. *IEEE Transactions on Information Theory*, vol. 35, n° 6, 1989, p. 1344.

[BT93] Brouwer (A.E.) et Tolhuizen (L.M.G.M.). – A sharpening of the Johnson bound for binary linear codes and the non-existence of linear codes with Preparata parameters. *Designs, Codes and Cryptography*, n° 3, 1993, pp. 95–98.

[Bus52] Bush (K.A.). – Orthogonal arrays of index unity. *Ann. Math. Stat.*, vol. 23, 1952, pp. 426–434.

[BV93] Brouwer (A.E.) et Verhoeff (T.). – An updated table of minimum-distance bounds for binary linear codes. *IEEE Transactions on Information Theory*, vol. 39, 1993, pp. 662–677. – également disponible sur ftp://ftp.win.tue.nl/pub/math/codes/table[234].gz.

[Can95] Canteaut (A.). – A new algorithm for finding minimum-weight words in large linear codes. *In : Cryptography and Coding - 5th IMA Conference*, éd. par Boyd (C.), Lecture Notes in Computer Science, n° 1025. pp. 205–212. – Springer-Verlag, 1995.

[Car94] Carlet (C.). – *Fonctions booléennes en théorie des codes correcteurs d'erreurs et en cryptologie.* – Habilitation à diriger des recherches, Université de Picardie, 1994.

[CC93] Chabanne (H.) et Courteau (B.). – *Application de la méthode de décodage itérative d' Omura à la cryptanalyse du système de McEliece.* – Rapport de recherche 122, Canada, Université de Sherbrooke, octobre 1993.

[CC94] Canteaut (A.) et Chabanne (H.). – A further improvement of the work factor in an attempt at breaking McEliece's cryptosystem. *In : EUROCODE 94 - Livre des Résumés*, éd. par Charpin (P.). – INRIA, 1994.

[CC95a] Canteaut (A.) et Chabaud (F.). – *Improvements of the attacks on cryptosystems based on error-correcting codes.* – Rapport de recherche LIENS-95-21, Ecole Normale Supérieure, Juillet 1995.

[CC95b] Canteaut (A.) et Chabaud (F.). – A new algorithm for finding minimum-weight words in a linear code: application to primitive narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, 1995. – à paraître. Paru sous forme de rapport de recherche INRIA RR-2685.

[CC96a] Camion (P.) et Canteaut (A.). – Construction of t-resilient functions over a finite alphabet. *In : Advances in Cryptology - EUROCRYPT'96*, éd. par Maurer (U.), Lecture Notes in Computer Science, n° 1070. pp. 283–293. – Springer-Verlag, 1996.

[CC96b] Camion (P.) et Canteaut (A.). – Generalization of Siegenthaler inequality and Schnorr-Vaudenay multipermutations. *In : Advances in Cryptology - CRYPTO'96*, éd. par Koblitz (N.), Lecture Notes in Computer Science, n° 1109. – Springer-Verlag, 1996.

[CCCS92] Camion (P.), Carlet (C.), Charpin (P.) et Sendrier (N.). – On correlation-immune functions. *In : Advances in Cryptology - CRYPTO'91*, éd. par Feigenbaum (J.), Lecture Notes in Computer Science, n° 576. pp. 86–100. – Springer-Verlag, 1992.

[CCO69] Chien (R.T.), Cunningham (B.D.) et Oldham (I.B.). – Hybrid method for finding roots of a polynomial with application to BCH decoding. *IEEE Transactions on Information Theory*, vol. 15, 1969, pp. 329–335.

[CG90] Coffey (J.T.) et Goodman (R.M.). – The complexity of information set decoding. *IEEE Transactions on Information Theory*, vol. IT-36, n° 5, 1990, pp. 1031–37.

[CGH+85] Chor (B.), Goldreich (O.), Hastad (J.), Freidmann (J.), Rudich (S.) et Smolensky (R.). – The bit extraction problem or t-resilient functions. *In : Proc. 26th IEEE Symposium on Foundations of Computer Science*, pp. 396–407. – 1985.

[Cha92] Chabaud (F.). – Asymtotic analysis of probabilistic algorithms for finding short codewords. *In : EUROCODE 92*, éd. par Camion (P.), Charpin (P.) et Harari (S.), CISM Courses and Lectures, n° 339. pp. 175–183. – Springer-Verlag, 1992.

[Cha96] Chabaud (F.). – *Recherche de performance dans l'algorithmique des corps finis. Applications à la cryptographie.* – Thèse de doctorat, Ecole Polytechnique, 1996.

[Cop96] Coppersmith (D.). – Finding a small root of a univariate modular equation. *In : Advances in Cryptology - EUROCRYPT'96*, éd. par Maurer (U.), Lecture Notes in Computer Science, n° 1070. pp. 155–165. – Springer-Verlag, 1996.

[CR84] Chor (B.) et Rivest (R.L.). – A knapsack type public-key cryptosystem based on arithmetic in finite fields. *In : Advances in Cryptology - CRYPTO'84*, éd. par Blakley (G.R.) et Chaum (D.), Lecture Notes in Computer Science, n° 196. pp. 54–65. – Springer-Verlag, 1984.

[CV95] Chabaud (F.) et Vaudenay (S.). – Links between differential and linear cryptanalysis. *In : Advances in Cryptology - EUROCRYPT'94*, éd. par De Santis (A.), Lecture Notes in Computer Science, n° 950. pp. 356–365. – Springer-Verlag, 1995.

[Del72] Delsarte (P.). – Bounds for unrestricted codes, by linear programming. *Philips Research Reports*, n° 27, 1972, pp. 272–289.

[Del73a] Delsarte (P.). – *An algebraic approach to the association schemes of coding theory.* – Thèse, Université catholique de Louvain, 1973.

[Del73b] Delsarte (P.). – Four fundamental parameters of a code and their combinatorial signifiance. *Information and Control*, vol. 23, n° 5, décembre 1973, pp. 407–438.

[DH76] Diffie (W.) et Hellman (M.E.). – New directions in cryptography. *IEEE Transactions on Information Theory*, vol. IT-22, n° 6, 1976, pp. 644–654.

[Dob95] Dobbertin (H.). – Cryptanalysis of MD4. *In : Fast Software Encryption*, éd. par Gollmann (D.), Lecture Notes in Computer Science, n° 1039. pp. 53–69. – Springer-Verlag, 1995.

[Dob96] Dobbertin (H.). – Cryptanalysis of MD5 compress. *In : EUROCRYPT'96.* – 1996. Non publié.

[Ebe92] Eberle (H.). – A high-speed DES implementation for network applications. *In : Advances in Cryptology - CRYPTO'92*, éd. par Brickell (E.F.), Lecture Notes in Computer Science, n° 740. pp. 527–545. – Springer-Verlag, 1992.

[ElG84] ElGamal (T.). – A public-key cryptosystem and a signature scheme based on discrete logarithms. *In : Advances in Cryptology - CRYPTO'84*, éd. par Blakley (G.R.) et Chaum (D.), Lecture Notes in Computer Science, n° 196. pp. 10–18. – Springer-Verlag, 1984.

[Evs83] Evseev (G.S.). – Complexity of decoding for linear codes. *Problemy Peredachi Informatsii*, vol. 19, n° 1, 1983, pp. 3–8.

[Far] Farr (E.H.). – non publié.

[Fon95] Fontaine (C.). – *Etude exploratoire sur le rayon de recouvrement des codes de Reed-Muller d'ordre 1.* – Rapport de DEA, Ecole Normale Supérieure, Paris, 1995.

[For66] Forney, Jr. (G.D.). – *Concatenated codes.* – The MIT Press, 1966.

[FS86] Fiat (A.) et Shamir (A.). – How to prove yourself: practical solutions to identification and signature problems. *In : Advances in Cryptology - CRYPTO'86*, éd. par Odlyzko (A.M.), Lecture Notes in Computer Science, n° 263. pp. 186–194. – Springer-Verlag, 1986.

[Gib91a] Gibson (J.K.). – *Equivalent Goppa codes and McEliece's public key cryptosystem.* – Rapport de recherche, Londres, Department of Computer Science, Birbeck College, avril 1991.

[Gib91b] Gibson (J.K.). – Equivalent Goppa codes and trapdoors to McEliece's public key cryptosystem. *In : Advances in Cryptology - EUROCRYPT'91*, éd. par Davies (D.W.), Lecture Notes in Computer Science, n° 547. pp. 517–521. – Springer-Verlag, 1991.

[Gir90] Girault (M.). – A (non-practical) three-pass identification protocol using coding theory. *In : Advances in Cryptology - AUSCRYPT'90*, éd. par Seberry (J.) et Pieprzyk (J.), Lecture Notes in Computer Science, n° 453. pp. 265–272. – Springer-Verlag, 1990.

[GMR85] Goldwasser (S.), Micali (S.) et Rackoff (C.). – The knowledge complexity of interactive proof systems. *In : Proceedings of the 14th ACM Symposium on the Theory of Computing*, pp. 291–304. – 1985.

[Gol89] Golić (J. DJ.). – On the linear complexity of functions of periodic GF(q) sequences. *IEEE Transactions on Information Theory*, vol. IT-35, n° 1, 1989, pp. 69–75.

[GS72] Goethals (J.-M.) et Snover (S.L.). – Nearly perfect binary codes. *Discrete Math.*, n° 3, 1972, pp. 65–88.

[GS95] Gopalakrishnan (K.) et Stinson (D.R.). – Three characterizations of non-binary correlation-immune and resilient functions. *Designs, Codes and Cryptography*, vol. 5, 1995, pp. 241–251.

[Gui] Guillot (P.). – Algorithme pour le codage à poids constant. – non publié.

[Har88] Harari (S.). – A new authentication algorithm. *In : Coding Theory and Applications*, éd. par Cohen (G.) et Wolfmann (J.), Lecture Notes in Computer Science, n° 388. pp. 91–105. – Springer-Verlag, 1988.

[Hei87] Heiman (R.). – *On the security of cryptosystems based on linear error correcting codes.* – M.Sc. Thesis, Weizmann Institute of Science, Israël, 1987.

[Her77] Herlestam (T.). – On linearization of nonlinear combination of linear binary sequences generators. *In : International Symposium on Information Theory.* – Ithaca, New-York, 1977.

[Her82] Herlestam (T.). – On the complexity of functions of linear shift register sequences. *In : IEEE International Symposium on Information Theory.* – Les Arcs, France, 1982.

[Her83] Herlestam (T.). – On the complexity of certain crypto generators. *In : Security, IFIP/Sec'83*, éd. par Fåk (V.A.). – North-Holland, 1983.

[Her86] Herlestam (T.). – On functions of linear shift register sequences. *In : Advances in Cryptology - EUROCRYPT '85*, éd. par Pichler (F.), Lecture Notes in Computer Science, n° 219. pp. 119–129. – Springer-Verlag, 1986.

[Hoc59] Hocquenghem (A.). – Codes correcteurs d'erreurs. *Chiffres*, n° 2, 1959, pp. 147–156.

[HS73] Helgert (H.J.) et Stinaff (R.D.). – Shortened BCH codes. *IEEE Transactions on Information Theory*, 1973, pp. 818–820.

[HW38] Hardy (G.H.) et Wright (E.M.). – *An introduction to the theory of numbers.* – Oxford Science Publications, 1938, cinquième édition.

[Kas64] Kasami (T.). – A decoding procedure for multiple-error-correcting cyclic codes. *IEEE Transactions on Information Theory*, vol. IT-10, 1964, pp. 134–138.

[KL72] Kasami (T.) et Lin (S.). – Some results on the minimum weight of primitive BCH codes. *IEEE Transactions on Information Theory*, novembre 1972, pp. 824–825.

[KLP66] Kasami (T.), Lin (S.) et Peterson (W.W.). – *Some results on cyclic codes which are invariant under the affine group.* – Rapport de recherche AFCRL-66-622, Bedford, Massachusetts, Air Force Cambridge Research Laboratories, août 1966.

[Knu69] Knuth (D.E.). – *The art of computer programming - Seminumerical algorithms.* – Addison-Wesley, 1969.

[KR96] Knudsen (L.R.) et Robshaw (M.J.B.). – Non-linear approximations in linear cryptanalysis. *In : Advances in Cryptology - EUROCRYPT'96*, éd. par Maurer (U.), Lecture Notes in Computer Science, n° 1070. pp. 224–236. – Springer-Verlag, 1996.

[KS60] Kemeny (J.G.) et Snell (J.L.). – *Finite Markov chains.* – Springer-Verlag, 1960.

[KT69] Kasami (T.) et Tokura (N.). – Some remarks on BCH bounds and minimum weights of primitive binary BCH codes. *IEEE Transactions on Information Theory*, vol. 15, n° 3, 1969, pp. 408–413.

[Lan84] Lang (S.). – *Algebra.* – Addison Wesley, 1984.

[LB88] Lee (P.J.) et Brickell (E.F.). – An observation on the security of McEliece's public-key cryptosystem. *In : Advances in Cryptology - EUROCRYPT'88*, éd. par Günter (C.G.), Lecture Notes in Computer Science, n° 330. pp. 275–280. – Springer-Verlag, 1988.

[LCF90] Lin (M.-C.), Chang (T.-C.) et Fu (H.-L.). – Information rate of McEliece's public-key cryptosystem. *Electronics Letters*, vol. 26, n° 1, 1990, pp. 16–17.

[LDW94] Li (Y.X.), Deng (R.H.) et Wang (X.M.). – On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, vol. IT-40, n° 1, 1994, pp. 271–273.

[Leo88] Leon (J.S.). – A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, vol. 34, n° 5, 1988, pp. 1354–1359.

[LM90] Lai (X.) et Massey (J.L.). – A proposal for a new block encryption standart. *In : Advances in Cryptology - EUROCRYPT'90*, éd. par Damgård (I.B.), Lecture Notes in Computer Science, n° 473. pp. 369–404. – Springer-Verlag, 1990.

[LN83] Lidl (R.) et Niederreiter (H.). – *Finite fields.* – Cambridge University Press, 1983.

[Luc78] Lucas (E.). – Sur les congruences des nombres eulériens et des coefficients différentiels des fonctions trigonométriques suivant un module premier. *Bull. S.M.F.*, 1878, pp. 49–54.

[Mac63] MacWilliams (F.J.). – A theorem on the distribution of weights in a systematic code. *Bell Syst. Tech. J.*, vol. 42, 1963, pp. 79–94.

[Man94] Mantel (W.). – Residues of recurring series. *Nieuw Arch. Wisk.*, vol. 2, n° 1, 1894, pp. 172–184.

[Mas69] Massey (J.L.). – Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, vol. 15, janvier 1969, pp. 122–127.

[Mas93a] Massey (J.L.). – Minimal codewords and secret sharing. *In : Proceedings of the 6th Joint Swedish-Russian International Workshop on Information Theory*, pp. 276–279. – 1993.

[Mas93b] Massey (J.L.). – Some applications of coding theory in cryptography. *In : 4th IMA Conference on Cryptography and Coding.* – 1993.

[Mas94] Massey (J.L.). – SAFER K-64: a byte-oriented block-ciphering algorithm. *In : Fast Software Encryption*, éd. par Anderson (R.), Lecture Notes in Computer Science, n° 809. pp. 1–17. – Springer-Verlag, 1994.

[Mat93] Matsui (M.). – Linear cryptanalysis method for DES cipher. *In : Advances in Cryptology - EUROCRYPT'93*, éd. par Helleseth (T.), Lecture Notes in Computer Science, n° 765. pp. 386–397. – Springer-Verlag, 1993.

[Mau90] Maurer (U.M.). – *Provable security in cryptography.* – Thèse, ETH Zürich, 1990.

[McE72] McEliece (R.J.). – Weight congruences for p-ary cyclic codes. *Discrete Math.*, n° 3, 1972, pp. 177–192.

[McE78] McEliece (R.J.). – A public-key cryptosystem based on algebraic coding theory. *JPL DSN Progress Report*, 1978, pp. 114–116.

[MH78] Merkle (R.C.) et Hellman (M.). – Hiding and signatures in traptoor knapsacks. *IEEE Transactions on Information Theory*, vol. IT-24, n° 5, 1978, pp. 55–530.

[MI83] Matsumoto (T.) et Imai (H.). – A class of asymmetric cryptosystems based on polynomials over finite rings. *In : IEEE Internatinal Symposium on Information Theory*, pp. 131–132. – 1983.

[MI88] Matsumoto (T.) et Imai (H.). – Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. *In : Advances in Cryptology - EUROCRYPT'88*, éd. par Günter (C.G.), Lecture Notes in Computer Science, n° 330. pp. 419–453. – Springer-Verlag, 1988.

[MM90] Maurer (U.M.) et Massey (J.L.). – Perfect local randomness in pseudo-random sequences. *In : Advances in Cryptology - CRYPTO'89*, éd. par Brassard (G.), Lecture Notes in Computer Science, n° 435. pp. 100–112. – Springer-Verlag, 1990.

[MM91] Maurer (U.M.) et Massey (J.L.). – Local randomness in pseudo-random sequences. *Journal of Cryptology*, vol. 4, 1991, pp. 135–149.

[Mor79] Moreno (O.). – Symmetries of binary Goppa codes. *IEEE Transactions on Information Theory*, vol. IT-25, n° 5, 1979, pp. 609–612.

[MS77] MacWilliams (F.J.) et Sloane (N.J.A.). – *The theory of error-correcting codes*. – North-Holland, 1977.

[MS81] McEliece (R.J.) et Sarwate (D.V.). – On sharing secrets and Reed-Solomon codes. *Communications of the ACM*, n° 24, 1981, pp. 583–584.

[MS90] Meier (W.) et Staffelbach (O.). – Nonlinearity criteria for cryptographic functions. *In : Advances in Cryptology - EUROCRYPT'89*, éd. par Quisquater (J.-J.) et Vandewalle (J.), Lecture Notes in Computer Science, n° 434. pp. 549–562. – Springer-Verlag, 1990.

[Mul54] Muller (D.E.). – Application of Boolean algebra to switching circuit design and to error detection. *IEEE Transactions on Computers*, vol. 3, 1954, pp. 6–12.

[Nie86] Niederreiter (H.). – Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, vol. 15, n° 2, 1986, pp. 159–166.

[Omu72] Omura (J.K.). – Iterative decoding of linear codes by a modulo-2 linear programm. *Discrete Math*, n° 3, 1972, pp. 193–208.

[Pet65] Peterson (W.W.). – *On the weight structure and symetry of BCH codes.* – Rapport de recherche AFCRL-65-515, Bedford, Massachusetts, Air Force Cambridge Research Laboratories, juillet 1965.

[Pra62] Prange (E.). – The use of information sets in decoding cyclic codes. *IRE Transactions*, vol. IT-8, 1962, pp. S5–S9.

[PW61] Peterson (W.W.) et Weldon (E.J.). – *Error-Correcting Codes.* – MIT Press, 1961.

[Rao47] Rao (C.R.). – Factorial experiments derivable from combinatorial arrangements of arrays. *J. Roy. Statist.*, vol. 9, 1947, pp. 128–139.

[Ree54] Reed (I.S.). – A class of multiple-error-correcting codes and the decoding scheme. *IEEE Transactions on Information Theory*, vol. 4, 1954, pp. 38–49.

[Rio79] Riordan (J.). – *Combinatorial identities.* – R.E. Krieger Publishing Compagny, 1979.

[Rot76] Rothaus (O.S.). – On bent functions. *Journal of combinatorial Theory (A)*, vol. 20, 1976, pp. 300–305.

[RS87] Rueppel (R.A.) et Staffelbach (O.J.). – Products of linear recurring sequences with maximum complexity. *IEEE Transactions on Information Theory*, vol. 33, n° 1, 1987, pp. 124–131.

[RSA78] Rivest (R.L.), Shamir (A.) et Adleman (L.M.). – A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, vol. 21, n° 3, 1978, pp. 120–126.

[Rue86] Rueppel (R.A.). – *Analysis and Design of stream ciphers.* – Springer-Verlag, 1986.

[Sch88] Schnorr (C.P.). – On the construction of random number generators and random function generators. *In : Advances in Cryptology - EUROCRYPT'88*, éd. par Günther (C.G.), Lecture Notes in Computer Science, n° 330. pp. 225–232. – Springer-Verlag, 1988.

[Sch93] Schnorr (C.P.). – FFT-Hash II, efficient cryptographic hashing. *In : Advances in Cryptology - EUROCRYPT' 92*, éd. par Rueppel (R.A.), Lecture Notes in Computer Science, n° 658. pp. 45–54. – Springer-Verlag, 1993.

[Sch96] Schneier (B.). – *Applied cryptography.* – John Wiley and Sons, 1996, seconde édition.

[Sel66] Selmer (E.S.). – *Linear recurrence relations over finite fields.* – Thèse, Université de Bergen, Norvège, 1966.

[Sen91] SENDRIER (N.). – *Codes correcteurs d'erreurs à haut pouvoir de correction*. – Thèse de doctorat, Université Paris VI, décembre 1991.

[Sen94] SENDRIER (N.). – On the structure of a randomly permuted concatenated code. *In : EUROCODE 94 - Livre des résumés*, éd. par CHARPIN (P.). pp. 169–173. – INRIA, 1994.

[Sen95a] SENDRIER (N.). – Efficient generation of binary words of given weight. *In : Cryptography and Coding - 5th IMA Conference*, éd. par BOYD (C.), Lecture Notes in Computer Science, n° 1025. pp. 184–187. – Springer-Verlag, 1995.

[Sen95b] SENDRIER (N.). – *On the structure of a randomly permuted concatenated code*. – Rapport de recherche RR-2460, INRIA, janvier 1995.

[Sen96] SENDRIER (N.). – *An algorithm for finding the permutation between two equivalent binary codes*. – Rapport de recherche RR-2853, INRIA, avril 1996.

[Sha49] SHANNON (C.E.). – Communication theory of secrecy systems. *Bell system technical journal*, vol. 28, 1949, pp. 656–715.

[Sid94] SIDELNIKOV (V.M.). – A public-key cryptosystem based on binary Reed-Muller codes. *Discrete Math. Appl.*, vol. 4, n° 3, 1994, pp. 191–207.

[Sie84] SIEGENTHALER (T.). – Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, vol. IT-30, n° 5, septembre 1984, pp. 776–780.

[Sie85] SIEGENTHALER (T.). – Decrypting a class of stream ciphers using ciphertext only. *IEEE Transactions on Computers*, vol. C-34, n° 1, janvier 1985, pp. 81–84.

[SM95] STINSON (D.R.) et MASSEY (J.L.). – An infinite class of counterexamples to a conjecture concerning nonlinear resilient functions. *Journal of Cryptology*, vol. 8, n° 3, 1995, pp. 167–173.

[SS92] SIDELNIKOV (V.M.) et SHESTAKOV (S.O.). – On cryptosystems based on generalized Reed-Solomon codes. *Diskretnaya Math*, vol. 4, 1992, pp. 57–63. – en russe.

[Ste89a] STERN (J.). – An alternative to Fiat-Shamir protocol. *In : Advances in Cryptology - EUROCRYPT'89*, éd. par QUISQUATER (J.-J.) et VANDEWALLE (J.), Lecture Notes in Computer Science, n° 434. pp. 173–180. – Springer-Verlag, 1989.

[Ste89b] Stern (J.). – A method for finding codewords of small weight. *In : Coding Theory and Applications*, éd. par Cohen (G.) et Wolfmann (J.), Lecture Notes in Computer Science, n° 388. pp. 106–113. – Springer-Verlag, 1989.

[Ste93] Stern (J.). – A new identification scheme based on syndrome decoding. *In : Advances in Cryptology - CRYPTO'93*, éd. par Stinson (D.R.), Lecture Notes in Computer Science, n° 773. pp. 13–21. – Springer-Verlag, 1993.

[Sti93] Stinson (D.R.). – Resilient functions and large sets of orthogonal arrays. *Congressus Numer.*, vol. 92, 1993, pp. 105–110.

[SV94] Schnorr (C.P.) et Vaudenay (S.). – Parallel FFT-Hashing. *In : Fast Software Encryption*, éd. par Anderson (R.), Lecture Notes in Computer Science, n° 809. pp. 149–156. – Springer-Verlag, 1994.

[SV95] Schnorr (C.-P.) et Vaudenay (S.). – Black box cryptanalysis of hash networks based on multipermutations. *In : Advances in Cryptology - EUROCRYPT'94*, éd. par De Santis (A.), Lecture Notes in Computer Science, n° 950. pp. 47–57. – Springer-Verlag, 1995.

[Sze59] Szegö (G.). – *Orthogonal polynomials.* – American Mathematical Society Colloquium Publications, 1959.

[TCG91] Tardy-Corfdir (A.) et Gilbert (H.). – A known plaintext attack of FEAL-4 and FEAL-6. *In : Advances in Cryptology - CRYPTO'91*, éd. par Feigenbaum (J.), Lecture Notes in Computer Science, n° 576. pp. 172–182. – Springer-Verlag, 1991.

[Vau93] Vaudenay (S.). – FFT-Hash II is not yet collision-free. *In : Advances in Cryptology - CRYPTO'92*, éd. par Brickell (E.F.), Lecture Notes in Computer Science, n° 740. pp. 587–593. – Springer-Verlag, 1993.

[Vau95a] Vaudenay (S.). – *La sécurité des primitives cryptographiques.* – Thèse de doctorat, Université Paris 7, 1995.

[Vau95b] Vaudenay (S.). – On the need for multipermutations: cryptanalysis of MD4 and SAFER. *In : Fast Software Encryption*, éd. par Preneel (B.), Lecture Notes in Computer Science, n° 1008. pp. 286–297. – Springer-Verlag, 1995.

[Vér92] Véron (P.). – *Cryptographie et codes de Goppa.* – Rapport de DEA, Université d'Aix-Marseille II, 1992.

[Vér95a] VÉRON (P.). – Cryptanalysis of Harari's identification scheme. *In : Cryptography and Coding - 5th IMA Conference*, éd. par BOYD (C.), Lecture Notes in Computer Science, n° 1025. pp. 264–269. – Springer-Verlag, 1995.

[Vér95b] VÉRON (P.). – *Problème SD, Opérateur Trace, schémas d'identification et codes de Goppa.* – Thèse de doctorat, Université de Toulon et du Var, 1995.

[vL82] VAN LINT (J.H.). – *Introduction to Coding Theory.* – Springer-Verlag, 1982.

[vT88] VAN TILBURG (J.). – On the McEliece public-key cryptosystem. *In : Advances in Cryptology - CRYPTO'88*, éd. par GOLDWASSER (S.), Lecture Notes in Computer Science, n° 403. pp. 119–131. – Springer-Verlag, 1988.

[vT94] VAN TILBURG (J.). – *Security-analysis of a class of cryptosystems based on linear error-correcting codes.* – Thèse de doctorat, Technische Universiteit Eindhoven, 1994.

[vT95] VAN TILBORG (H.C.A.). – Authentication codes: an area where coding and cryptology meet. *In : Coding and Cryptography - 5th IMA Conference*, éd. par BOYD (C.), Lecture Notes in Computer Science, n° 1025. pp. 169–183. – Springer-Verlag, 1995.

[XM88] XIAO (G.) et MASSEY (J.L.). – A spectral characterization of correlation-immune combining functions. *IEEE Transactions on Information Theory*, vol. IT-34, n° 3, 1988, pp. 569–571.

[Zie59] ZIERLER (N.). – Linear recurring sequences. *J. Soc. Indus. Appl. Math.*, vol. 7, 1959, pp. 31–48.

[ZM73] ZIERLER (N.) et MILLS (W.H.). – Products of linear recurring sequences. *Journal of Algebra*, vol. 27, 1973, pp. 147–157.

[ZZ95] ZHANG (X.) et ZHENG (Y.). – On nonlinear resilient functions. *In : Advances in Cryptology - EUROCRYPT'95*, éd. par GUILLOU (L.) et QUISQUATER (J.J.), Lecture Notes in Computer Science, n° 921. pp. 274–288. – Springer-Verlag, 1995.

Index

absorbant, *voir* état
Adams, C.M., 33
Adleman, L.M., 9
Augot, D., 112, 114
authentification, **9–11**

Baritaud, T., 184
BCH, *voir* code, borne
Bennett, C.H., 123
Berlekamp, E.R., 20, 47, 157
Berlekamp-Massey, 20, 157
Bierbrauer, J., 144,
    voir aussi borne
borne
    BCH, **108**
    Bierbrauer, **144**
    Bush, **141**
    Gilbert-Varshamov, **43**
    Plotkin, **143**
    programmation linéaire, **141–142**
    Rao, **141**
    Singleton, **218**
Bose, R.C., 108
Brassard, G., 123
Brickell, E.F., 24,
    voir aussi décodage
Brynielsson, L., 166
Bush, K.A., *voir* borne

Camion, P., 121, 125
caractère, **126**
Carlet, C., 125, 214
Chabanne, H., 75, 76
Chabaud, F., 64, 75
chaîne de Markov
    de l'algorithme DCP, 81–83
    de l'algorithme de Stern, 81, 84
Charpin, P., 112, 125
chiffrement, **7**
    à clef publique, **9**
    à clef secrète, **8–9**
    à flot, **8**
    à mots de poids faible, **18–39**, 99–106
      McEliece, **18–22**, 24, 99–101
      Niederreiter, **22–26**, 99–101
      Sidelnikov, 28
    Chor-Rivest, 5
    DES, **8–9**
    ElGamal, 5
    IDEA, **8**
    Matsumoto-Imai, 5
    Merkle-Hellman, 5
    par blocs, **8**
    provably-secure, **180**
    RSA, **9**, 26
    SAFER, 182
Chor, B., 5, 123, 130
code
    BCH, 108–116
    concaténé, 29, **207**
    cyclique, **220**
      ensemble de définition, 220
      zéro, 220
    équivalent, **219**
    étendu, **221**
    Goppa, **29–39**
      3-correcteur, 36
    Justesen, 207
    Kerdock, 140
    linéaire, **218**
    MDS, 190, **218**
    poinçonné, 61, **221**
    raccourci, **221**
    Reed-Muller, 28

généralisé, 214
poinçonné, **110**
Reed-Solomon généralisé, 29
systématique, **139**
coefficient de complexité, **49**
Coffey, J.T., 54
combinaison de LFSRs, *voir* LFSR
complexité linéaire, *voir* LFSR
concaténé, *voir* code
confusion, **182**, 183
corrélation, *voir* sans corrélation
coset, **219**
courbe, 184
Courteau,B., 76
cryptanalyse par boîtes noires, **183**, 192

DCP, *voir* décodage
décodage
algorithmes exhaustifs, **50–51**
REC, **50**
REE, **50**
borné, **19**
complet, **48**
DCP, **61–63**
DCP itératif, **79**
itératif, **75–106**
Lee-Brickell, **56–58**
Leon, **58–61**
par ensembles d'information, **51–56**
Stern, **63–66**
Stern itératif, **80**
version duale, **70–71**
Delsarte, P., 129, 134, 135, 141, 190
DES, *voir* chiffrement
Diffie, W., 9
diffusion, **183**, 184
distance
de Hamming, **217**
duale, 134, **134**, 135
minimale, **217**
dual
d'un code linéaire, **219**
d'un sous-groupe, **135**

effacements, **56**
ElGamal, T., 5
ensemble d'information, **52**, *voir aussi* décodage
ensemble de redondance, **52**
entropie binaire, **43**
équations de Newton, **112**
équilibrée, **124**
état
absorbant, **86**
persistant, **85**
transient, **86**
Evseev, G.S., 56

facteur de travail, **49**
Farr, E.H., 109
FFT-Hashing, *voir* hachage
Fiat, A., 11
fonction
courbe, *voir* courbe
résiliente, *voir* résiliente
sans corrélation, *voir* sans corrélation
forme algébrique normale, **151**
degré, **169**, 176
Fourier, *voir* transformation

générateur pseudo-aléatoire localement parfait, **180**
générateur pseudo-aléatoire localement parfait, **182**
Gibson, J.K., 33–35
Gilbert, H., 184
Gilbert-Varshamov, *voir* borne
Girault, M., 184, *voir aussi* identification à mots de poids faible
Golić, J.DJ., 164
Goodman, R.M., 54
Gopalakrishnan, K., 125, 126, 130, 144
Goppa, *voir* code
groupe
d'automorphismes, **220**
des permutations, **219**

hachage, **10**
FFT-Hashing, **184–185**
MD4, 6
MD5, 6
Harari, S., 39
Helgert, H.J., 112
Hellman, M.E., 5
Hellman, M.E., 9
Herlestam, T., 163, 166
Hocquenghem, A., 108
hull, **28**

IDEA, *voir* chiffrement
idempotents, **112**
identification, **11**
à apport nul de connaissance, **11**
à mots de poids faible, **39–44**, 103–106
Girault, **39–40**
Stern, **40–41**, 103
Véron, **41–43**, 103
Fiat-Shamir, 11
Imai, H., 5

Justesen, *voir* code

Karatsuba, 26
Kasami, T., 56, 109, 111
Kerdock, *voir* code
Knudsen, L.R., 215
Krawtchouck, *voir* polynôme

Lee-Brickell, *voir* décodage
lemme de la combinaison linéaire, 130
Leon, J.S., *voir* décodage
LFSR, **152–160**
combinaison, 161–169, 209
complexité linéaire, **157**
période, 154, 159
polynôme caractéristique, **154**
polynôme caractéristique minimal, **156**
polynôme de rétroaction, **156**
série génératrice, **156**
Lin, S., 109, 111
Lucas, E., 166

MacWilliams, F.J., 108, 135, *voir aussi* transformation
Massey, J.L., 20, 126, 130, 139, 140, 157, 180, 182, 213
matrice
de parité, **218**
fondamentale, 86
génératrice, **218**
Matsumoto, T., 5
Maurer, U., 180
McEliece, R.J., 47, 52, *voir aussi* chiffrement à mots de poids faible, 213
MD4, *voir* hachage
MD5, *voir* hachage
Meijer, H., 33
Merkle, R.C., 5
Mills, W.H., 162
Moreno, O., 33
multipermutation, **184**, 189–199

Newton, *voir* équations de Newton
Niederreiter, H., *voir* chiffrement à mots de poids faible

Odlyzko, A.M., 24
Omura, J.K., 76

période, *voir* LFSR
persistant, *voir* état
Peterson, W.W., 109, 110
Plotkin, *voir* borne
poids
de Hamming, **217**
minimal, **218**
polynôme
caractéristique, *voir* LFSR
caractéristique minimal, *voir* LFSR
de Krawtchouk, **133**
de rétroaction, *voir* LFSR
générateur, **220**
Prange, E., 51
problème

de la détermination des poids des cosets, **47**
de la recherche de mots de poids minimal dans les cosets, **22**, 23
du décodage borné, **19**, 23
du décodage complet, **48**
du recouvrement, **54**
provably-secure, **180**

Rao, C.R., 125,
voir aussi borne
Ray-Chaudhuri, D.K., 108
recherche d'un mot de poids minimal, **67**
Reed-Muller, *voir* code
Reed-Solomon, *voir* code
registre à décalage à rétroaction linéaire, *voir* LFSR
résiliente, **124**
caractérisation, 126, 130, 131
composition, 202–203
forme algébrique normale, **151**
ordre de non-linéarite, **169**–**176**, 176
Rivest, R.L., 5
Rivest, R.L., 9
Robert, J.-M., 123
Robshaw, M.J.B., 215
Rothaus, O.S., 184
RSA, *voir* chiffrement
Rueppel, R.A., 164

SAFER, *voir* chiffrement
sans corrélation, **124**
caractérisation, 126, 130, 131
composition, 202–203
forme algébrique normale, **151**
ordre de non-linéarite, **169**–**176**, 176
Sarwate, D.V., 213
Schnorr, C.P., 180, 183, 184, 192, 198
Selmer, E.S., 162
Sendrier, N., 20, 28, 40, 112, 125
série génératrice, *voir* LFSR
Shamir, A., 9, 11
Shannon, C.E., 182
Shestakov, S.O., 29
Sidelnikov, V.M., 28, 29
Siegenthaler, T., 123, 169, 176
signature, **10**
Staffelbach, O.J., 164
Stern, J., *voir* identification à mots de poids faible, décodage
Stinaff, R.D., 112
Stinson, D.R., 125, 126, 130, 139, 140, 144
suite ML, **159**
syndrome, **219**

tableau orthogonal, **125**, 126, 129, 134, 189
Tokura, N., 111
transformation
Fourier, **129**
MacWilliams, **133**
transient, *voir* état
translaté, **219**

van Tilborg, H.C.A., 47, 213
van Tilburg, J., 53, 76
Vaudenay, S., 183, 184, 192, 198
Véron, P., 39,
voir aussi identification à mots de poids faible

Xiao, G., 126, 130

zero-knowledge, *voir* identification à apport nul de connaissance
Zhang, X., 206
Zheng, Y., 206
Zierler, N., 162

Table des figures

1.1 Principe général d'un algorithme de chiffrement 8
1.2 Algorithme de chiffrement RSA 9
1.3 Signature numérique utilisant l'algorithme RSA 10

2.1 Evolution du taux de transmission des systèmes de McEliece et de Niederreiter en fonction de la dimension des codes de Goppa irréductibles utilisés en longueur 1024 25

3.1 Motifs d'erreurs corrigés à chaque itération par l'algorithme de décodage par ensembles d'information 52
3.2 Coefficient de complexité des algorithmes de décodage classiques jusqu'à la capacité de correction 55
3.3 Coefficient de complexité des algorithmes de décodage classiques jusqu'à la distance minimale 55
3.4 Motifs d'erreurs corrigés à chaque itération par l'algorithme de Lee-Brickell 57
3.5 Motifs d'erreurs corrigés à chaque itération par l'algorithme de Leon (version originale) 58
3.6 Motifs d'erreurs corrigés à chaque itération par l'algorithme DCP 62
3.7 Motifs d'erreurs corrigés à chaque itération par l'algorithme de Stern 64
3.8 Forme des mots de poids de poids minimal détectés à chaque itération par l'algorithme DCP 68
3.9 Forme des mots de poids de poids minimal détectés à chaque itération par l'algorithme de Stern 69

4.1 Remise à jour de la forme systématique de la matrice génératrice pour des ensembles d'information voisins 78
4.2 Forme de la matrice de transition du processus markovien associé à l'algorithme de décodage DCP 84
4.3 Forme de la matrice de transition du processus markovien associé à l'algorithme de Stern 85
4.4 Taux théorique de réussite du décodage d'un code binaire aléatoire de longueur 256 et de dimension 128 en fonction du nombre d'itérations effectuées par l'algorithme de Stern itératif 89

4.5 Influence des paramètres p et σ sur le facteur de travail de l'algorithme de Stern utilisé pour décoder un code binaire aléatoire . 92
4.6 Taux expérimental de réussite du décodage d'un code binaire aléatoire de longueur 256 et de dimension 128 en fonction du nombre d'itérations effectuées par l'algorithme de Stern itératif . 93
4.7 Evolution du facteur de travail de l'algorithme de Stern itératif optimisé pour le décodage de codes binaires aléatoires $[n, nR]$ . . 95
4.8 Coefficient de complexité de l'algorithme de Stern itératif pour le décodage de codes binaires aléatoires 96
4.9 Evolution du facteur de travail de l'algorithme de Stern itératif optimisé pour la recherche d'un mot de poids minimal dans un code binaire aléatoire $[n, nR]$ 97
4.10 Coefficient de complexité de l'algorithme de Stern itératif pour la recherche d'un mot de poids minimal dans un code binaire aléatoire 98
4.11 Taux de réussite de l'attaque du système de chiffrement de McEliece en fonction du nombre d'itérations effectuées par l'algorithme de Stern itératif optimisé 101
4.12 Effort de calcul nécessaire pour cryptanalyser le système de McEliece en fonction du taux de déchiffrement 102
4.13 Effort de calcul nécessaire pour cryptanalyser le schéma d'identification de Stern en fonction du taux de réussite 103
4.14 Effort de calcul nécessaire pour cryptanalyser le schéma d'identification de Véron en fonction du taux de réussite 103

7.1 Fonctionnement d'un registre à décalage à rétroaction linéaire . . 152
7.2 Exemple de LFSR de longueur 10 158
7.3 LFSR minimal produisant la même suite que le précédent 158
7.4 Combinaison de plusieurs LFSRs 161
7.5 Exemple de produit de deux LFSRs de complexité linéaire optimale 165

8.1 Exemple de boîte de confusion 182
8.2 Exemple de boîte de diffusion 183
8.3 Description de la fonction de compression $g_{k,s}$ 185
8.4 Fonction de compression $g_{2,1}$ 186
8.5 Fonction de compression $g_{3,3}$ 186
8.6 Inversion de $g_{2,1}$ quand les boîtes ne sont pas des multipermutations 188
8.7 Réseau de collisions de la fonction de compression $g_{2,1}$ 191
8.8 Recherche de collisions pour $g_{2,1}$, premier cas 193
8.9 Recherche de collisions pour $g_{2,1}$, second cas 195

9.1 Combinaison de LFSRs à l'aide d'une fonction composée 210

Liste des tableaux

2.1 Système de chiffrement à clef publique de McEliece 18
2.2 Algorithme permettant de retrouver un texte clair à partir de deux de ses chiffrés par le système de McEliece 21
2.3 Système de chiffrement à clef publique de Niederreiter 23
2.4 Comparaison des performances des systèmes de McEliece et de Niederreiter 27
2.5 Classification des codes de Goppa $\Gamma(\mathbf{F}_{2^m}, g)$, où g est un polynôme de degré 3 37
2.6 Classes d'équivalence des codes de Goppa irréductibles 3-correcteurs en longueur $n = 128$ 38
2.7 Classes d'équivalence des codes de Goppa irréductibles 3-correcteurs en longueur $n = 256$ 38
2.8 Schéma d'identification de Girault 40
2.9 Schéma d'identification de Stern 41
2.10 Schéma d'identification de Véron 42
2.11 Comparaison des performances des différents schémas d'identification à mots de poids faible 43

3.1 Décodage par recherche exhaustive parmi tous les mots du code . 50
3.2 Décodage par recherche exhaustive parmi tous les vecteurs d'erreurs 51
3.3 Décodage par ensembles d'information 52
3.4 Algorithme de Lee-Brickell 57
3.5 Algorithme de Leon (version originale) 59
3.6 Variation, en fonction du rang des colonnes sélectionnées, du nombre d'opérations binaires effectuées lors d'une itération de l'algorithme de Leon 60
3.7 Algorithme de Leon (modifié) 61
3.8 Algorithme DCP 62
3.9 Algorithme de Stern 65
3.10 Facteur de travail des algorithmes de décodage classiques pour cryptanalyser les systèmes à mots de poids faibles 66
3.11 Algorithme DCP utilisé pour la recherche de mots de poids minimal 68
3.12 Algorithme de Stern utilisé pour la recherche de mots de poids minimal 69

3.13 Algorithme de Stern, version duale 70
3.14 Facteur de travail des différentes versions optimisées de l'algorithme de décodage utilisant un code poinçonné (DCP) pour cryptanalyser les systèmes à mots de poids faibles 73
3.15 Facteur de travail des différentes versions optimisées de l'algorithme de Stern pour cryptanalyser les systèmes à mots de poids faibles 73

4.1 Coût relatif de l'élimination de Gauss dans le facteur de travail des algorithmes DCP et Stern 76
4.2 Algorithme DCP itératif pour le décodage (version GD) 79
4.3 Algorithme de Stern itératif pour la recherche de mots de poids minimal (version GM) 80
4.4 Résultats expérimentaux obtenus pour le décodage d'un code de longueur 256 et de dimension 128 avec l'algorithme de Stern itératif 93
4.5 Paramètres optimaux et facteur de travail de l'algorithme de Stern itératif pour décoder des codes binaires aléatoires de longueur n et de dimension $n/2$ 94
4.6 Paramètres optimaux et facteur de travail de l'algorithme de Stern itératif pour trouver un mot de poids minimal dans des codes binaires aléatoires de longueur n et de dimension $n/2$ . . . 96
4.7 Moyenne et écart-type du nombre d'itérations nécessaires à l'attaque des cryptosystèmes à mots de poids faible 99
4.8 Facteur de travail des différentes versions optimisées de l'algorithme DCP itératif pour cryptanalyser les systèmes à mots de poids faibles 100
4.9 Facteur de travail des différentes versions optimisées de l'algorithme de Stern itératif pour cryptanalyser les systèmes à mots de poids faibles 100
4.10 Distance de décodage accessible par l'algorithme de Stern itératif 104
4.11 Poids accessibles par l'algorithme de Stern itératif 105

5.1 Relations d'inclusion entre les codes BCH au sens strict et les codes de Reed-Muller poinçonnés en longueur 511 111
5.2 Distance minimale des codes BCH au sens strict de longueur 511 115

6.1 Borne sur la taille d'un tableau orthogonal binaire 145
6.2 Indice minimal d'un tableau orthogonal binaire 146
6.3 Indice minimal d'un tableau orthogonal sur un groupe abélien à 4 éléments 146
6.4 Indice minimal d'un tableau orthogonal sur un groupe abélien à 8 éléments 147
6.5 Ordre de résilience maximal d'une fonction sur $\mathbf{F}_2$ 147
6.6 Ordre de résilience maximal d'une fonction sur $\mathbf{F}_4$ 148
6.7 Ordre de résilience maximal d'une fonction sur $\mathbf{F}_8$ 148

Table des matières

Introduction **5**

1 Introduction à la cryptographie **7**
1 Le chiffrement 7
1.1 Le chiffrement à clef secrète 8
1.2 Le chiffrement à clef publique 9
2 L'authentification 10
2.1 La signature numérique 10
2.2 L'identification 11

I Attaque des cryptosystèmes à mots de poids faible **13**

Introduction **15**

2 Principaux cryptosystèmes à mots de poids faible **17**
1 Systèmes de chiffrement à mots de poids faible 18
1.1 Principe général du système de McEliece 18
1.2 Principe général du système de Niederreiter 22
1.3 Conditions sur la famille de codes secrets 28
1.4 Une famille de codes secrets particulière : les codes de Goppa irréductibles 29
1.5 Attaques induites par l'utilisation des codes de Goppa . . 32
2 Schémas d'identification à mots de poids faible 39
2.1 Schéma d'identification de Stern 40
2.2 Schéma d'identification de Véron 41
2.3 Attaque des schémas d'identification à mots de poids faible 43

3 Le problème général du décodage d'un code linéaire **47**
1 La NP-complétude du décodage complet 47
2 Notations 49
3 Algorithmes exhaustifs 50
3.1 Recherche exhaustive parmi les mots du code (REC) . . . 50
3.2 Recherche exhaustive parmi les vecteurs d'erreurs (REE) 50
4 Décodage par ensembles d'information (IS) 51

5 Généralisations du décodage par ensembles d'information 56
5.1 Algorithme de Lee-Brickell (LB) 56
5.2 Algorithme de Leon (LEON) 58
5.3 Algorithme de décodage utilisant un code poinçonné (DCP) 61
5.4 Algorithme de Stern (S) 63
5.5 Applications des algorithmes classiques de décodage à la cryptanalyse des systèmes à mots de poids faibles 66
6 Optimisation des algorithmes de décodage classiques 67
6.1 Décodage vs. recherche d'un mot de poids minimal 67
6.2 Approche duale 67
6.3 Optimisation des paramètres 71

4 Un algorithme de décodage itératif 75
1 Principe de l'algorithme itératif 76
2 Modélisation des algorithmes itératifs par un processus markovien 78
2.1 Description du modèle 78
2.2 Nombre moyen d'itérations des algorithmes itératifs . . . 86
2.3 Variance du nombre d'itérations des algorithmes itératifs 87
2.4 Distribution du nombre d'itérations des algorithmes itératifs 88
3 Facteur de travail des algorithmes itératifs 90
4 Résultats expérimentaux 92
5 Approximation du facteur de travail de l'algorithme de Stern itératif 94
5.1 Décodage d'un code aléatoire 94
5.2 Recherche de mots de poids minimal dans un code aléatoire 95
6 Attaque des cryptosystèmes à mots de poids faible 98
6.1 Facteur de travail de l'attaque des cryptosystèmes à mots de poids faible par les algorithmes itératifs 98
6.2 Attaques partielles des cryptosystèmes à mots de poids faible 102
6.3 Optimisation des paramètres du système de McEliece . . 104

5 Distance minimale des codes BCH de longueur 511 107
1 Définitions 108
2 Conditions suffisantes pour que la borne BCH soit atteinte 109
3 Autres méthodes 112
4 Applications des algorithmes itératifs 113

Conclusion sur la sécurité des systèmes à mots de poids faible 117

II Construction de fonctions résilientes sur un alphabet fini 119

Introduction 121

6 Caractérisations des fonctions résilientes sur un alphabet fini 123
1 Définitions . . . 124
2 Caractérisation en termes de tableaux orthogonaux . . . 125
3 Caractérisation en termes de transformée de Fourier . . . 126
3.1 Caractères d'un groupe abélien . . . 126
3.2 Transformation de Fourier . . . 128
3.3 Caractérisation des fonctions sans corrélation à l'aide de caractères . . . 129
4 Caractérisation en termes de matrices . . . 130
5 Construction de fonctions résilientes à partir de codes . . . 132
5.1 Codes et tableaux orthogonaux . . . 132
5.2 Codes et fonctions résilientes . . . 139
6 Bornes sur l'ordre de résilience d'une fonction . . . 140
6.1 Bornes classiques . . . 140
6.2 Borne de la programmation linéaire . . . 141
6.3 Bornes explicites dérivées de la borne de la programmation linéaire . . . 142
6.4 Tables donnant l'indice minimal d'un tableau orthogonal et l'ordre de résilience maximal d'une fonction . . . 145

7 Ordre de non-linéarité d'une fonction résiliente 149
1 Forme algébrique normale d'une fonction sans corrélation . . . 150
2 Combinaison de registres à décalage à rétroaction linéaire . . . 152
2.1 Registres à décalage à rétroaction linéaire et suites à récurrence linéaire . . . 152
2.2 Complexité linéaire d'un registre à décalage . . . 156
2.3 Combinaison de suites à récurrence linéaire . . . 161
3 Non-linéarité d'une fonction sans corrélation . . . 169
4 Construction de fonctions résilientes de non-linéarité optimale . . 172
5 Non-linéarité d'une fonction q-aire sans corrélation sur un sur-corps 176

8 Autres applications des fonctions sans corrélation 179
1 Générateurs aléatoires localement parfaits . . . 180
1.1 Définition et motivations cryptographiques . . . 180
1.2 Lien avec les fonctions sans corrélation . . . 181
2 Multipermutations . . . 182
2.1 Définition et motivations cryptographiques . . . 182
2.2 Exemple : FFT-Hashing . . . 184
2.3 Lien avec les fonctions sans corrélation . . . 189
2.4 Ordre de non-corrélation binaire d'une multipermutation 190

9 Construction de fonctions résilientes par composition 201
1 Construction par composition . . . 202
2 Exemples et corollaires . . . 203
3 Composition de fonctions linéaires et dual d'un code concaténé . 207

4 Application à la combinaison de LFSRs 209

Conclusion de l'étude sur les fonctions résilientes **211**

Perspectives **213**

A Quelques notions de théorie des codes **217**
1 Concepts de base 217
2 Codes linéaires 218
3 Groupe d'automorphismes - Codes cycliques 219
4 Construction de nouveaux codes à partir de codes connus 221