Sécurité de protocoles cryptographiques fondés sur les codes correcteurs d'erreurs

Léonard Dallot

UNIVERSITÉ de CAEN/BASSE-NORMANDIE

U.F.R. : Sciences

ÉCOLE DOCTORALE : SIMEM

THÈSE

présentée par

Léonard Dallot

et soutenue

le 15 juillet 2010

en vue de l'obtention du

DOCTORAT de l'UNIVERSITÉ de CAEN

spécialité : Informatique et applications

(Arrêté du 7 août 2006)

Sécurité de protocoles cryptographiques fondés sur les codes correcteurs d'erreurs

MEMBRES du JURY

Nicolas Sendrier	Directeur de recherche	INRIA Rocquencourt	(rapporteur)
German Saez	Professeur	Universitat Polytècnica de Catalunya	(rapporteur)
Claude Carlet	Professeur	Université Paris 8	
Marc Joye	Habilité à diriger des recherches	Technicolor France	
Philippe Gaborit	Professeur	Université de Limoges	
Javier Herranz	Assistant professor	Universitat Polytècnica de Catalunya	
Damien Vergnaud	Maître de conférences	Ecole Normale Supérieure	(invité)
Brigitte Vallée	Directrice de recherche	CNRS, Université de Caen	(directrice)
Ayoub Otmani	Maître de conférences	Université de Caen	(co-encadrant)

Mis en page avec la classe thloria.

Remerciements

Me voici à quelques jours de soutenir et de nouveau devant une page blanche... Je repense à tous ceux qui ont permis à cette thèse d'exister ; vous êtes nombreux !

Mes premiers remerciements vont tout d'abord à Ayoub Otmani pour avoir accepté de m'encadrer. Tes conseils, tes questions et ta rigueur de rédaction m'ont été précieux tout au long de cette thèse. Ce chemin parcouru ensemble n'aurait pu exister sans Brigitte Vallée, qui a accepté de me co-encadrer. Merci d'avoir toujours été là quand il le fallait.

Un grand merci également à Fabien Laguillaumie qui a toujours trouvé du temps pour m'expliquer un détail que j'avais du mal à comprendre, me relire, discuter d'une idée ou simplement aller faire un tour à la Garsouille (parfois pour parler crypto). Je voudrais remercier tout aussi chaleureusement Damien Vergaud, un collaborateur de choix, souvent disponible, plein d'idées et toujours prêt à les partager.

Je voudrais exprimer ma profonde gratitude à Nicolas Sendrier, pour l'attention qu'il a porté à mon travail, en particulier en me faisant l'honneur de rapporter ce mémoire ; ainsi qu'à German Saez pour avoir accepté au pied levé la tâche délicate de rapporter ce mémoire sans connaître mon travail, dans un délai court qui plus est.

Merci également à Claude Carlet, Philippe Gaborit, Javier Herranz et Marc Joye d'avoir accepté de participer à mon jury, surtout à une date aussi atypique.

Je souhaite également remercier mes co-auteurs, sans qui cette thèse n'aurait pu voir le jour : Ayoub, Damien et Jean-Pierre Tillich. Ce fût un réel plaisir de travailler avec vous.

Je voudrais adresser mes remerciements à tous ceux que j'ai eu la chance de côtoyer aux journées C2, aux écoles de jeunes chercheurs du GdR IM ou en conférences, sans qui ce n'aurait pas été pareil : Maria, Andréa, Pierre-Louis, Nadia (par la suite devenue collègue), Damien, Guilhem, Benoît, Patrick, Romain, Ludovic, Hélène, et tant d'autres... Merci également à toute l'équipe SECRET de l'INRIA pour leur accueil lors de mes visites.

Pour conclure le volet scientifique de mes remerciements, je voudrais remercier aussi tous mes collègues du GREYC, de l'ENSI et de l'IUT de Saint-Lô pour les nombreuses discussions que nous avons eues, devant la machine à café, au coin d'un bureau, au R.U, ou ailleurs. Merci en particulier à Cyril, Florent, Céline (les deux), Romain (et son jeu de Wagic, ainsi que son humour — noir, gras ou décalé), Hugo, Matthieu, Boris, Matthieu (pas le même), Jean-Hugues, Sébastien, Guillaume, Loïc, Ali, Julien, Jean-Marie et tous ceux que j'ai oublié de citer. J'adresse un merci tout particulier à Véronique pour m'avoir accueilli dans son bureau — un peu toxique — qui est devenu le notre. Je souhaiterais aussi remercier tous ceux qui nous permettent de faire notre travail en toute sérénité : Agnès, Antony, Hélène, Davy, Véronique, Virginie, pour ne citer que vous, vos compétences m'ont été précieuses.

Et puisqu'il y a une vie en dehors du labo, je voudrais remercier tous ceux qui ont rendu ces années si agréables – au point que je ne les ai (presque) pas vues passer. Ma famille tout d'abord : mon père, ma mère, mes frères Constantin et Barnabé, sans oublier Victorine, ma (belle-)sœur.

Merci de m'avoir supporté (dans tous les sens du terme) toutes ces années. Merci Grand-Mère, pour tes colis de cocas, tes encouragements, et surtout pour la relecture de ce mémoire qui a du te sembler écrit dans une bien drôle de langue.

Un grand merci également à mes compagnons de soirées caennaises : Alexis, Aurélien, Céline, Élodie, Pierrick, Teddy, Lydie, Michaël, ... Sans oublier Fabien (encore) et Iwen, parce qu'il est possible de faire la fête sans se retenir et finir par parler crypto à trois heures du matin !

Je voudrais adresser un merci tout spécial à Élodie, Teddy et Pierrick pour (entre autres) une soirée *Golden Potes* mémorable !

Je voudrais de nouveau adresser mes remerciements à Fabien pour sa connaissance sans limite du reggae, pour m'avoir toujours motivé à aller voir les concerts qu'il fallait. Merci, également de nouveau, à Teddy et Pierrick (ainsi que Beb et Maïté) pour m'avoir suivi à tous ces concerts ou ces sound systems, même ceux qu'il ne fallait pas... J'en profite pour remercier également — ils l'ignoreront sûrement — les équipes du BBC et du Cargö, ainsi que Dub Livity, pour les nombreuses soirées musicales qu'ils m'ont permis de vivre.

Merci aussi à mes amis d'enfance, David, Ludo et Corentin pour leur attachement sans faille et leur soutien inconditionnel, même à distance et quand les visites s'espaçaient faute de temps.

Enfin, je voudrais adresser le plus grand des mercis à celle qui partage ma vie depuis maintenant un an et demi, Aurélie. Pour ton soutien sans faille, pour tes nombreuses lectures, ta patience (parfois) et tout simplement ces nombreux moments où tu me rends heureux, merci Petit Chat !

Mais j'allais oublier : un grand merci à vous tous, qui avez pris la peine d'ouvrir ce mémoire et de lire ces remerciements. Merci à tous ceux qui sont venu m'écouter ! Et, même si probablement peu d'entre vous s'aventureront à lire la suite, bonne lecture !

Table des matières

Notations

$\mathbb{N}$	Ensemble des entiers naturels
$\mathbb{N}^*$	Ensemble des entiers naturels privés de zéro
$\mathbb{R}$	Ensemble des nombres réels
$\mathbb{R}_+$	Ensemble des nombres réels positifs
$\mathbb{F}_q$	Corps fini à q éléments
$\mathcal{I}_k$	Matrice identité de dimension k
$X \backslash Y$	Ensemble X privé de ensemble Y
$\lvert X \rvert$	Cardinal de l'ensemble X
$\lvert x \rvert$	Taille (en bits) d'une suite de bits x
$\lfloor x \rfloor, \lceil x \rceil$	Partie entière inférieure de x, partie entière supérieure de x
$\{0,1\}^*$	Ensemble des mots binaires de longueur quelconque
$\{0,1\}^k$	Ensemble des mots binaires de longueur k
1^κ	Suite de κ bits de valeur 1
$a \Vert b$	Concaténation des suites de bits représentant a et b
$a \oplus b$	Ou exclusif bit à bit entre les suites de bits représentant a et b
$\mathcal{C}^\perp$	Code dual du code $\mathcal{C}$
$supp(\mathbf{x})$	Support du vecteur $\mathbf{x}$
$wt(\mathbf{x})$	Poids de Hamming du vecteur $\mathbf{x}$
$dist(\mathbf{x}, \mathbf{y})$	Distance de Hamming entre les vecteurs $\mathbf{x}$ et $\mathbf{y}$
$\mathcal{B}_{n,k}$	Ensemble des codes bianaires de longeur n et de dimension k
$\mathcal{W}_{n,t}$	Ensemble des mots de poids t de $\mathbb{F}_2^n$
$\mathcal{G}_{m,t}$	Ensemble des codes de Goppa binaires de longueur $n = 2^m$ ayant un polynôme générateur de degré t
$\mathbb{F}_2[x]$	Ensemble des polynômes univariés à coefficients dans $\mathbb{F}_2$

$\mathbf{u}(x) \star \mathbf{v}(x)$	Polynôme d'intersection entre $\mathbf{u}(x)$ et $\mathbf{v}(x)$
$\mathsf{PTM}(n)$	Machine de Turing fonctionnant en temps polynomial en la taille n des entrées
$\mathsf{PPTM}(n)$	Machine de Turing probabiliste fonctionnant en temps polynomial en la taille n des entrées
$a \leftarrow b$	La valeur de b est affectée à la variable a
$a \xleftarrow{R} X$	Affectation à a d'un élément de X tiré aléatoirement selon la distribution uniforme
$a \leftarrow \mathcal{A}(\cdots)$	La sortie de l'algorithme $\mathcal{A}$ est affectée à a
$\langle \mathcal{P}; \mathcal{V} \rangle$(x, y)	Sortie du protocole interactif entre $\mathcal{P}$ et $\mathcal{V}$ ayant x et y comme entrées communes

Introduction

La cryptologie est une science ancienne dont le nom signifie « science du secret ». Elle a pour objet la conception et l'analyse des méthodes permettant de masquer de l'information. Il est courant de la séparer en deux branches distinctes mais intimement liées : la cryptographie qui propose les solutions visant à assurer le secret, et la cryptanalyse qui s'attache à dévoiler les faiblesses des ces systèmes.

Apparues durant l'antiquité, les premières techniques permettant la transmission secrète d'un message appartiennent à ce qu'on appelle aujourd'hui la stéganographie. Il s'agit ici uniquement de dissimuler l'information pour éviter qu'elle ne soit identifiée comme telle. La cryptographie, qui ne concernait à l'origine que le chiffrement de messages (c'est-à-dire leur réécriture dans un langage ou un alphabet connu des seuls protagonistes de l'échange), adopte une approche différente : l'objectif n'est pas d'empêcher un éventuel espion de découvrir le message mais de le modifier de telle sorte que celui-ci soit inintelligible pour tout autre que son destinataire légitime et éventuellement son expéditeur. Les premiers procédés de chiffrement datent également de l'antiquité ; les exemples les plus célèbres sont la scytale et le chiffrement de César. Il est courant de comparer le chiffrement d'un message à l'utilisation d'un coffre muni d'une serrure. La transmission d'un message consiste à enfermer celui-ci dans le coffre puis à le refermer à l'aide d'une clef ; le destinataire du message, en possession d'une copie de la clef, peut ouvrir le coffre et récupérer le message qu'il contient. Bien sûr, s'il est impossible à un éventuel espion de récupérer la clef, il lui est toujours possible de forcer la serrure pour obtenir le message. La cryptographie à clef secrète reproduit ce schéma : le mécanisme de chiffrement (notre coffre) est paramétré par une information secrète (la clef) ; le destinataire doit être en possession de la même information pour retrouver le message. Trouver les faiblesses de la serrure est l'objectif des cryptanalystes. Puisque une même clef (gardée secrète par les deux protagonistes) est nécessaire aussi bien pour le chiffrement et le déchiffrement, un schéma obéissant à ce principe appartient à ce qu'on nomme aujourd'hui la cryptographie symétrique ou cryptographie à clef secrète.

Si au cours des siècles la cryptographie a passionné bon nombre de scientifiques (on pourra à ce sujet consulter l'ouvrage de S. Singh *Histoire des codes secrets* [Sin01]), c'est durant la Seconde Guerre mondiale qu'elle gagna ses lettres de noblesse. En effet, les efforts de cryptanalyse

des Alliés contre la machine Enigma leur permirent de repérer plusieurs attaques et de découvrir certaines positions stratégiques de l'Axe. Parmi les avancées les plus notables de cette période se trouvent les « Bombes de Turing » puis les « machines de Newman », des machines à « casser les codes » à l'origine des ordinateurs modernes.

Aussi efficace soit-elle, la cryptographie symétrique souffre d'un inconvénient majeur : la nécessité de partager une clef commune. En effet, avant tout échange chiffré, il est nécessaire de s'accorder sur la clef par le biais d'un canal sûr (une rencontre physique, par exemple). Le problème est encore accentué lors d'une communication simultanée avec plusieurs protagonistes. Une solution consiste à utiliser une autorité de confiance, laquelle est chargée de communiquer aux différentes parties les clefs nécessaires à leurs communications par le biais de canaux sûrs préalablement établis. Mais là encore, se pose le problème de l'établissement de ces canaux, nécessitant toujours un partage de clefs initial entre l'autorité et ses membres.

En 1976, W. Diffie et M. Hellman publièrent l'article fondateur *New Directions in Cryptography* [DH76] dans lequel ils proposèrent une solution simple et élégante au partage de secret. Ils y introduisirent l'idée qui marque le début de la cryptographie moderne : la cryptographie à clef publique ou cryptographie asymétrique. Dans cette approche, si Alexiane souhaite recevoir des messages chiffrés de Barnabé et Constantin, il lui suffit de créer une paire de clefs fortement liées entre elles. L'une d'elles est alors rendue publique, par exemple sur internet, tandis que l'autre est gardée secrète par Alexiane. Constantin et Barnabé peuvent alors récupérer la clef publiée et l'utiliser pour chiffrer leurs messages à destination d'Alexiane, laquelle utilise la clef qu'elle a gardée secrète pour les déchiffrer. Si la clef publique ne permet pas de retrouver la clef privée, la sécurité de leurs communications est alors garantie. Tout usager du réseau peut donc placer sa clef de chiffrement dans un répertoire public. Ceci permet à tout utilisateur du cryptosystème d'envoyer à tout autre utilisateur des messages chiffrés, de façon à ce que seul le destinataire légitime soit à même de le déchiffrer. Une communication confidentielle est donc permise entre deux personnes sans nécessiter de communication préalable. L'analogie avec un coffre permet à nouveau d'illustrer le concept de cryptographie à clef publique. Cette fois, les coffres renfermant les messages ne sont plus munis de serrures mais sont fermés à l'aide de cadenas. Alexiane peut disposer, ouverts, des cadenas pour lesquels elle dispose des clefs, dans un emplacement convenu et librement accessible. Constantin et Barnabé n'ont alors plus qu'à récupérer l'un de ces cadenas pour enfermer leurs messages dans un coffre que seule Alexiane peut ouvrir.

S'ils proposèrent le concept de cryptographie à clef publique, W. Diffie et M. Hellman ne donnèrent dans leur article aucune réalisation d'un tel cryptosystème et c'est deux ans plus tard que A. Rivest, A. Shamir et L. Adelman proposèrent le premier cryptosystème à clef publique, fondé sur l'arithmétique modulaire, qui est aujourd'hui connu sous le nom de RSA [RSA78]. La même année, R. J. McEliece proposa un autre cryptosystème à clef publique

[McE78] à l'aide de la théorie des codes correcteurs d'erreurs.

La théorie des codes correcteurs d'erreurs est née à la fin des années 40 sous l'impulsion de R. Hamming, qui travaillait alors sur un modèle de calculateurs de faible fiabilité. Si, durant la semaine, les erreurs apparaissant durant le traitement pouvaient être corrigées manuellement, cela était impossible durant les périodes chômées. Pour pallier cette situation, R. Hamming introduisit le premier code correcteur d'erreurs. A la même période, C. Shannon donnait naissance à la théorie de l'information qu'il formalisa comme une branche des mathématiques. Hamming développa les prémices de la théorie des codes, l'inscrivant dans la théorie de l'information, et décrivit sa solution comme exemple. Si la cryptographie vise à protéger les données contre la malveillance, la théorie des codes correcteurs d'erreurs tente de les rendre insensibles aux erreurs apparaissant naturellement sur les canaux de communication. La stratégie qu'elle emploie consiste à transmettre plus de données que l'information en nécessite ; ces données supplémentaires constituent donc une redondance de l'information. Si celle-ci est correctement structurée, elle permet alors de détecter, voire de corriger, d'éventuelles erreurs introduites par le canal.

Depuis leur introduction, les codes correcteurs d'erreurs constituent un domaine de recherche actif et de nombreuses familles de codes ont été introduites par la suite. Ces recherches ont alors conduit à l'émergence de nouveaux problèmes réputés difficiles, comme le décodage à distance bornée dans les codes aléatoires. C'est sur celui-ci que s'appuie en grande partie la construction de R. J. McEliece. Les codes correcteurs d'erreurs étant destinés à fiabiliser les communications, les calculs réalisés sont par nature extrêmement rapides ; cette propriété se retrouve dans les cryptosystèmes fondés sur la théorie des codes. Si le chiffrement asymétrique introduit une grande souplesse d'utilisation, les techniques fondées sur l'arithmétique modulaire sont considérablement plus lentes que celles utilisées dans les systèmes de chiffrement symétriques, tandis que les cryptosystèmes fondés sur les codes correcteurs d'erreurs permettent d'approcher les vitesses de chiffrement des algorithmes symétriques. Cependant la grande taille des clefs nécessaires au chiffrement et au déchiffrement dans les schémas utilisant la théorie de codes correcteurs d'erreurs et la nécessité d'ajouter une redondance à la donnée chiffrée ont fait que les techniques fondées sur l'arithmétique sont actuellement, et de loin, les plus utilisées.

Cette hégémonie pose un sérieux problème : la sécurité de la quasi-totalité des applications cryptographiques actuelles repose sur deux problèmes supposés difficiles de l'arithmétique modulaire, à savoir le problème de la factorisation et le problème du logarithme discret. Toutes ces applications sont donc vulnérables à une unique percée théorique ou matérielle (un éventuel ordinateur quantique, par exemple, serait à même de venir à bout de l'ensemble de ces cryptosystèmes). La diversité est une manière de prévenir ce risque et il est donc du devoir de la communauté cryptographique de proposer des alternatives aux systèmes fondés sur l'arithmétique modulaire. Actuellement, les pistes les plus sérieuses pour se préparer à ce qu'on appelle la cryptographie post-quantique sont la cryptographie fondée sur les réseaux euclidiens et la cryp-

tographie fondée sur les codes correcteurs d'erreurs, qui fait l'objet de ce mémoire.

Longtemps, l'évaluation des techniques cryptographiques est restée essentiellement empirique : les cryptographes proposaient un schéma et les cryptanalystes tentaient d'en déceler les faiblesses ; la sécurité du schéma restait acquise tant qu'aucune d'entre elles n'était révélées. Cependant, l'apparition des cryptosystèmes à clef publique a permis par la suite l'émergence de nouvelles techniques à l'origine de nouvelles primitives cryptographiques telles que la signature électronique ou l'identification. En effet, si historiquement la cryptographie s'est intéressée en premier lieu à la transmission confidentielle de messages, les objectifs de la cryptographie se sont diversifiés avec la démocratisation de l'ordinateur domestique et la multiplication des échanges informatiques. On peut citer notamment :

- l'identification, au cours de laquelle un utilisateur prouve son droit d'accéder à certaines informations ou son identité ;
- l'authentification, qui garantit qu'un message provient bien de la source annoncée ;
- l'intégrité des données, qui permet de s'assurer qu'un message n'a pas été modifié lors de sa transmission ;
- la non-répudiation, qui garantit qu'une personne ne peut nier avoir commis une action ;
- la protection de l'anonymat, qui permet de réaliser certaines actions sensibles sans qu'il soit possible de remonter à sa source.

Dans ces conditions, une évaluation empirique des constructions cryptographiques a rapidement atteint ses limites. S. Goldwasser et S. Micali proposèrent en 1984 une nouvelle approche de l'évaluation de ces cryptosystèmes : la sécurité prouvée ou sécurité réductionniste. Dans ce modèle, les propriétés de sécurité du cryptosystème sont formalisées au sein d'un modèle de confrontation entre un adversaire abstrait contre la construction analysée et un challenger exécutant les algorithmes étudiés, avec l'hypothèse claire que l'adversaire a accès au système aussi bien qu'à des ressources de calcul. Il est ensuite montré que cet adversaire peut être utilisé pour résoudre efficacement un ou plusieurs problèmes algorithmiques réputés difficiles (la factorisation ou le problème du décodage borné, par exemple). Si ces problèmes sont difficiles à résoudre, un tel adversaire ne peut alors clairement pas exister ; le schéma est donc sûr (sous réserve que le modèle considéré soit adapté) tant que les problèmes algorithmiques sous-jacents restent difficiles.

La sécurité réductionniste constitue aujourd'hui un pan important de la recherche en cryptographie, curieusement peu répandu au sein de la communauté de la cryptographie fondée sur les codes correcteurs d'erreurs. C'est donc l'étude des techniques de sécurité réductionniste dans le cadre de la cryptographie fondée sur les codes correcteurs d'erreurs qui fait l'objet de ce mémoire. Celui-ci est structuré en trois parties et sept chapitres, les trois derniers étant consacrés aux résultats obtenus.

La première partie, composée de trois chapitres, forme une introduction générale à la cryp-

tographie et aux codes correcteurs d'erreurs. Les deux premiers chapitres introduisent la cryptographie, dans sa forme générale. Le premier présente les outils mathématiques et informatiques nécessaires à la suite, ainsi que les deux principales primitives de la cryptographie asymétrique : le chiffrement et la signature. Le second est consacré à la sécurité réductionniste. Il y est décrit comment établir celle-ci, le lien avec la sécurité d'un cryptosystème lors d'un usage concret et les modèles de confrontation permettant de prouver la sécurité des schémas de chiffrement et de signature. Le dernier chapitre de cette partie présente les bases de la théorie des codes correcteurs d'erreurs et s'intéresse aux problèmes de décodage dans un code aléatoire.

La partie suivante, comprenant deux chapitres, est consacrée aux schémas de chiffrement utilisant la théorie des codes correcteurs d'erreurs. Le quatrième chapitre est consacré aux principaux schémas de chiffrement fondés sur les codes correcteurs d'erreurs, le schéma de McEliece [McE78], bien sûr, et deux variantes proposées l'une par H. Niederreiter [Nie86] et l'autre par B. Biswas et N. Sendrier [BS08]. La description de ces schémas nécessite de s'intéresser en premier lieu à une famille de codes particulière, celle des codes de Goppa. Une analyse rigoureuse de la sécurité réductionniste de ces trois schémas est proposée. Ces analyses nous conduisent en particulier à démontrer que les problèmes CGPBD (pour *Computational Goppa Paramatrized Bounded Decoding*) et CGPSD (pour *Computational Goppa Paramatrized Syndrome Decoding*), souvent considérés comme étant à la base de la sécurité de ces cryptosystèmes, ne sont pas nécessairement les plus pertinents. Nous introduisons donc deux nouveaux problèmes que nous nommons respectivement RCGPBD (pour *Restricted Computational Goppa Paramatrized Bounded Decoding*) et RCGPSD (pour *Restricted Computational Goppa Paramatrized Syndrome Decoding*).

Le cinquième chapitre de ce mémoire présente les cryptanalyses de deux variantes du cryptosystème de McEliece visant à réduire la taille des clefs, nous permettant ainsi d'illustrer le besoin de fournir une preuve rigoureuse de la sécurité des cryptosystèmes proposés. Ces cryptanalyses sont le résultat de travaux réalisés en collaboration avec A. Otmani et J.P. Tillich, dont l'un a été tout d'abord publié à la conférence SCC 2008 [OTD08a] puis, conjointement lors d'une édition spéciale du journal *Mathematics in Computer Science* consacrée à la programmation symbolique et à la cryptographie [OTD08b].

Les deux chapitres suivants forment la dernière partie de ce mémoire, consacrée aux techniques de signature fondées sur les codes correcteurs d'erreurs. Le sixième chapitre présente une preuve réductionniste de la sécurité d'un schéma de signature proposé en 2001 par N. Courtois, M. Finiasz et N. Sendrier, sous réserve d'y introduire une légère modification. Ce résultat a fait l'objet de la publication [Dal08]. Enfin, nous démontrons dans un septième et dernier chapitre que les techniques utilisées dans le schéma de N. Courtois, M. Finiasz et N. Sendrier peuvent être utilisées pour construire un nouveau schéma de signature de cercle à seuil, permettant à un ensemble d'utilisateurs de signer conjointement un document tout en

préservant leur anonymat parmi un ensemble fixé d'utilisateurs. Bien entendu, nous fournissons une preuve de sécurité de ce nouveau schéma, lequel a été publié en collaboration avec D. Vergnaud lors de IMACC 2009 [DV09].

Première partie

Préliminaires

Chapitre 1

Eléments de cryptographie

Sommaire

1.1 Cadre théorique de la cryptographie à clef publique

Avant de pouvoir présenter les principaux éléments de la cryptographie à clef publique, il est nécessaire de présenter les outils permettant leur existence. Nous présentons tout d'abord le modèle théorique à l'origine de la conception des ordinateurs et nous servant aujourd'hui de modèle de référence pour la conception et l'analyse de protocoles cryptographiques. Ensuite, nous définissons les fonctions mathématiques permettant la construction des primitives cryptographiques présentées à la section suivante.

1.1.1 Modele théorique des ordinateurs – Machines de Turing et complexité

Les machines de Turing sont des machines théoriques à l'origine de la conception des ordinateurs actuels. Comme leur nom l'indique, elles ont été introduites par A. Turing en 1937 [Tur37]. La thèse de Church assure que tout calcul et donc tout algorithme peuvent être simulés par des machine de Turing. La notion de machine de Turing permet de manipuler formellement les problèmes de calculabilité et de traitement de l'information ainsi que la notion de complexité. Ces machines permettent donc de formaliser de façon abstraite la difficulté d'un problème et constituent le modèle de calcul le plus couramment utilisé en cryptologie.

Une *machine de Turing* est composée de cinq éléments :

- un *ruban infini*, possédant une extrémité gauche mais pas d'extrémité droite, divisé en cases de même taille ;
- une *tête de lecture/écriture* qui se déplace sur le ruban. A chaque impulsion, la tête peut soit rester sur place, soit reculer (si possible) ou avancer d'une case. La tête a le droit d'écrire dans la case qu'elle visite ;
- un *ensemble fini de symboles* qui seront inscrits et lus sur la bande par la tête de lecture. Un ensemble typique de symboles est $\{0, 1, s, d, f\}$ où 0 et 1 correspondent à la numérotation binaire, s correspond à un séparateur, d marque le début de la bande et f indique la fin d'un programme sur la bande ;
- un *ensemble fini d'états* qui permettent de différencier les différentes situations possibles. On peut notamment distinguer deux états particuliers : l'état de départ D, fourni par les données initiales, et l'état final F correspondant à la fin de l'exécution du programme et dont le résultat peut être observé ;
- *un ensemble fini d'instructions* : en fonction du symbole c lu par la tête et de l'état courant E, la tête écrit un nouveau symbole c' puis effectue un déplacement à gauche (déplacement noté -1), un déplacement à droite (noté $+1$) ou reste en place (déplacement noté 0). La machine se trouve alors dans un nouvel état E'.

Celle-ci peut être définie formellement de la façon suivante :

Définition 1.1.1 (Machine de Turing) *Une* machine de Turing *est définie par le* 7*-uplet* $(\Sigma, \mathcal{Q}, \sigma, \delta, \Delta, q_0, \mathcal{F})$ *où*

- Σ *est un alphabet fini,*
- $\mathcal{Q}$ *est l'ensemble (fini) des états,*
- $\sigma : \mathcal{Q} \times \Sigma \to \Sigma$ *est une fonction d'écriture,*
- $\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q}$ *est une fonction de changement d'état,*
- $\Delta : \mathcal{Q} \times \Sigma \to \{G, D, \perp\}$ *est une fonction de déplacement,*
- $D \in \mathcal{Q}$ *est l'état initial et*
- $\mathcal{F} \subset \mathcal{Q}$ *est un ensemble d'états finaux.*

Complexité

A une machine de Turing donnée M, il est possible d'associer différentes complexités, relativement une entrée $\mathcal{I}$. Nous nous intéresserons ici à la *complexité en temps*, notée $T_M(\mathcal{I})$. Elle est donnée par le nombre d'états nécessaire pour passer de l'état initial à l'état final.

La complexité d'un calcul est dite *polynomiale* si la complexité en temps de la machine de Turing effectuant ce calcul peut être majorée par un polynôme en la taille n des données initiales. On parle alors de *machine de Turing fonctionnant en temps polynomial* (en anglais, *Polynomial time Turing Machine*) et on la note $\mathsf{PTM}(n)$. Cette complexité est définie formellement de la façon suivante :

Définition 1.1.2 (Machine de Turing fonctionnant en temps polynomial) *Soient une machine de Turing* M *et* $T_M(n) \stackrel{def}{=} \sup\{T_M(\mathcal{I}),\ |\mathcal{I}| = n\}$.

La complexité en temps de M *est dite* polynomiale *si*

$$\exists n_0, \exists c \in \mathbb{N}\backslash\{0\}, \forall n \geq n_0, T_M(n) \leq n^c$$

M *est alors appelée une* machine de Turing fonctionnant en temps polynomial.

Le fonctionnement des machines de Turing peut être modifié de diverses façons afin d'en augmenter ou d'en restreindre les capacités. Nous en présentons ici trois variantes.

Machines de Turing probabilistes La définition 1.1.1 définit une machine de Turing comme entièrement déterministe : elle est incapable de faire des choix lors de son exécution et deux entrées identiques produiront toujours la même sortie. Il est possible de briser ce déterminisme en lui adjoignant une source de caractères aléatoires. On appelle ces machines des *machines de Turing probabilistes*.

Plus concrètement, on ajoute à la définition des machines de Turing une *bande aléatoire* ω infinie à droite sur laquelle est stockée une suite infinie s de caractères choisis aléatoirement selon la distribution uniforme dans l'alphabet Σ. Afin de distinguer une telle machine d'une

machine de Turing déterministe, nous noterons une telle machine $M^\omega = (\Sigma, \mathcal{Q}, \sigma, \delta, \Delta, q_0, \mathcal{F}, s)$. La complexité en temps d'une machine de Turing probabiliste peut différer selon la bande aléatoire qui lui est fournie, nous désignerons alors la complexité en temps de la machine M^ω relativement à une entrée $\mathcal{I}$ et la suite s par la quantité $T_{M^\omega,s}(\mathcal{I})$. On pose alors $T_{M^\omega,s}(n) = \sup\{T_{M^\omega,s}(\mathcal{I}), |\mathcal{I}| = n\}$.

Ceci nous permet de définir la complexité en temps relativement à une entrée de taille n d'une machine de Turing probabiliste qui s'arrête (c'est-à-dire dont le nombre de pas élémentaires est borné) pour toute entrée $\mathcal{I}$ de taille n et toute suite s écrite sur ω. Notons $k(n)$ le nombre maximum de caractères aléatoires utilisés pour une entrée de taille n et $S_{k(n)}$ l'ensemble des suites de $k(n)$ caractères uniformément distribués sur $\Sigma^{k(n)}$. La complexité en temps de M^ω relativement à une entrée de taille n , notée $T_{M^\omega}(n)$ est définie par

$$T_{M^\omega}(n) = \frac{1}{|S_{k(n)}|} \sum_{s \in S_{k(n)}} T_{M^\omega,s}(n).$$

Une machine de Turing probabiliste de complexité en temps polynomial est appelée une *machine de Turing probabiliste polynomiale* et est notée PPTM (de l'anglais *Probabilistic Polynomial time Turing Machine*)

Machines de Turing interactives Les définitions des machines de Turing, probabilistes ou non, ne prennent pas en compte la possibilité pour deux machines (ou plus) de communiquer. Les *machines de Turing (resp. probabilistes) interactives* permettent de pallier ce manque : il s'agit de machines de Turing (resp. probabilistes) pourvues d'un ruban de communication en lecture/écriture.

Machines de Turing à oracles Le dernier type de machines de Turing que nous verrons forme les *machines de Turing à oracles*. Ces machines peuvent interroger des "oracles" capables de répondre à une question d'un type donné pour un coût constant. Si une telle machine M a accès à un oracle $\mathcal{O}$, nous la noterons $M^{\mathcal{O}}$. Une machine de Turing à oracle disposant d'un ruban aléatoire est appelée une *machine de Turing probabiliste à oracle*.

Machines de Turing et algorithmes

Les variantes des machines de Turing permettent de formaliser les notions d'algorithme et d'efficacité. Pour exécuter un algorithme sur une machine de Turing, on inscrit ses entrées sur le ruban, éventuellement on lui adjoint un ruban aléatoire ou ses oracles, puis la machine est placée dans l'état initial D. Si la machine s'arrête, la sortie est alors inscrite sur le ruban. Les machines de Turing permettent ainsi de représenter les algorithmes sous la forme de machines abstraites.

Dans la suite de ce mémoire,

- un *algorithme déterministe* sera représenté par une PTM,
- un *algorithme probabiliste* sera représenté par une PPTM,
- un *algorithme efficace* sera représenté par une machine de Turing interactive (éventuellement probabiliste) fonctionnant en temps polynomial.
- les *adversaires* (aussi appelés attaquants) seront représentés par des PPTM, éventuellement à oracles, fonctionnant en temps polynomial.

En plus de la notion d'algorithme, la définition des machines de Turing interactives permet de formaliser la notion de *protocole interactif* :

Définition 1.1.3 (Protocole interactif) *Un* Protocole interactif *est l'ensemble de deux machines de Turing interactives $\mathcal{P}$ et $\mathcal{V}$ appelées respectivement prouveur et vérificateur partageant un même ruban de communication qui leur permet d'échanger des données. On note un tel protocole $(\mathcal{P}, \mathcal{V})$ et on désigne sa sortie par $\langle \mathcal{P}; \mathcal{V} \rangle$.*

1.1.2 Outils mathématiques

Fonctions à sens unique

Les fonctions à sens unique sont au cœur de la réalisation de protocoles cryptographiques à clef publique. Informellement, il s'agit de fonctions facilement calculables, mais difficilement inversibles. La formalisation des fonctions facilement calculables repose sur la notion d'algorithme efficace, mais aussi sur la notion de fonction négligeable. Les fonctions négligeables servent aussi, comme nous le verrons par la suite, à quantifier la sécurité des cryptosystèmes.

Définition 1.1.4 (Fonction négligeable) *Une fonction $v : \mathbb{N} \to \mathbb{R}_+$ est dite* négligeable *si*

$$\forall c \in \mathbb{N}\backslash\{0\}, \exists k_c \in \mathbb{N}, \forall k \geq k_c, v(k) < k^{-c}$$

Les fonctions négligeables permettent de définir les notions de probabilités négligeables et écrasantes :

Définition 1.1.5 (Probabilité négligeable, probabilité écrasante) *Une probabilité p dépendant d'un paramètre (dit* paramètre de sécurité*) κ est dite* négligeable *si p est une fonction négligeable de κ.*

Une probabilité p dépendant d'un paramètre de sécurité κ est dite écrasante *si la probabilité $1 - p$ est négligeable.*

Définition 1.1.6 (Fonction à sens unique) *Une fonction $f : \mathcal{D} \to \mathcal{D}'$ est dite* à sens unique *si :*

- *il existe un algorithme efficace qui prenant $x \in \mathcal{D}$ en entrée renvoie $f(x)$,*

– *pour tout algorithme efficace $\mathcal{A}$, pour tout k suffisamment grand, la probabilité*

$$\Pr[x \overset{R}{\leftarrow} \mathcal{D}; y \leftarrow f(x); x' \leftarrow \mathcal{A}(1^k, y) : f(x') = f(x)]$$

est négligeable.

Exemple Afin d'illustrer ce concept, nous présentons ici un exemple de fonction à sens unique issu de la théorie des codes correcteurs d'erreurs. Nous en donnons un certain nombre d'intuitions sans toutefois justifier des éléments provenant de la théorie des codes. Ces aspects font l'objet du chapitre 3.

Soit H une matrice binaire *aléatoire* de taille $(n-k) \times n$ et considérons la fonction f qui associe à un élément x de $\mathcal{D} = \mathbb{F}_2^n$ l'élément Hx^T de $\mathcal{D}' \subset \mathbb{F}_2^{n-k}$. Le calcul de Hx^T peut être réalisé en faisant au plus n additions de colonnes de $n-k$ bits, soit $n(n-k) = O(n^2)$ additions binaires, ce qui est réalisé en temps polynomial.

En revanche, nous verrons que la théorie des codes correcteurs d'erreurs nous garantit qu'étant donné un vecteur $y = Hx^T \in \mathcal{D}' \subset \mathbb{F}_2^{n-k}$ et la matrice H, il est *difficile* de retrouver un vecteur $x' \in \mathcal{D} = \mathbb{F}_2^n$ tel que $Hx'^T = Hx^T$, c'est-à-dire que, pour tout algorithme efficace $\mathcal{A}$ et pour toute matrice H suffisamment grande, la probabilité $\Pr[x \overset{R}{\leftarrow} \mathbb{F}_2^n; y \leftarrow Hx^T; x' \leftarrow \mathcal{A}(y) : Hx'^T = Hx^T]$ est négligeable. Notre fonction $f : \mathcal{D} = \mathbb{F}_2^n \to \mathcal{D}' \subset \mathbb{F}_2^{n-k}, x \mapsto Hx^T$ est donc une fonction à sens unique.

Fonctions de hachage

Une fonction de hachage est une fonction qui transforme un élément d'un ensemble de grande taille, voire même infini, en un élément d'un domaine de taille plus petite. Initialement utilisées en informatique pour condenser une donnée — idéalement uniformément répartie dans l'ensemble d'arrivée — les fonctions de hachage sont également largement utilisées en cryptographie moderne. Pour le cryptographe, le *condensé* ainsi obtenu peut être vu comme une empreinte de la donnée reçue en entrée : si l'ensemble d'arrivée est suffisamment grand, deux données numériques différentes auront presque toujours deux condensés différents, cependant la seule connaissance d'un condensé ne permet pas de reconstituer les données originales. Elles n'ont donc pas vocation à transporter de l'information mais d'en offrir une empreinte compacte permettant de la caractériser, un peu de la même manière qu'une empreinte biométrique caractérise une personne physique.

En cryptographie, les fonctions de hachage forment donc une classe particulière de fonctions à sens unique de $\{0,1\}$ vers un ensemble $\mathcal{D}$ tel que $|\mathcal{D}| \leq 2^k$ avec k suffisamment petit. Cette caractérisation n'est cependant pas suffisante pour en assurer la sécurité et la fonction devra vérifier certaines propriétés supplémentaires. Il n'existe pas à l'heure actuelle de définition universelle-

ment admise d'une fonction de hachage cryptographiquement sûre, mais les trois propriétés énoncées dans la définition suivante sont généralement admises.

Définition 1.1.7 (Fonctions de hachage cryptographiquement sûres) *Une famille* $\mathbb{H}$ *de fonctions* $\mathcal{H} : \{0,1\} \to \mathcal{D}$ *(*$|\mathcal{D}| \leq 2^k$*) est* une famille de fonctions de hachage cryptographiquement sûres *si elle vérifie les trois propriétés suivantes :*

- résistance à la préimage *: toute fonction* $\mathcal{H} \in \mathbb{H}$ *est une fonction à sens unique ;*
- résistance à la seconde préimage *: il n'existe pas d'algorithme efficace permettant, étant donnés un message* $m \in \{0,1\}^*$ *et une fonction* $\mathcal{H} \in \mathbb{H}$*, de trouver un message* $m' \neq m$ *tel que* $\mathcal{H}(m') = \mathcal{H}(m)$ *;*
- résistance aux collisions *: il n'existe pas d'algorithme efficace permettant, étant donné une fonction* $\mathcal{H} \in \mathbb{H}$*, de trouver deux éléments différents* m *et* m' *dans* $\{0,1\}^*$ *tels que* $\mathcal{H}(m') = \mathcal{H}(m)$*.*

Dans la suite de ce mémoire, nous supposerons que toutes les fonctions de hachage utilisées sont de ce type.

Fonctions à sens unique à trappe

Une autre classe de fonctions à sens unique est largement utilisée en cryptographie asymétrique : les *fonctions à sens unique à trappe*. Informellement, il s'agit de fonctions facilement calculables, difficiles à inverser *sans la connaissance d'une donnée particulière* (la trappe) mais dont la connaissance permet l'inversion efficace.

Définition 1.1.8 (Fonction à sens unique à trappe) *Une fonction* $f : \mathcal{D} \to \mathcal{D}'$ *est dite* à sens unique à trappe *si :*

- f *est une fonction à sens unique,*
- *il existe un algorithme efficace* $\mathcal{I}$ *et un entier positif* c *tels que, pour tout entier* k*, il existe une suite de* l *bits* $t_k \in \{0,1\}^l$ *telle que* $l \leq k^c$ *et*

$$\forall x \in \mathcal{D},\ \mathcal{I}(1^k, f(x), t_k) = x' \text{ et } f(x') = f(x).$$

1.2 Primitives cryptographiques à clefs publiques

Cette section vise à présenter les différentes primitives cryptographiques que nous verrons dans la suite de ce mémoire. Nous en donnons ici uniquement la définition formelle ; les exigences de sécurité qui leurs seront associées seront définies par la suite (section 2.3). Les définitions données ici se restreignent volontairement au seul domaine de la cryptographie asymétrique.

1.2.1 Chiffrement

Le chiffrement est le procédé cryptographique permettant d'assurer la transmission confidentielle de messages. Supposons qu'un utilisateur A (Alexiane, par exemple) souhaite transmettre à un utilisateur B (disons Barnabé) des données de façon confidentielle. Pour y parvenir, Alexiane chiffre son message à l'aide de la clef publique de Barnabé, lequel utilise sa clef secrète (ou privée) pour le déchiffrer. De cette façon, tout le monde a la possibilité d'envoyer un message chiffré à Barnabé à condition d'avoir accès à sa clef publique. Cependant, puisqu'il est le seul à connaître sa clef secrète, il est le seul à même de retrouver le message clair à partir du message chiffré.

Définition 1.2.1 (Schéma de chiffrement à clef publique) *Un* schéma de chiffrement à clef publique *PKE est la donnée de quatre algorithmes :*

- *PKE.Initialiser, un algorithme* probabiliste *d'initialisation du système. Il prend en entrée un paramètre de sécurité κ et renvoie les paramètres publiques $\mathcal{P}$: $\mathcal{P} \leftarrow$ PKE.Initialiser(1^κ) ;*
- *PKE.GénérerClefs, un algorithme* probabiliste *de génération de clefs. Il prend en entrée les paramètres du système $\mathcal{P}$ et renvoie une paire de clefs (une clef privée et une clef publique associée) : $(SK, PK) \leftarrow$ PKE.GenererClefs$(\mathcal{P})$;*
- *PKE.Chiffrer, un algorithme (éventuellement probabiliste) de chiffrement. Il prend en entrée les paramètres publiques $\mathcal{P}$, la clef publique PK du destinataire et un message $m \in \mathcal{M}$. Il renvoie un chiffré $c \in \mathcal{C}$ de m : $c \leftarrow$ PKE.Chiffrer$(\mathcal{P}, PK, m)$;*
- *PKE.Déchiffrer, un algorithme* déterministe *de déchiffrement. Il prend en entrée les paramètres publiques $\mathcal{P}$, la clef secrète SK du destinataire et un chiffré $c \in \mathcal{C}$. Il renvoie le message m dont c est le chiffré selon la clef publique : $m \leftarrow$ PKE.Dechiffrer$(\mathcal{P}, SK, c)$;*

tels que le schéma soit consistant, *c'est-à-dire tels que $\forall \kappa \in \mathbb{N}^*$, $\forall \mathcal{P} \leftarrow$ PKE.Initialiser(1^κ), $\forall (PK, SK) \leftarrow$ PKE.GénérerClefs$(\mathcal{P})$:*

$$\forall m \in \mathcal{M},\ \forall c \leftarrow \text{PKE.Chiffrer}(\mathcal{P}, PK, m),\ \text{PKE.Dechiffrer}(\mathcal{P}, SK, c) = m.$$

Il n'est pas rare que les paramètres du système soient liés à la génération des clefs privées et publiques. Dans ce cas, les algorithmes PKE.Initialiser et PKE.GenererClef peuvent fusionner en un seul.

1.2.2 Signature

La signature numérique a pour but d'assurer par des moyens informatiques les mêmes garanties qu'une signature manuscrite, à savoir l'*authentification de l'origine des messages* (s'assurer que l'expéditeur du message est bien celui qu'il prétend être), l'*intégrité* des messages (s'assurer qu'un message n'a pas été modifié entre la signature du document et sa réception) et

la *non-répudiation* (s'assurer qu'un signataire ne pourra pas nier avoir apposé sa signature au document).

Une signature numérique est une donnée de petite taille apposée au document signé qui dépend à la fois du message et de la clef *privée* de l'expéditeur. La validité de cette donnée peut être vérifiée à l'aide de la clef publique du signataire. Lier la signature à la fois à la clef privée du signataire (qu'il est censé être le seul à connaître) et au document permet de s'assurer de son origine et de son intégrité. La non-répudiation est quant à elle assurée par la vérification à l'aide de la clef publique (dite vérification universelle) puisque celle-ci est accessible à tout le monde.

Définition 1.2.2 (Schéma de signature) *Un* schéma de chiffrement à clef publique S *est la donnée de quatre algorithmes :*

- $\mathsf{S.Initialiser}$*, un algorithme* probabiliste *d'initialisation du système. Il prend en entrée un paramètre de sécurité* κ *et renvoie les paramètres publiques* $\mathcal{P}$ *:* $\mathcal{P} \leftarrow \mathsf{S.Initialiser}(1^\kappa)$ *;*
- $\mathsf{S.GenererClefs}$*, un algorithme* probabiliste *de génération de clefs. Il prend en entrée les paramètres* $\mathcal{P}$ *du système et renvoie une paire de clefs (une clef privée et une clef publique associée) :* $(SK, PK) \leftarrow \mathsf{S.GenererClefs}(\mathcal{P})$ *;*
- $\mathsf{S.Signer}$*, un algorithme (éventuellement probabiliste) de signature. Il prend en entrée les paramètres* $\mathcal{P}$ *du système, la clef privée* SK *du signataire et un message* $m \in \{0,1\}^*$*. Il renvoie une signature* $\sigma \in \mathcal{S}$ *du message* m *avec la clef privée* SK *:* $\sigma \leftarrow \mathsf{S.Signer}(\mathcal{P}, SK, m)$ *;*
- $\mathsf{S.Vérifier}$*, un algorithme* déterministe *de vérification. Il prend en entrée les paramètres* $\mathcal{P}$ *du système, la clef publique* PK *du signataire, un message* $m' \in \{0,1\}^*$ *et une signature candidate* $\sigma' \in \mathcal{S}$*. Il renvoie* valide *si* σ' *est une signature valide du message* m' *pour la clef privée correspondant à* PK *et* $\mathsf{invalide}$ *sinon :* $\{\mathsf{valide}, \mathsf{invalide}\} \leftarrow \mathsf{S.Vérifier}(\mathcal{P}, PK, m', \sigma')$*.*

tels que le schéma soit consistant*, c'est-à-dire tels que* $\forall \kappa \in \mathbb{N}^*$*,* $\forall \mathcal{P} \leftarrow \mathsf{S.Initialiser}(1^\kappa)$*,* $\forall (PK, SK) \leftarrow \mathsf{S.GénérerClefs}(\mathcal{P})$ *:*

$$\forall m \in \{0,1\}^*,\ \forall \sigma \leftarrow \mathsf{S.Signer}(\mathcal{P}, SK, m),\ \mathsf{PKE.Vérifier}(\mathcal{P}, PK, m, \sigma) = \mathsf{valide}.$$

Comme pour le chiffrement, les paramètres du système peuvent être liés à la génération des clefs. Dans ce cas, les algorithmes S.Initialiser et S.GenererClef peuvent également fusionner.

Chapitre 2

Sécurité réductionniste de primitives cryptographiques

Sommaire

2.1 Prouver la sécurité des protocoles cryptographiques

La *sécurité prouvée* ou *Sécurité Réductionniste* a été introduite par S. Goldwasser et S. Micali dans un article de 1984 [GM84]. Dans cette approche, il est montré qu'il existe une réduction algorithmique (au sens de la définition 2.1.1) entre le problème posé par une attaque clairement identifiée sur un cryptosystème donné et un problème algorithmique défini. Si le coût algorithmique minimal (exprimé en nombre d'opérations et/ou en probabilité de succès) de résolution de ce problème peut être évalué, le coût algorithmique minimal de l'attaque ajouté au coût de la réduction ne peuvent être que supérieurs à cette estimation. Ainsi la complexité minimale de cette attaque peut être bornée.

2.1.1 Modéliser la sécurité

Dans la sécurité réductionniste, le choix du modèle de sécurité est crucial : c'est lui qui nous permet d'identifier les menaces contre lesquelles nous serons protégés. Il s'agit avant tout d'établir les objectifs d'un éventuel adversaire. Supposons qu'Alexiane ait l'habitude d'envoyer des courriers chiffrés à Barnabé et Constantin. Un espion pourrait souhaiter lire leurs correspondances. Dans ce cas, il lui faudra décrypter les messages expédiés par Alexiane. Mais l'espion pourrait vouloir uniquement connaître lequel de ces deux correspondants est le destinataire d'un message. Ceci ne nécessite pas forcement de décrypter le message, même si cela permettrait d'atteindre l'objectif. Ensuite, il faut déterminer quels sont les moyens dont pourrait disposer un adversaire. Il s'agit là de se poser des questions comme « de quelle machine dispose l'adversaire ? Une machine de Turing standard ? Un ordinateur quantique ? Une machine de puissance infinie ? » ou « Dispose-t-il de messages clairs avec leurs chiffrés ? Peut-il en obtenir de nouveaux ? » ou bien encore « Peut-il corrompre certains des utilisateurs ? » etc.

Un modèle de sécurité est donc composé d'un objectif et d'un ensemble de moyens dont dispose un éventuel adversaire. Si ce modèle est réaliste une réduction de sécurité permettra lors de l'utilisation de la primitive considérée de s'assurer de sa sécurité.

Modèles de calcul idéalisés

Cependant, pour prouver la sécurité de certaines constructions, il est parfois nécessaire de se placer dans un modèle de calcul idéalisé (par opposition au modèle standard) comme le modèle de l'oracle aléatoire ou le modèle du chiffrement idéal.

Modèle de l'oracle aléatoire Le *modèle de l'oracle aléatoire* ou ROM (pour *Random Oracle Model*, en anglais) est un modèle de calcul idéalisé introduit par M. Bellare et P. Roggaway en 1993 [BR93]. Ce modèle consiste à poser comme hypothèse que les fonctions de hachage se conduisent comme des fonctions aléatoires, c'est-à-dire que l'on ne peut pas distinguer leurs

valeurs dans une suite de valeurs aléatoires, à condition qu'elles produisent toujours le même résultat pour un même élément.

Lors de preuves de sécurité dans ce modèle, les fonctions de hachage sont vues comme étant fournies par un *oracle aléatoire*. Un appel à la fonction de hachage consiste alors à envoyer une requête à l'oracle contenant la suite de bits à hacher. L'oracle répond à toute nouvelle requête en fournissant une valeur uniformément distribuée dans l'espace de hachage. L'oracle est considéré maintenir une liste des réponses fournies afin de conserver le déterminisme de la fonction de hachage. Cet oracle appartient à l'environnement de l'adversaire, lequel n'a donc aucun contrôle sur lui. Sa seule interaction possible avec la fonction de hachage est la soumission de requêtes et la réception des réponses associées.

Il n'existe pas à l'heure actuelle de fonction permettant d'instancier une vraie fonction aléatoire. En 1998, R. Canetti and O. Goldreich et S. Halevi ont montré l'existence de constructions cryptographiques prouvées sûres dans le modèle de l'oracle aléatoire qui, une fois les oracles remplacés par des fonctions réelles, sont trivialement faibles [CGH04]. La sécurité réductionniste dans ce modèle est donc fondamentalement plus faible que la sécurité réductionniste dans le modèle standard. Cependant, pour des protocoles plus naturels que ceux présentés dans l'article de Canetti *et al.*, une preuve de sécurité dans le modèle de l'oracle aléatoire donne de forts arguments montrant qu'une attaque sur une instance réelle du schéma, pourvu qu'elle ne contredise aucune autre hypothèse de la preuve, découvrirait une propriété indésirable de la fonction de hachage utilisée .

Modèle du chiffrement idéal Ce modèle de calcul idéalisé est destiné à manipuler les fonctions de chiffrement symétrique ; de la même façon que le modèle de l'oracle aléatoire permet de manipuler les fonctions de hachage. Ce modèle suppose l'existence d'une famille de permutations aléatoires E_k paramétrée par un entier k. Pour chaque entier i, E_i et E_i^{-1} sont des permutations aléatoires de $\{0,1\}^\ell$.

Lors de preuves de sécurité dans ce modèle, la famille de fonctions de chiffrement considérée est modélisée par un oracle auquel l'adversaire soumet des requêtes de chiffrement ou de déchiffrement. Cet oracle peut être interrogé sur un entier k et un message $m \in \{0,1\}^\ell$ pour une requête de chiffrement (calculer $E_k(m)$) ou de déchiffrement (calculer $E_k^{-1}(m)$). L'oracle produit une valeur aléatoire pour toute nouvelle requête et fournit de nouveau la même réponse si la requête est répétée. De plus, l'oracle respecte la bijectivité des fonctions de chiffrement.

Le modèle du chiffrement idéal a tout d'abord été montré plus fort que le modèle de l'oracle aléatoire par J. S. Coron, Y. Dodis, C. Malinaud et P. Puniya [CDMP05] avant que l'équivalence entre les deux modèles soit établie à CRYPTO 2008 par J. S. Coron, J. Patarin et Y. Seurin [CPS08]. La confiance à accorder aux constructions prouvées dans chacun de ces deux modèles est identique ; ces deux modèles peuvent donc être utilisés conjointement au

sein d'une même construction sans en affaiblir la sécurité. En pratique, le chiffrement idéal est instancié à l'aide d'un schéma de chiffrement symétrique présentant de bonnes propriétés aléatoires, par exemple le standard américain AES [DR02].

2.1.2 Identifier les problèmes difficiles

Une fois le modèle de sécurité clairement établi, on veut montrer que l'existence d'un adversaire efficace dans ce modèle de sécurité implique l'existence d'un algorithme efficace pour résoudre un problème jugé *difficile*. Il est donc nécessaire d'identifier et d'étudier de tels problèmes.

La théorie de la complexité définit depuis longtemps des classes de problèmes considérés comme difficiles, telle que la classe des problèmes NP-complet ou plus généralement les problèmes NP-difficiles. Cependant, ces classes de complexité ne caractérisent que l'existence d'instances difficiles d'un problème alors que l'on souhaite en cryptographie que le problème soit difficile *en moyenne*. Choisir un problème NP-complet n'est donc pas suffisant, comme en témoignent les cryptanalyses des cryptosystèmes fondés sur des problèmes de type « sac à dos » (par exemple [Sha84], [Vau98]). Choisir un problèmes NP-complet est bien entendu possible, à condition de choisir ses paramètres avec précautions. On pourra se référer par exemple à [Mic10] pour une discussion.

Classes de complexité

Une part importante de la théorie de la complexité est consacrée à l'étude des problèmes de décision. Ce sont des problèmes dont la réponse est soit Oui soit Non. Sur ces problèmes, on définit en particulier les classes de complexité suivantes :

- la classe P des problèmes de décision que l'on peut résoudre en temps polynomial ;
- la classe NP des problèmes *Non-déterministes Polynomiaux* est formée des problèmes de décision qui peuvent être décidés sur une machine non déterministe en temps polynomial ; de façon équivalente, elle est la classe des problèmes décisionnels dont une solution Oui est vérifiable en temps polynomial, à l'aide d'une information supplémentaire appelée *certificat* ;
- de façon analogue, la classe $co - \mathsf{NP}$ des problèmes de décision dont une solution Non est vérifiable en temps polynomial.

Intuitivement, les problèmes de NP sont les problèmes qui peuvent être résolus en énumérant l'ensemble des solutions possibles et en les testant à l'aide d'un algorithme polynomial. On a évidemment $\mathsf{P} \subset \mathsf{NP}$, mais la conjecture $\mathsf{P} \neq \mathsf{NP}$ est toujours ouverte.

La définition des machines de Turing probabilistes nous permet de définir la notion de réduction algorithmique de problèmes et celle de C-complétude d'un problème.

Définition 2.1.1 (Réduction algorithmique de problèmes) *Soient $\mathbb{P}_1$ et $\mathbb{P}_2$ deux problèmes algorithmiques. Le problème $\mathbb{P}_1$ se réduit au problème $\mathbb{P}_2$ si l'existence d'une machine de Turing M_2 résolvant le problème $\mathbb{P}_2$ permet de construire une machine de Turing $M_1^{M_2}$ résolvant le problème $\mathbb{P}_1$ en utilisant M_2 comme oracle et qui soit de complexité inférieure à celle de M_2.*

Dans un tel cas, $\mathbb{P}_2$ est dit au moins aussi difficile *que $\mathbb{P}_1$ (noté $\mathcal{P}_2 \succeq \mathcal{P}_1$) et $\mathbb{P}_1$ est dit* au moins aussi facile *que $\mathbb{P}_2$ (noté $\mathcal{P}_1 \preceq \mathcal{P}_2$). Les problèmes $\mathbb{P}_1$ et $\mathbb{P}_2$ sont dits* équivalents *si $\mathbb{P}_1$ est à la fois au moins aussi difficile et au moins aussi facile que $\mathbb{P}_2$.*

Définition 2.1.2 (Problème C-difficile, Problème C-complet) *Soit C une classe de complexité. Un problème est dit* C-difficile *s'il est au moins aussi difficile que tous les problèmes dans C ; s'il appartient lui même à C, il est alors dit* C-complet.

2.1.3 Réduction de sécurité

Une réduction de sécurité est similaire à la notion de réduction algorithmique de problème. Elle consiste à utiliser un éventuel adversaire contre le schéma étudié pour construire un algorithme permettant de résoudre un problème identifié comme difficile. Pour y parvenir, l'adversaire est plongé dans une simulation reproduisant l'environnement correspondant au modèle de sécurité dans lequel le schéma est analysé. Puis la simulation est modifiée de façon à forcer l'adversaire à résoudre lors de son attaque une instance aléatoire de notre problème difficile.

Ces modifications introduites dans la simulation de l'environnement peuvent influencer les performances de l'adversaire et ainsi permettre la construction d'un algorithme plus ou moins efficace (que ce soit en nombre d'opérations élémentaires effectuées ou en probabilité de succès) pour résoudre le problème considéré. Il est donc important d'évaluer l'influence de chacune des modifications effectuées sur la simulation. Plus les performances de l'algorithme ainsi obtenues seront proches de celles de l'adversaire dans son environnement d'origine, plus la réduction sera dite *fine*.

Approche par jeu

L'évaluation de la qualité d'une réduction est une étape importante des preuves de sécurité réductionnistes mais celles-ci peuvent parfois être compliquées à établir. Au cours du temps, divers formalismes visant à simplifier cette évaluation ont étés proposés. Nous présentons ici une méthode, connue sous le nom de *preuve par jeux*, apparue dans la littérature sous diverses formes et avec différents niveaux de formalisme. En 1994, V. Shoup a proposé un tutoriel [Sho04], revu pour la dernière fois en janvier 2006, sur une de ces méthodes qui semble atteindre un niveau raisonnable de rigueur mathématique.

Dans cette approche, la sécurité d'une primitive cryptographique est définie par un *jeu de sécurité* joué entre un *adversaire* et une entité extérieure appelée *challenger* qui contrôle l'en-

vironnement de l'adversaire. Adversaire et challenger sont tous deux des machines de Turing probabilistes pouvant communiquer entre elles, et le jeu peut donc être modélisé par un *espace de probabilité*. En règle générale, la définition d'une propriété de sécurité est reliée à un événement particulier S. La preuve de sécurité est établie en produisant une séquence de jeux Jeu 0, Jeu 1, ..., Jeu n, où Jeu 0 est le jeu de sécurité original pour le type d'adversaire considéré contre la primitive de sécurité. Pour $i = 1, \ldots, n$, la construction définit un événement S_i dans le Jeu i, généralement relié naturellement à la définition de S. Dans Jeu 0, l'événement S_0 est naturellement l'événement S lui-même. La preuve montre d'une part qu'il est possible de relier (polynomialement) les probabilités $\Pr[S_i]$ et $\Pr[S_{i+1}]$ et d'autre part que $\Pr[S_n]$ peut être évaluée simplement. Il est alors possible d'évaluer simplement la probabilité $\Pr[S_0] = \Pr[S]$.

Ceci donne le canevas général des preuves par jeu. Cependant, lors de la construction de telles preuves, il est souhaitable que les changements entre deux jeux consécutifs soient minimes de façon à ce que l'analyse de ces changements soit aussi simple que possible. V. Shoup présente trois types de transitions auxquelles il est possible de se restreindre [Sho04].

Transitions fondées sur l'indistinguabilité Lors de ces transitions, le changement réalisé, s'il était détecté par l'adversaire, permettrait de construire une méthode efficace pour distinguer deux distributions qui sont indistinguables (soit statistiquement, soit calculatoirement).

Par exemple, supposons que les distributions P_1 et P_2 soient considérées comme calculatoirement indistinguables. Pour prouver que la quantité $|\Pr[S_i] - Pr[S_{i+1}]|$ est négligeable, on montre qu'il est possible de construire un *distingueur* qui renvoie 1 avec probabilité $\Pr[S_i]$ s'il reçoit en entrée un élément qui provient de P_1, et renvoie 1 avec probabilité $\Pr[S_{i+1}]$ s'il reçoit en entrée un élément qui provient de P_2. Une méthode classique consiste à concevoir les deux jeux de façon à ce qu'ils puissent facilement être récrit comme un unique *jeu hybride* recevant une entrée auxiliaire. Si celle-ci provient de P_1 (resp. P_2), alors le jeu hybride est exactement le Jeu i (resp. Jeu $i + 1$). Dans ce cas, le distingueur exécute simplement le jeu hybride avec son entrée et renvoie 1 si l'événement approprié se produit.

Transitions fondées sur des événements d'échec Lors de ces transitions, les jeux i et $i + 1$ fonctionnent de façon identique à moins qu'un événement d'échec E se produise. Dans l'idéal, les deux jeux sont définis sur le même espace de probabilité (les seules différences entre les deux jeux proviennent de la façon dont sont calculées certaines variables aléatoires). Dans ce cas, dire que les deux jeux fonctionnent de façon identique à moins qu'un événement d'échec E se produise est équivalent à dire que $S_i \wedge \neg E \iff S_{i+1} \wedge \neg E$, c'est-à-dire que les événements $S_i \wedge \neg E$ et $S_{i+1} \wedge \neg E$ sont *les mêmes*. Si cela est vérifié, il est alors possible d'utiliser le lemme suivant :

Lemme 2.1.1 (Lemme de la différence – Difference Lemma) *Soient A, B, E des événements d'une même distribution de probabilité. Si $A \wedge \neg E \iff B \wedge \neg E$, alors*

$$|\Pr[A] - \Pr[B]| \leq \Pr[E]$$

.

Preuve – La preuve est purement calculatoire :

$$\begin{aligned} |\Pr[A] - \Pr[B]| &= |\Pr[A \wedge F] + \Pr[A \wedge \neg F] - \Pr[B \wedge F] - \Pr[B \wedge \neg F]| \\ &= |\Pr[A \wedge F] - \Pr[B \wedge F]| \\ &\leq \Pr[F] \end{aligned}$$

La seconde égalité provient de l'équivalence $A \wedge \neg E \iff B \wedge \neg E$ qui établit en particulier que $\Pr[A \wedge \neg F] = \Pr[B \wedge \neg F]$. L'inégalité finale est donnée par le fait que $\Pr[A \wedge F]$ et $\Pr[B \wedge F]$ sont deux quantités positives plus petites que $\Pr[F]$. $\diamondsuit$

Pour montrer que la quantité $|\Pr[S_i] - Pr[S_{i+1}]|$ est négligeable, il suffit donc de montrer que la probabilité $\Pr[S]$ est négligeable.

Transitions de pont Ce type de transitions se produit lors de la redéfinition de la manière dont certaines quantités sont calculées, de façon totalement équivalente. Les changements réalisés sont purement conceptuels et donc $\Pr[S_i] = Pr[S_{i+1}]$. La raison d'être de ces transitions est de préparer le terrain à une autre transition, de l'un des deux types précédents. En principe, ces transitions ne sont pas nécessaires, cependant elles permettent une compréhension plus aisée de la preuve.

2.2 De la sécurité réductionniste à la sécurité pratique

L'approche par sécurité réductionniste se heurte à plusieurs obstacles. Tout d'abord, si cette approche protège contre l'attaque considérée et celles qui peuvent lui être reliées, elle ne protège en rien contre un scénario d'attaque totalement nouveau. Le choix d'un modèle de sécurité cohérent est donc primordial. Ensuite, il est nécessaire que le coût de la résolution du problème algorithmique soit clairement établi : toute avancée significative dans la résolution de celui-ci réduirait d'autant les assurances de sécurité obtenues par la réduction. Et même alors que ce coût est établi, il doit présenter une croissance maximale en fonction de la taille de ses entrées, idéalement exponentielle voire sous-exponentielle afin de rester aussi longtemps que possible au dessus de la courbe de progression de la puissance des ordinateurs (estimée par les lois empiriques de Moore). Enfin, une réduction de sécurité coûteuse fera diminuer d'autant la borne sur la complexité de l'attaque ainsi obtenue. Une approche plus fine (si elle existe) pourrait permettre

de diminuer la taille des paramètres à utiliser afin d'assurer une difficulté suffisante au problème algorithmique pour compenser le coût de la réduction.

La situation idéale serait donc une réduction en temps constant à un problème algorithmique dont la difficulté progresse exponentiellement en la taille de ses entrées. Malheureusement, cette situation est peu réaliste : souvent, en plus d'une progression fonction de la taille des entrées, de nouveaux paramètres introduits par le modèle de sécurité agissent sur la complexité de la réduction. Les premiers modèles pour la sécurité réductionniste ont tout d'abord été définis pour le chiffrement [GM84] puis pour la signature [GMR85, GMR88]. Cependant, même si les réductions proposées avaient des complexités polynomiales, le coût de ces réductions restait trop élevé. Par la suite, M. Bellare et P. Rogaway ont introduit le concept de *sécurité exacte*, suivis par K. Ohta et T. Okamoto qui ont proposé celui de *sécurité concrète* puis par D. Pointcheval qui a proposé celui de *sécurité pratique*. Supposons l'existence d'un attaquant $\mathcal{A}$ réalisant une attaque en temps t et supposons l'existence d'une réduction permettant de résoudre le problème difficile en temps $f(t)$. On définit alors les trois types de sécurité suivants :

- la sécurité *asymptotique* lorsque f est majorée par un polynôme en t ;
- la sécurité *exacte* lorsque f est explicite ;
- et la sécurité *pratique* lorsque f est "petite" (linéaire, par exemple).

2.3 Modèles de sécurité pour le chiffrement et la signature

2.3.1 Chiffrement

Dans le cadre d'une attaque sur un schéma de chiffrement, un adversaire commence par cibler l'utilisateur qu'il veut attaquer et se procure sa clef publique. Il tente alors d'atteindre l'un des objectifs suivants :

- **le bris total.** L'adversaire parvient à déduire la clef privée de la clef publique de l'utilisateur ciblé.
- **le bris du sens unique de la fonction de chiffrement** (One-wayness – OW). L'adversaire est capable de retrouver le message clair correspondant à tout message chiffré à l'aide de la clef privée de l'utilisateur ciblé.

L'adversaire peut également être moins ambitieux et tenter de s'attaquer à l'une des propriétés suivantes :

- **non-malléabilité – NM.** L'adversaire tente de produire un chiffré c' d'un message m' inconnu à partir d'un chiffré c connu d'un message m inconnu.
- **indistinguabilité – IND** (ou *sécurité sémantique*). Cette propriété n'est présente que pour des schémas de chiffrement probabilistes. L'adversaire tente de distinguer les chiffrés de deux messages différents de son choix.

Pour réaliser ces attaques, l'adversaire peut avoir des ressources diverses à sa disposition. On distingue les attaques suivantes :

- **attaque à textes clairs choisis** (Chosen Plaintext Attack – CPA) : l'adversaire peut obtenir les chiffrés des messages clairs de son choix. En cryptographie à clef publique, l'attaquant est toujours à même de mener ce type d'attaque puisqu'il dispose de la clef publique de l'utilisateur ciblé ;
- **attaque à chiffrés choisis** (Chosen Cyphertext Attack – CCA) : l'adversaire peut obtenir, pendant une période déterminée de son attaque, le déchiffrement de chiffrés de son choix ;
- **attaque à chiffrés choisis adaptative** (Adaptative Chosen Cyphertext Attack – CCA2) : l'adversaire peut obtenir tout au long de son attaque le déchiffrement de chiffrés de son choix. Il est donc en mesure d'adapter ses demandes de déchiffrement au cours de son attaque.

Dans la suite de ce mémoire, on dira d'un schéma présentant une propriété de sécurité XX face à un adversaire menant une attaque YY qu'il est XX-YY. Par exemple, un schéma indistinguable pour un adversaire menant une attaque à chiffrés choisis adaptative sera dit IND-CCA2. En 1998, M. Bellare, A. Desai, D. Pointcheval et P. Rogaway ont démontré un certain nombre de relations (ou de séparations) entre les différents modèles de sécurité ci-dessus [BDPR98]. Ces relations auxquelles est ajoutée la place du modèle OW-CPA sont représentées sur la figure 2.1.

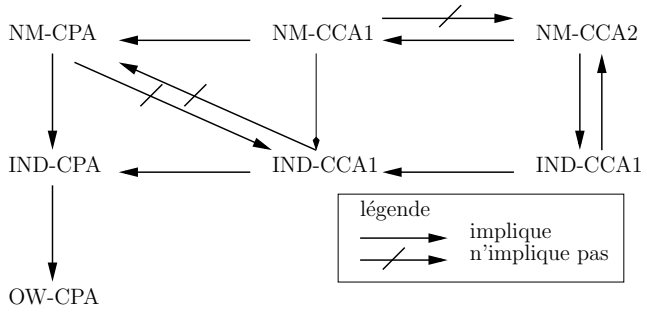


Figure 2.1 – Relations entre les modèles de sécurité pour le chiffrement à clef publique

2.3.2 Signature

Lors d'une attaque sur un schéma de signature, un adversaire commence par cibler l'utilisateur qu'il veut attaquer et se procure sa clef publique. Il tente alors d'atteindre l'un des objectifs suivants :

- **le bris total** : comme pour un schéma de chiffrement, l'adversaire parvient à retrouver la clef privée de l'utilisateur ciblé ;

- **la contrefaçon universelle** (Universal Forgery – UF) : l'attaquant est capable de signer pour l'utilisateur ciblé n'importe quel message ;
- **la contrefaçon sélective** (Selective Forgery – SF) : L'attaquant est capable de créer une signature pour l'utilisateur ciblé un message de son choix choisi dans une liste de messages ;
- **la contrefaçon existentielle** (Existential Forgery – EF) : l'attaquant est capable de fournir une signature valide pour un message de son choix et la clef publique de l'utilisateur ciblé.

Ces objectifs forment une hiérarchie de difficultés puisque, de façon évidente, un adversaire réussissant un bris total est capable de produire une contrefaçon universelle, un adversaire réalisant une contrefaçon universelle est à même de produire une contrefaçon sélective et un adversaire réalisant une contrefaçon sélective est alors capable de produire une contrefaçon existentielle. La contrefaçon existentielle est donc l'objectif le plus facile à atteindre et, en raisonnant par contraposée, si aucun adversaire ne peut parvenir à produire une contrefaçon existentielle, aucun adversaire ne sera à même de produire l'un des modèles d'attaque plus difficile.

Pour mener à bien l'une de ces attaques, un adversaire peut disposer de différents moyens. On distingue trois types d'attaques :

- **attaque sans message** : l'attaquant est uniquement en possession de la clef publique de l'utilisateur ciblé.
- **attaque à messages connus** (Known Message Attack – KMA) : l'attaquant est en possession non seulement de la clef publique de l'utilisateur ciblé, mais aussi d'une liste de couples composés d'un message et de sa signature par la clef privée de l'utilisateur ciblé.
- **attaque à messages choisis** (Chosen Message Attack – CMA) : l'attaquant possède la clef publique de l'utilisateur ciblé et est capable d'obtenir des signatures de messages de son choix. Il est alors capable d'adapter au cours de l'attaque le choix des messages dont il obtient la signature, c'est pourquoi on parle aussi d'attaque à messages choisis adaptative.

De la même façon que pour les objectifs, ces modèles forment une hiérarchie dont l'attaque à messages choisis est la plus facile. Dans la mesure du possible, on tentera donc de prouver l'impossibilité de réaliser une contrefaçon existentielle lors d'une attaque à messages choisis. On parle alors de sécurité EF-CMA.

Chapitre 3

Codes Correcteurs d'erreurs pour la cryptographie

Sommaire

3.1 Éléments de théorie des codes correcteurs d'erreurs

Les codes correcteurs d'erreurs visent à protéger les données lors de leur transmission par le biais d'un canal bruité. Les canaux de transmissions sont nombreux : ils peuvent être spatiaux (une liaison ADSL, par exemple) ou temporels (un CD-ROM, par exemple), reposer sur des techniques physiques différentes, ... et les sources d'erreurs (le bruit) sont plus nombreuses encore.

Prenons l'exemple d'une ligne électrique sur laquelle peuvent être émis deux types d'impulsions, l'une représentée par le symbole 1 et l'autre par le symbole 0. Lorsque la transmission se passe bien le symbole reçu est le même que le symbole émis. Supposons que parfois, disons avec probabilité p, la transmission se passe mal et un symbole est reçu pour un autre. Ce type de canal, extrêmement simple est appelé un *canal binaire symétrique* (Fig. 3.1).

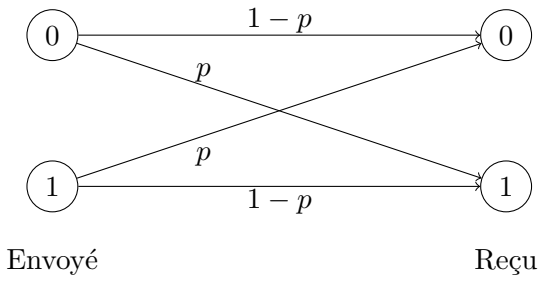


FIGURE 3.1 – Canal binaire symétrique avec probabilité d'erreur p.

Afin de donner aux messages une certaine résistance aux erreurs produites par ce canal, ceux-ci sont *encodés* en leur ajoutant de l'information. Un bloc de k symboles $\mathbf{u} = u_1 u_2 \dots u_k$ ($u_i \in \{0,1\}$) est encodé en un *mot de code* $\mathbf{x} = x_1 x_2 \dots x_n$ ($x_i \in \{0,1\}$) avec $n \geq k$. L'ensemble de ces mots de code forment un *code*. L'objectif de la théorie des codes correcteurs d'erreurs est de faire en sorte que les $n-k$ symboles ajoutés permettent de retrouver le mot de code transmis (et ainsi de retrouver le message original) tout en minimisant cette quantité $n-k$.

3.1.1 Codes linéaires

Les codes actuellement les plus utilisés forment la famille des *codes linéaires*. Dans ces codes, les messages que l'on veut coder sont vus comme des vecteurs de longueur k d'éléments d'un corps fini à q éléments $\mathbb{F}_q$. Ces éléments de $\mathbb{F}_q^k$ sont transformés en un élément de $\mathbb{F}_q^n, n \geq k$ par une *application linéaire*. L'ensemble des images de $\mathbb{F}_q^k$ par cette application forme le *code* $\mathcal{C}$ qui est donc un sous-espace vectoriel de $\mathbb{F}_q^n$ et peut être décrit entièrement par une base formée de k vecteurs linéairement indépendants de $\mathbb{F}_q^n$.

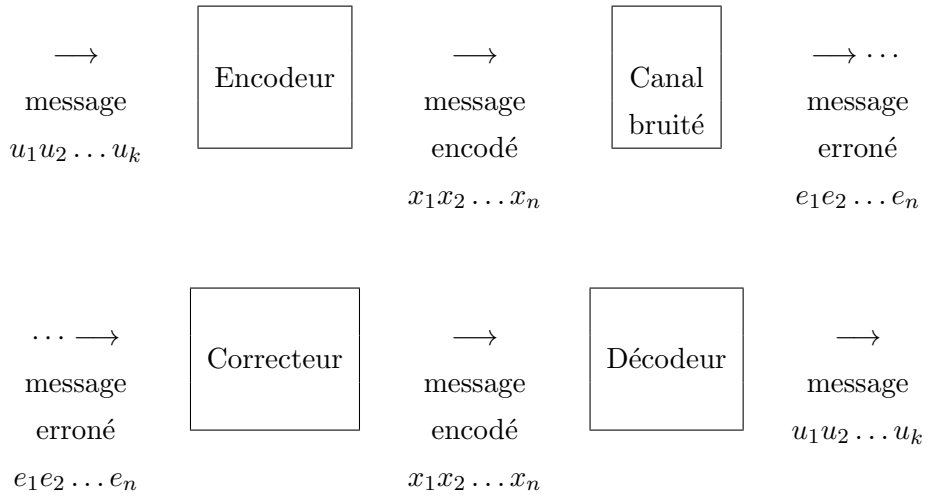


FIGURE 3.2 – Protection de l'information lors du passage par un canal bruité à l'aide d'un code correcteur d'erreur.

Définition 3.1.1 (Matrice génératrice) *Soit $\mathcal{C}$ un code linéaire sur $\mathbb{F}_q$ de longueur n et de dimension k. Une* matrice génératrice *G de $\mathcal{C}$ est une matrice $k \times n$ dont les lignes forment une base de $\mathcal{C}$.*

Si G s'écrit sous la forme $G = (\mathcal{I}_k|A)$ où $\mathcal{I}_k$ est la matrice identité de dimension k et A est une matrice $k \times (n-k)$, on dit alors que G est sous forme systématique*.*

L'encodage d'un élément x de $\mathbb{F}_q^k$ en un élément $c \in \mathcal{C}$ peut être réalisé simplement en calculant le produit $c = xG$. Si la matrice G est sous forme systématique, les k premiers symboles de c sont exactement les k éléments de x ; ils sont appelés *symboles d'information*, les $n-k$ derniers symboles forment les *symboles de redondance*.

Un code linéaire peut également être défini par une matrice de parité dont nous donnons la définition ci-dessous.

Définition 3.1.2 (Code dual) *Soit $\mathcal{C}$ un code de longueur n sur $\mathbb{F}_q$. Le* code dual *de $\mathcal{C}$, noté $\mathcal{C}^\perp$ est l'ensemble des vecteurs qui sont orthogonaux à tous les mots du code $\mathcal{C}$:*

$$\mathcal{C}^\perp = \left\{\mathbf{x} \in \mathbb{F}_q^n \mid \mathbf{x} \cdot \mathbf{c} = 0 \text{ pour tout } \mathbf{c} \in \mathcal{C}\right\}$$

Définition 3.1.3 (Matrice de parité) *Soit $\mathcal{C}$ un code linéaire de matrice génératrice G. Une* matrice de parité *H de $\mathcal{C}$ est une matrice génératrice du code $\mathcal{C}^\perp$. On a : $\mathcal{C} = \ker H$, c'est-à-dire :*

$$\mathcal{C} = \{x \in \mathbb{F}_q^n \mid Hx^T = 0\}.$$

De plus, si G est sous forme systématique $G = (\mathcal{I}_k|A)$, alors $H = (A^T|\mathcal{I}_{n-k})$ où $\cdot^T$ désigne la transposition.

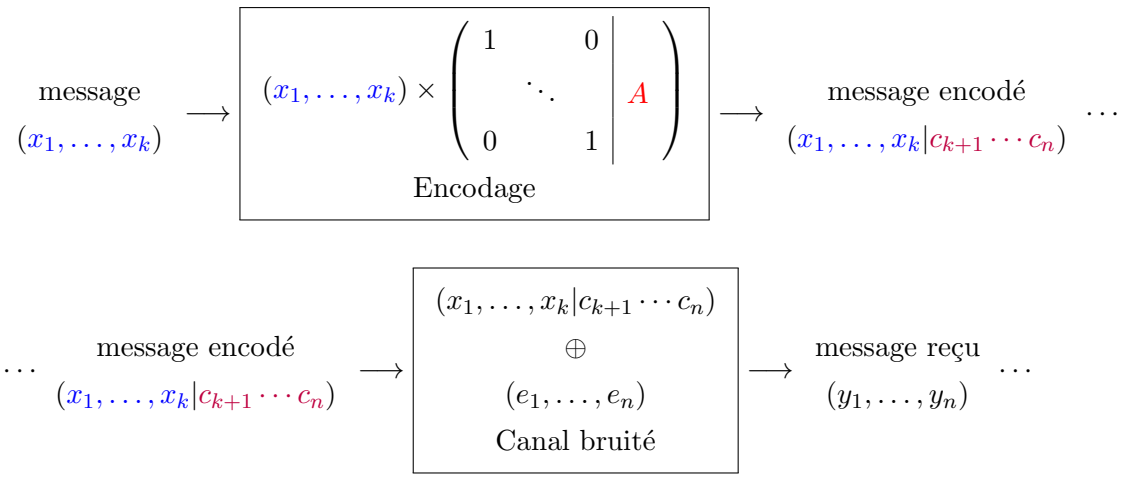


FIGURE 3.3 – Encodage et transmission d'un message par un code linéaire dont la matrice génératrice est sous forme systématique.

Durant la transmission, un vecteur d'erreur $e \in \mathbb{F}_q^n$ est ajouté au vecteur c pour former le message reçu $y = c + e$. Informellement, on appellera *correction* le fait de retrouver e tel que $y - e \in \mathcal{C}$ et *décodage* l'action consistant à retrouver à partir d'un mot $y \in \mathcal{C}$ le message m tel que $y = mG$ où G est une matrice génératrice du code $\mathcal{C}$. Si G est sous forme systématique, ce décodage peut alors être fait simplement en lisant les k premiers éléments de $y - e$. Par abus de langage, on appellera parfois *décoder* l'action de retrouver une erreur e telle que $y - e \in \mathcal{C}$.

Afin de définir formellement les notions précédentes, il est utile d'introduire les définitions suivantes :

Définition 3.1.4 (Support, Distance et Poids de Hamming) *Soient* $\mathbf{x} = (x_1, \ldots, x_n)$ *et* $\mathbf{y} = (y_1, \ldots, y_n)$ *deux vecteurs de* $\mathbb{F}_q^n$.

- *Le* support *de* $\mathbf{x}$, *noté* $supp(\mathbf{x})$ *est l'ensemble des indices des positions non nulles de* $\mathbf{x}$ *:* $supp(\mathbf{x}) = \{1 \leq i \leq n \mid x_i \neq 0\}$.
- *Le* poids de Hamming *de* $\mathbf{x}$, *noté* $wt(\mathbf{x})$ *est le nombre de positions non nulles de* $\mathbf{x}$ *:* $wt(\mathbf{x}) = |supp(\mathbf{x})|$.
- *La* distance de Hamming *entre* $\mathbf{x}$ *et* $\mathbf{y}$, *notée* $dist(\mathbf{x}, \mathbf{y})$ *est le nombre de positions où ils diffèrent. De façon évidente,* $dist(\mathbf{x}, \mathbf{y}) = wt(\mathbf{x} - \mathbf{y})$.

Le poids et la distance de Hamming forment respectivement une norme et une distance sur $\mathbb{F}_2^n$.

Nous établissons les définitions suivantes :

Définition 3.1.5 (Encodeur, décodeur, t-correction) *Soit* $\mathcal{C}$ *un code défini par une matrice de parité* G *de taille* $k \times n$ $(n \geq k)$.

- *Un encodeur est une fonction* $\mathcal{E}nc : \mathbb{F}_q^k \to \mathcal{C} \subset \mathbb{F}_q^n$ *qui à* x *associe* xG *et telle que* $\mathcal{E}nc(x) \neq \mathcal{E}nc(x')$ *si* $x \neq x'$ *(*$\mathcal{E}nc$ *est surjective).*
- *Un décodeur est une fonction* $\mathcal{D}ec : \mathcal{C} \to \mathbb{F}_q^k$ *qui associe* x *à* y *si* $y = \mathcal{E}nc(x)$
- *Un* t*-correcteur est une fonction* $\mathcal{C}orr : \mathbb{F}_q^n \to \mathcal{C}$ *qui renvoie* c *si et seulement si* $x = c + e$ *où* $e \in \mathbb{F}_q^n$ *et* $wt(e) \leq t$ *et renvoie* $\perp$ *(erreur) sinon.*

Par abus de langage, on dira que $\mathcal{D}ec$ *corrige* t *erreurs (ou est* t*-correcteur) si et seulement si* $\forall e \in \mathbb{F}_q^n$ *tq* $wt(e) \leq t$, $\mathcal{D}ec(\mathcal{E}nc(x) + e) = x$.

On dira que le décodage est complet *(ou total) si quelque soit le mot décodé, le décodeur renvoie toujours un mot de code. Si* $\mathcal{D}ec(\mathcal{E}nc(x) + e) = x' \neq x$, *on dira qu'une* erreur de décodage *s'est produite.*

Le troisième paramètre d'un code linéaire $\mathcal{C}$ par ordre d'importance, après sa longueur et sa dimension, est sa *distance minimum.* Il s'agit du minimum d des distances de Hamming entre deux de ses mots :

$$\begin{aligned} d &= \min\ dist(\mathbf{x}, \mathbf{y}) \\ &= \min\ wt(\mathbf{x} - \mathbf{y}), \quad \mathbf{x} \in \mathcal{C}, \mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y} \end{aligned}$$

Dans le cas d'un code linéaire, la différence de deux mots du code est elle aussi dans le code. On a donc

$$d = \min_{\mathbf{w} \in \mathcal{C}, \mathbf{w} \neq \mathbf{0}} wt(\mathbf{w})$$

Définition 3.1.6 (code $[n, k, d]$**)** *Un code linéaire de longueur* n*, de dimension* k *et de distance minimum* d *est appelé un code* $[n, k, d]$.

Calculer la distance minimum d'un code de longueur n, de dimension k peut être extrêmement coûteux puisqu'il faut énumérer les 2^k mots du code et calculer leur poids, ce qui est rarement réalisable en pratique. Il existe cependant une borne, dite *du singleton*, permettant de l'estimer.

Théorème 3.1.1 (Borne du singleton) *Tout code* $[n, k, d]$ *vérifie la majoration* $n - k \geq d - 1$.

Si cette borne est atteinte, le code est dit MDS *(pour* Maximum Distance Separable*) ou* parfait.

Preuve – Un mot de code contenant un unique symbole d'information est de poids au plus $n - k + 1$. Par suite, $d \leq n - k + 1$. $\diamondsuit$

De plus, la borne de Gilbert-Varshamov permet de s'assurer de l'existence de codes de paramètres donnés.

Théorème 3.1.2 (Borne de Gilbert-Varshamov) *Il existe un code linéaire sur* $\mathbb{F}_q$ *de longueur* n*, de dimension* k *et de distance minimale* $\geq d$ *pourvu que*

$$GV_q(n, d) \stackrel{def}{=} \sum_{i=0}^{d-1} \binom{n}{i} (q-1)^i \geq q^{n-k}.$$

La preuve de ce théorème peut être trouvée dans [Moo05] ou [Wel88], par exemple.

Décodage à maximum de vraisemblance

Le décodage à maximum de vraisemblance consiste à associer au mot reçu le mot de code le plus proche au sens de la distance de Hamming. Par exemple, soit le code binaire $\mathcal{C}$ défini par la matrice de parité

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

La première colonne de la figure 3.4 énumère les mots du code et leur poids ; on observe alors que la distance minimale de $\mathcal{C}$ est $d = 3$. Supposons que durant deux transmissions du mot $\mathbf{c} = (0, 1, 1, 0, 1, 1)$ les erreurs $\mathbf{e} = (0, 1, 0, 0, 0, 0)$ et $\mathbf{e}' = (0, 0, 1, 0, 1, 0)$ se produisent. Les mots reçus sont donc respectivement $\mathbf{y} = \mathbf{c} + \mathbf{e} = (0, 0, 1, 0, 1, 1)$ et $\mathbf{y}' = \mathbf{c} + \mathbf{e}' = (0, 1, 0, 0, 0, 1)$. Les deux dernières colonnes de la figure 3.4 énumèrent les distances de $\mathbf{y}$ et $\mathbf{y}'$ à chacun des mots du code ; le message $\mathbf{y}$ est donc décodé en $\mathbf{c} = (0, 1, 1, 0, 1, 1)$ alors que $\mathbf{y}'$ est décodé par le mot $\mathbf{c}' = (0, 1, 0, 1, 0, 1)$. Le décodage à maximum de vraisemblance fournit donc un critère de décodage complet.

Mot de code	poids de Hamming	distance au mot $(0, 0, 1, 0, 1, 1)$	distance au mot $(0, 1, 0, 0, 0, 1)$
$(1, 0, 0, 0, 1, 1)$	3	2	3
$(0, 1, 0, 1, 0, 1)$	3	4	1
$(0, 0, 1, 1, 1, 0)$	3	2	5
$(1, 1, 0, 1, 1, 0)$	4	5	4
$(1, 0, 1, 1, 0, 1)$	4	3	4
$(0, 1, 1, 0, 1, 1)$	4	1	2
$(1, 1, 1, 0, 0, 0)$	3	4	3

FIGURE 3.4 – Décodage à maximum de vraisemblance

Décodage à distance bornée

Si le décodage complet est intéressant, sa mise en œuvre est souvent coûteuse et pas toujours souhaitable : on peut préférer ne corriger que les erreurs dont on est sûr. Dans ce contexte, la propriété 3.1.1 nous permet d'établir un autre critère de décodage, celui-ci non complet : le *décodage à distance bornée*, défini pour une certaine borne t. S'il s'avère qu'un message a été altéré par moins de t erreurs, alors le message est corrigé, sinon il est rejeté.

Propriété 3.1.1 *Soient $\mathcal{C}$ un code $[n,k,d]$ sur $\mathbb{F}_q$ et $t = \lfloor \frac{d-1}{2} \rfloor$. Soit $\mathcal{B}(y,t)$ la boule de centre y et de rayon t, c'est-à-dire l'ensemble des vecteurs $v \in \mathbb{F}_q^n$ tels que $dist(y,v) \leq t$. Alors pour tout y de $\mathbb{F}_q^n$, $\mathcal{B}(y,t)$ contient au plus un mot $x \in \mathcal{C}$.*

PREUVE – Supposons l'existence de deux mots x et x' appartenant à $\mathcal{B}(y,t) \cap \mathcal{C}$. Leur appartenance à $\mathcal{B}(y,t)$ garantit que leurs distances à y sont toutes deux inférieures à t et par inégalité triangulaire que $dist(x,x') \leq 2t < d$. Or, x et x' sont également des mots de $\mathcal{C}$, leur distance ne peut donc être inférieure à d. Il ne peut donc pas exister simultanément deux mots de $\mathcal{C}$ dans $\mathcal{B}(y,t)$. $\diamondsuit$

Cette propriété nous permet aussi de voir que la capacité de correction à distance bornée d'un code ne dépend pas de l'ordre de ses symboles. On définit l'équivalence de code comme suit :

Définition 3.1.7 (Équivalence de codes) *Deux codes sont dits* équivalents *si leurs matrices génératrices (resp. de parité) se déduisent l'une de l'autre par permutation de colonnes. On notera cette équivalence $\mathcal{C}_1 \equiv \mathcal{C}_2$ et on étendra cette notation à leurs matrices génératrices (resp. de parité).*

Par exemple, les codes de génératrices $G = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$ et $G' = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$ sont équivalents.

Décodage par syndrome

Le décodage par syndrome est une technique permettant de réaliser un décodage à distance bornée. Elle s'appuie sur la notion de *coset* :

Définition 3.1.8 (Coset) *Soit $\mathcal{C}$ un code $[n,k,d]$ sur $\mathbb{F}_q$. Pour tout vecteur $\mathbf{x} \in \mathbb{F}_q^n$, on appelle* Coset *(ou* translaté*) de $\mathcal{C}$ l'ensemble*

$$\mathbf{a} + \mathcal{C} = \{\mathbf{a} + \mathbf{c} : \mathbf{c} \in \mathcal{C}\}.$$

Tout vecteur $\mathbf{x} \in \mathbb{F}_q^n$ est dans un coset (au moins $\mathbf{x} + \mathcal{C}$) et chaque coset contient q^k vecteurs. Deux vecteurs $\mathbf{x}$ et $\mathbf{y}$ sont dans le même coset si et seulement si $(\mathbf{x} - \mathbf{y}) \in \mathcal{C}$. On peut également remarquer que deux cosets soit sont disjoints, soit coïncident : il ne peut pas y avoir de chevauchement partiel entre deux cosets (voir [MS77], Ch. 1, §4, par exemple). Cette propriété permet de partitionner l'espace $\mathbb{F}_q^n$ en cosets de $\mathcal{C}$:

$$\mathbb{F}_q^n = \mathcal{C} \cup (\mathbf{x_1} + \mathcal{C}) \cup \cdots \cup (\mathbf{x}_\ell + \mathcal{C})$$

avec $\ell = q^{n-k} - 1$. Supposons qu'un vecteur $\mathbf{y} \in \mathbb{F}_q^n$ soit reçu ; il appartient à l'un des cosets ci dessus, disons $(\mathbf{x_i} + \mathcal{C})$. Il existe donc un mot $\mathbf{c} \in \mathcal{C}$ tel que $\mathbf{y} = \mathbf{x_i} + \mathbf{c}$. Soit $\mathbf{c}'$ le mot transmis,

l'erreur est donc $\mathbf{e} = \mathbf{y} - \mathbf{c}' = \mathbf{x_i} + \mathbf{c} - \mathbf{c}'$, c'est-à-dire qu'elle appartient à $(\mathbf{x_i} + \mathcal{C})$. Une stratégie de décodage à maximum de vraisemblance consiste donc à retrouver le coset d'un mot reçu y, puis à trouver le vecteur e de poids minimal dans ce coset, appelé « *coset leader* » .

La façon la plus simple d'identifier le coset dans lequel se trouve un mot, consiste à calculer son syndrome :

Définition 3.1.9 (Syndrome) *Soit H une matrice de parité d'un code $\mathcal{C}$ sur $\mathbb{F}_q$ de longueur n et de dimension k. Le syndrome d'un mot $\mathbf{x} \in \mathbb{F}_q$ est le vecteur*

$$\mathbf{s} = H\mathbf{x}^T$$

où $\mathbf{x}^T$ dénote le transposé de $\mathbf{x}$. Celui-ci est un vecteur colonne de longueur $n - k$.

Par définition de la matrice de parité, le syndrome d'un mot $\mathbf{x}$ est nul si et seulement si $\mathbf{x}$ est un mot de code. Si $\mathbf{y} = \mathbf{x} + \mathbf{e}$ avec $\mathbf{x} \in \mathcal{C}$, alors

$$s = H\mathbf{y}^T = H\mathbf{x}^T + H\mathbf{e}^T = H\mathbf{e}^T,$$

le syndrome d'un vecteur caractérise donc son erreur et est un invariant de coset. En effet, deux vecteurs $\mathbf{x}$ et $\mathbf{y}$ sont dans le même coset si et seulement si $(\mathbf{x} - \mathbf{y}) \in \mathcal{C}$, c'est-à-dire que $H(\mathbf{x} - \mathbf{y})^T = 0$ et donc que $H\mathbf{x}^T = H\mathbf{y}^T$. Le stockage des « coset leaders » dans une table indexée par leur syndrome permet donc (quand il est possible) de décoder un mot en erreur.

3.2 Difficulté des problèmes de décodage d'un code aléatoire

3.2.1 Code aléatoire

Nous désignerons par code aléatoire un code linéaire dont les k lignes linéairement indépendantes de la matrice génératrice (ou les n colonnes linéairement indépendantes de la matrice de parité) ont été générées aléatoirement, chaque élément étant supposé uniformément réparti dans $\mathbb{F}_q$. On supposera généralement, bien que cela n'ait jamais été prouvé, que de tels codes satisfont la borne de Gilbert-Varshamov (théorème 3.1.2).

Pour $q > 2$, il est montré que le nombre moyen de mots de poids t est proche de

$$\binom{n}{t}(q-1)q^{n-k}.$$

Malheureusement, malgré ces propriétés intéressantes, il n'existe pas à l'heure actuelle d'algorithme général réellement efficace permettant le décodage dans un code aléatoire, comme nous le verrons section 3.2.3. Dans la suite de ce mémoire, nous noterons $\mathcal{B}_{n,k}$ l'ensemble des codes binaires aléatoires de longueur n et de dimension k.

3.2.2 Problèmes de décodage dans un code aléatoire

Le problème décisionnel associé au décodage par syndrome dans un code aléatoire, que nous désignons dans la suite par DSD, a été montré NP-complet en 1978 par E. Berlekamp, R. J. McEliece et H. van Tilborg dans [BMv78] ; le problème calculatoire du décodage par syndrome dans un code aléatoire (désigné par CSD) est donc NP-difficile. Or, l'équivalence entre les problèmes de décodage pas syndrome et les problèmes de décodage borné a été montrée par Y.X. Li, R.H. Deng et X.M. Wang dans [LDW94]. Nous énonçons donc les problèmes suivants :

Problème DBD (Décodage borné décisionnel – Decisional Bounded decoding)
Entrée : (G, x, t) où G est une matrice binaire aléatoire $n \times k$ génératrice d'un code $\mathcal{C}_G$ de dimension k, x est un mot aléatoire de $\mathbb{F}_2^n$ et t est un entier positif.
Question : existe-t-il un mot d'erreur $e \in \mathbb{F}_2^n$ tel que $wt(e) \leq t$ et $(x + e) \in \mathcal{C}$?

Problème DSD (Décodage par syndrome décisionnel – Decisional Syndrome decoding)
Entrée : (H, s, t) où H est une matrice binaire aléatoire $(n - k) \times n$, matrice de parité d'un code $\mathcal{C}_H$ de dimension k, s est un vecteur aléatoire de $\mathbb{F}_2^{n-k}$ et t est un entier positif.
Question : existe-t-il un mot $x \in \mathbb{F}_2^n$ tel que $wt(e) \leq t$ et $Hx^T = s$?

Problème CBD (Décodage borné calculatoire – Computational Bounded Decoding)
Entrée : (G, x, t) où G est une matrice binaire aléatoire $k \times n$ génératrice d'un code $\mathcal{C}_G$ de dimension k, x est un mot aléatoire de $\mathbb{F}_2^n$ et w est un entier positif.
Sortie : un mot d'erreur $e \in \mathbb{F}_2^n$ tel que $wt(e) \leq t$ et $(x + e) \in \mathcal{C}_G$.

Problème CSD (Décodage par syndrome calculatoire – Computational Syndrome Decoding)
Entrée : (H, s, t) où H est une matrice binaire $(n-k) \times n$ aléatoire, matrice de parité d'un code $\mathcal{C}_H$ de dimension k, s est un vecteur aléatoire de $\mathbb{F}_2^{n-k}$ et t est un entier positif.
Sortie : un mot $x \in \mathbb{F}_2^n$ tel que $wt(e) \leq t$ et $Hx^T = s$.

3.2.3 Algorithmes de décodage dans un code aléatoire

Décodage par ensemble d'information

Actuellement, l'algorithme le plus efficace de décodage d'un code aléatoire est le « *décodage par ensemble d'information* » . Il existe plusieurs variantes de cet algorithme, décrit sous une forme simple par R. J. McEliece dans le rapport technique introduisant la cryptographie fondée sur les codes correcteurs d'erreurs [McE78]. De nouvelles variantes de l'algorithme ont par la suite été introduites, en 1988 par J. Léon [Leo88] puis par P. Lee et E. Brickell [LB88], en 1989 par J. Stern [Ste89], en 1994 par A. Canteaut et H. Chabanne [CC94], en 1998 par A. Canteaut et F. Chabaud [CC98] puis par A. Canteaut et N. Sendrier

[CS98], en 2008 par D. Bernstein, T. Lange et C. Peters [BLP08] et finalement en 2009 par M. Finiasz et N. Sendrier [FS09].

Tous ces algorithmes fonctionnent sur le même principe : la sélection d'un ensemble d'information pour la mise sous forme systématique de la matrice définissant le code ; puis la recherche de mots de petit poids dans la redondance. Chacune des variantes ci-dessus propose d'optimiser l'une ou l'autre de ces deux étapes. L'algorithme que nous présentons ici n'est pas à proprement parler un algorithme de décodage mais un algorithme de recherche d'un mot de petit poids dans un code. Cependant, il est possible de décoder un code linéaire en trouvant un mot de petit poids dans un code légèrement plus grand. En effet, si $\mathcal{C}$ est un code $[n, k, d]$ sur $\mathbb{F}_2$ et que $\mathbf{y} \in \mathbb{F}_2^n$ est à distance t d'un mot du code $\mathbf{c} \in \mathcal{C}$ alors $\mathbf{y} - \mathbf{c}$ est un mot de poids t du code $\mathcal{C} + \{\mathbf{y}\}$ (de dimension $k + 1$). Inversement, si $t < d$, alors un élément $e \in \mathcal{C} + \{\mathbf{y}\}$ de poids t ne peut appartenir à $\mathcal{C}$, ce qui signifie que $\mathbf{y} - \mathbf{e} \in \mathcal{C}$ est à distance t de y.

L'algorithme que nous décrivons ici a deux entrées :

– un entier $0 \leq t$,

– une matrice G de taille $k \times n$ génératrice d'un code $[n, k, d \geq 2t + 1]$ sur $\mathbb{F}_2$;

il admet également plusieurs paramètres p, l et σ qui seront optimisés par la suite.

Soit $\mathcal{N} = \{1, \ldots, n\}$ l'ensemble des indices des colonnes $G = (G_i)_{i \in \mathcal{N}}$ où G_i est la i^{eme} colonne de G. Pour tout ensemble $\mathcal{I} \subset \mathcal{N}$, on note $G = (A, B)_{\mathcal{I}}$ la décomposition de G selon $\mathcal{I}$, c'est-à-dire que $A = (G_i)_{i \in \mathcal{I}}$ et $B = (G_i)_{i \in \mathcal{N} \backslash \mathcal{I}}$. Un *ensemble d'information* est un ensemble $\mathcal{I}$ tel que $G' = (\mathcal{I}_k, A)_{\mathcal{I}}$ est équivalente à G ; $\mathcal{J} = \mathcal{N} \backslash \mathcal{I}$ est alors appelé *ensemble de redondance*.

L'algorithme proposé par P. Lee et E. Brickell [LB88] consiste à sélectionner aléatoirement à chaque itération un ensemble d'information $\mathcal{I}$ et à examiner tous les mots de code de poids au plus p. Ces mots correspondent aux combinaisons d'au plus p lignes de la matrice génératrice $(\mathcal{I}_k, A)_{\mathcal{I}}$ sous forme systématique. Cependant, calculer les $\binom{k}{p}$ combinaisons linéaires de $n - k$ bits est une opération coûteuse ; J. Léon a proposé dans [Leo88] de ne calculer dans un premier temps les combinaisons linéaires que pour un petit ensemble $\mathcal{S}$ de σ positions de l'ensemble de redondance. Si un mot de petit poids est ainsi trouvé, les combinaisons linéaires complètes sont alors calculées et examinées.

L'algorithme de Stern [Ste89] fonctionne de façon légèrement différente. Il sélectionne également un ensemble d'information $\mathcal{I}$, qu'il divise en deux sous-ensembles $\mathcal{I}_1$ et $\mathcal{I}_2$ de taille respective $\lfloor \frac{k}{2} \rfloor$ et $\lceil \frac{k}{2} \rceil$ et un sous-ensemble $\mathcal{S}$ de taille σ de l'ensemble de redondance. L'algorithme recherche un mot $\mathbf{c}$ vérifiant

$$\begin{cases} wt(\mathbf{c}_{|\mathcal{I}_1}) = wt(\mathbf{c}_{|\mathcal{I}_1}) = p \\ wt(\mathbf{c}_{|\mathcal{S}}) = 0 \\ wt(\mathbf{c}_{|\mathcal{J} \backslash \mathcal{S}}) \leq t - 2p \end{cases} \tag{3.1}$$

où $\mathbf{c}_{|\mathcal{I}}$ désigne la restriction du mot $\mathbf{c}$ aux seules positions dans $\mathcal{I}$.

Les mots vérifiant cette condition peuvent être calculés de la façon suivante à partir d'une

matrice génératrice sous forme systématique $G = (\mathcal{I}_k, A)$:

1. séparer les lignes de A en deux sous-ensembles A_1 et A_2 correspondant respectivement à $\mathcal{I}_1$ et $\mathcal{I}_2$;
2. sélectionner un sous-ensemble $\mathcal{S}$ de σ éléments de $\mathcal{J}$;
3. pour chaque combinaison linéaire Λ_1 de p lignes de A_1, calculer $\Lambda_{1|\mathcal{S}}$;
4. pour chaque combinaison linéaire Λ_2 de p lignes de A_2, calculer $\Lambda_{2|\mathcal{S}}$;
5. si $\Lambda_{1|\mathcal{S}} = \Lambda_{2|\mathcal{S}}$, vérifier que $wt((\Lambda_1 + \Lambda_2)_{|\mathcal{J}\setminus\mathcal{S}}) \leq t - 2p$; dans ce cas $\Lambda_1 + \Lambda_2$ vérifie la condition (3.1).

$$G = \left(\begin{array}{c|c||c|c} I_1 & 0 & S_1 & A_1 \\ \hline 0 & I_2 & S_2 & A_2 \end{array}\right)$$

FIGURE 3.5 – Selection des ensembles dans l'algorithme de Stern

Tous ces algorithmes sélectionnent à chaque itération un ensemble d'information et effectuent une élimination gaussienne sur une matrice de taille $k \times n$. Cette étape est en réalité la plus coûteuse de l'algorithme, c'est pourquoi A. CANTEAUT a proposé — tout d'abord avec H. CHABANNE [CC94] pour l'algorithme de Lee et Brickel puis avec F. CHABAUD [CC98] pour l'algorithme de Stern — de réduire le coût de l'élimination gaussienne en ne modifiant qu'une seule position de l'ensemble d'information à chaque itération. Le coût de la transition entre deux itérations est ainsi fortement réduit ; et même si le nombre total d'itérations augmente, pour certains codes, le coût moyen de l'algorithme obtenu est au final moindre que celui de l'algorithme original. Par la suite, D. BERNSTEIN, T. LANGE et C. PETERS ont proposé dans [BLP08] un compromis en échangeant non plus une seule position de l'ensemble d'information et de l'ensemble de redondance, mais plusieurs positions de chacun de ces deux ensembles. Ils ont également proposé de rechercher les collisions partielles des combinaisons linéaires sur plusieurs fenêtres L_i de l'ensemble de redondance au lieu d'une seule. Ces modifications, ajoutées à diverses techniques limitant les répétions de calculs ont permis d'atteindre de meilleures performances que celles obtenues jusque là.

Borne inférieure sur le coût du décodage par ensemble d'information Les récentes améliorations de l'algorithme de Stern proposées dans [CC98] et [BLP08] permettent d'en diminuer la complexité moyenne ; malheureusement ces mêmes améliorations rendent son évaluation bien plus difficile. Pour y parvenir, la progression de l'algorithme est modélisée par un processus Markovien et des paramètres de plus en plus nombreux sont à optimiser. De plus ces évaluations se placent du point de vue du cryptanalyste, qui souhaite une estimation de la complexité moyenne de l'algorithme, voire une estimation de la complexité dans le pire des cas. Pour

sa part, le cryptographe souhaite que le décodage soit le plus difficile possible ; une estimation du coût minimal de ces algorithmes est donc pour lui pertinente.

A Asiacrypt 2009, M. Finiasz et N. Sendrier ont proposé dans [FS09] une généralisation de l'algorithme de Stern et proposent une évaluation du coût minimal des algorithmes de décodage par ensemble d'information.

Théorème 3.2.1 ([FS09]) *Pour deux paramètres donnés p et ℓ, le coût minimal $WF_{ISD}(n,r,t)$ d'un algorithme de décodage par ensemble d'information agissant sur une matrice H_0 de taille $r \times n$, matrice de parité d'un code de longueur n et de dimension k sur $\mathbb{F}_2$ pour trouver un mot de $\mathcal{C}_0$ dont le poids est inférieur à t est donné par les formules suivantes :*

- *Si $\binom{n}{t} < 2^r$ ou si $\binom{n}{t} > 2^r$ et $\binom{r}{t-p}\binom{k}{p} \ll 2^r$, alors*

$$WF_{ISD}(n,r,t) \approx \min_p \frac{2\ell \min(\binom{n}{t}, 2^r)}{\lambda \binom{r-\ell}{t-p}\sqrt{\binom{k+\ell}{p}}} \text{ avec } \ell = \log(K_{w-p}\sqrt{\binom{k}{p}}) \text{ et } \lambda = 1 - e^{-1} \approx 0,63 \ ;$$

- *Si $\binom{n}{t} > 2^r$ et $\binom{r}{t-p}\binom{k}{p} \geq 2^r$, alors*

$$WF_{ISD}(n,r,t) \approx \min_p \frac{2\ell 2^{r/2}}{\sqrt{\binom{r-\ell}{t-p}}} \text{ avec } \ell \approx \log K_{w-p} \frac{2^{r/2}}{\sqrt{\binom{r}{t-p}}}$$

où K_{w-p} est le coût moyen d'une vérification $wt(s+(e_1+e_2)H^T) = w-p$ ($s \in \mathbb{F}_2^r$, $e_1 \in \mathcal{W}_{k+\ell,\lfloor p/2 \rfloor}$, $e_2 \in \mathcal{W}_{k+\ell,\lceil p/2 \rceil}$ et $H \equiv H_0$).

Décodage par paradoxe des anniversaires

Le paradoxe des anniversaires apporte également une solution aux problèmes de décodage, décrite et analysée par M. Finiasz et N. Sendrier dans [FS09]. Considérons une instance de SD pour une matrice de parité H de taille $r \times n$, un syndrome s et un poids t. Si le poids t est pair, séparons les colonnes de H en deux ensembles de même taille H_1 et H_2 tels que $H = (H_1|H_2)$. Construisons alors les ensembles $\mathcal{L}_1 = \left\{H_1 e_1^T | e_1 \in \mathcal{W}_{n/2,t/2}\right\}$ et $\mathcal{L}_2 = \left\{s + H_2 e_2^T | e_2 \in \mathcal{W}_{n/2,t/2}\right\}$. Les éléments communs à $\mathcal{L}_1$ et $\mathcal{L}_2$ sont tels que $H_1 e_1^T = s + H_2 e_2^T$, c'est-à-dire tels que $e_1 + e_2$ soit solution de notre instance de SD. La probabilité que l'une des solutions se sépare en deux parties égales de la matrice de parité est $\Pr_{n,t} = \frac{\binom{n/2}{t/2}^2}{\binom{n}{t}}$. Pour parvenir à résoudre le problème SD, il faudra donc répéter ces opérations $1/\Pr_{n,t}$ sur différentes permutations du code original.

Cette attaque a été généralisée par M. Finiasz et N. Sendrier pour construire l'algorithme de décodage présenté figure 3.6. Pour des paramètres fixés n, r, et t, cet algorithme utilise trois variables à optimiser : un entier ℓ et deux ensembles $\mathcal{W}_1$ et $\mathcal{W}_2$ de mots de poids constants. L'algorithme travaille autant que possible sur des syndromes de taille partielle $\ell < r$ et n'effectue la comparaison complète sur r bits uniquement en cas de correspondance partielle.

Entrées : H_0, une matrice de taille $r \times n$, un syndrome $s \in \mathbb{F}_2^r$ et un entier t

Données : un entier $\ell < r$, $W_1 \subset \mathcal{W}_{n,\lfloor t/2 \rfloor}$, $W_2 \subset \mathcal{W}_{n,\lceil t/2 \rceil}$, une table de hachage Λ

tant que *aucun mot n'a été trouvé* **faire**
  $P \xleftarrow{R} \mathcal{P}erm$;
  $H \leftarrow H_0 P$;
  pour chaque $e \in \mathcal{W}_1$ **faire**
    $i \leftarrow \mathsf{msb}(He^T)$;
    $\Lambda(i) \leftarrow \Lambda(i) \cup \{e\}$;
  fin
  pour chaque $e_2 \in \mathcal{W}_2$ **faire**
    $i \leftarrow \mathsf{msb}(s + He_2^T)$;
    pour chaque $e_1 \in \Lambda(i)$ **faire**
      si $He_1^T = s + He_2^T$ **alors retourner** $(e_1 + e_2)P^T$
    fin
  fin
fin

$\mathcal{P}erm$ désigne l'ensemble des matrices de permutations de taille $n \times n$ et $\mathsf{msb}_\ell(x)$ désigne les ℓ premiers bits d'un vecteur $x \in \mathbb{F}_2^r$.

FIGURE 3.6 – Algorithme de décodage par paradoxe des anniversaires [FS09]

Théorème 3.2.2 ([FS09]) *Le coût d'exécution $WF_{BPD}(n,r,t)$ de l'algorithme de décodage par paradoxe des anniversaires (figure 3.6) exécuté sur une matrice de taille $r \times n$ et un poids t est borné par*

$$WF_{BPD}(n,r,t) \geq \sqrt{2}L\log(K_0 L) \text{ où } L = \min\left(\sqrt{\binom{n}{w}}, 2^{r/2}\right)$$

et K_0 est le coût du test $He_1^T = s + He_2^T$.

De la même façon que les attaques sur les fonctions de hachage avaient été étendues par D. WAGNER en construisant plusieurs listes puis en utilisant le paradoxe des anniversaires successivement sur des couples de liste [Wag02], M. FINIASZ et N. SENDRIER étendent l'attaque précédente en une version généralisée. Son coût peut être évalué par la formule donnée par le théorème 3.2.3.

Théorème 3.2.3 ([FS09]) *Le coût d'exécution $WF_{GBPD}(n,r,t)$ de l'algorithme de décodage par paradoxe des anniversaires généralisé exécuté sur une matrice de taille $r \times n$ et un poids t est borné par*

$$WF_{GBPD}(n,r,t) \geq \frac{r-a}{a} 2^{\frac{r-a}{a}}, \text{ avec } a \text{ tel que } \frac{1}{2^a}\binom{n}{\frac{2w}{2^a}} = 2^{\frac{r-a}{a}}.$$

Deuxième partie

Chiffrement fondé sur les codes correcteurs d'erreurs

Chapitre 4

Chiffrement fondé sur la théorie des codes correcteurs d'erreurs

Sommaire

4.1 Codes de Goppa et difficulté algorithmique

Nous avons vu section 3.2 que la difficulté algorithmique des problèmes de décodage sur un code aléatoire est, dans le cas général, bien établie. Pour un code de longueur n, de dimension k et de matrice de parité G aléatoire , si les paramètres n et k sont correctement choisis, la fonction $f : m \mapsto mG + e$ où $e \xleftarrow{R} \mathcal{W}_{n,t}$ est donc à sens unique. Cependant, pour un usage dans le cadre d'un schéma de chiffrement il est nécessaire d'introduire une trappe permettant au destinataire de décoder $f(m)$ afin de recouvrer l'information transmise m, ce qui est impossible en utilisant un code aléatoire. Les codes de Goppa permettent d'atteindre cet objectif.

4.1.1 Codes de Goppa

Les codes de Goppa ont été introduits par V. D. Goppa en 1970 [Gop70]. Initialement étudiés pour leurs propriétés de codes correcteurs d'erreurs — ce qui permit l'apparition d'algorithmes de décodage efficaces [Mas69, Pat75] — ils ont été ensuite étudiés pour leurs propriétés cryptographiques avec l'apparition du cryptosystème de McEliece [McE78].

Les codes de Goppa sont des codes linéaires sur un corps fini $\mathbb{F}_q$, mais leur construction passe par une extension $\mathbb{F}_{q^m}$: ce sont des codes trace d'un code de Reed-Solomon généralisé (voir [MS77] par exemple). Un code de Goppa $\Gamma(g, \mathcal{L})$ est défini par un polynôme g de degré t à coefficients dans $\mathbb{F}_q^m$ et son support $\mathcal{L} = \{\alpha_0, \ldots, \alpha_{n-1}\}$ de n éléments de $\mathbb{F}_q^m$ qui ne sont pas racines de g. La matrice de parité de $\Gamma(g, \mathcal{L})$ est obtenue à partir de la matrice suivante :

$$H_{aux} = \begin{pmatrix} \frac{1}{g(\alpha_0)} & \cdots & \frac{1}{g(\alpha_{n-1})} \\ \vdots & & \vdots \\ \frac{\alpha_0^{t-1}}{g(\alpha_0)} & \cdots & \frac{\alpha_{n-1}^{t-1}}{g(\alpha_{n-1})} \end{pmatrix}$$

Chaque élément de cette matrice est ensuite décomposé en m éléments de $\mathbb{F}_q$ placés en colonnes, en utilisant une projection de $\mathbb{F}_{q^m}$ dans $\mathbb{F}_q^m$. On passe ainsi d'une matrice de taille $t \times n$ sur $\mathbb{F}_{q^m}$ à une nouvelle matrice de parité H de taille $mt \times n$ sur $\mathbb{F}_q$, laquelle définit $\Gamma(g, \mathcal{L})$. C'est donc un code de longueur n et de dimension $k \geq n - mt$; de plus, sa distance minimale est supérieure à $t+1$. En effet, H_{aux} s'écrit comme le produit d'une matrice de Vandermonde et d'une matrice diagonale inversible :

$$H_{aux} = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & & \vdots \\ \alpha_0^{t-1} & \cdots & \alpha_{n-1}^{t-1} \end{pmatrix} \times \begin{pmatrix} \frac{1}{g(\alpha_0)} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{g(\alpha_{n-1})} \end{pmatrix},$$

et donc toute sous-matrice carrée $t \times t$ de H_{aux} est inversible ; ainsi, il n'existe pas de mot de code de poids inférieur ou égal à t. Les codes de Goppa sont donc des codes $[n, k \geq n - mt, d \geq t + 1]$ sur $\mathbb{F}_q$.

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

Dans la suite de ce mémoire, nous noterons $\mathcal{G}_{m,t}$ l'ensemble des codes de Goppa binaires de longueur $n = 2^m$ ayant un polynôme générateur de degré t (de dimension minimale $k = n - mt$), nous désignerons le code de Goppa de support $\mathcal{L}$ et de polynôme générateur g par $\Gamma(\mathcal{L}, g)$ et l'ensemble des mots de longueur n et de poids t par $W_{n,t}$.

4.1.2 Problèmes difficiles

En considérant l'objectif de construire une fonction à sens unique fondée sur la théorie des codes correcteurs d'erreurs, les codes de Goppa binaires sont particulièrement intéressants. En effet, il est calculatoirement difficile de distinguer un code de Goppa binaire de rendement proche de 1/2 dont on ne connaît ni le support ni le polynôme générateur d'un code aléatoire de même longueur et de même dimension. Cela signifie intuitivement que la difficulté du décodage, dans un code de Goppa binaire de rendement proche de 1/2 dont on ne connaît ni le support ni le polynôme générateur, ne sera pas fondamentalement différente de celle du décodage dans un code aléatoire, pour un même jeu de paramètres. C'est pourquoi, nous énonçons le problème suivant introduit dans [CFS01, Sen02] :

Problème GD (Distinguabilité des codes de Goppa – Goppa Distinguishing)
Entrée : $\mathcal{C}_H$, un code de dimension k et de longueur $n = 2^m$ H décrit soit par une matrice génératrice G binaire de taille $k \times n$, soit par une matrice de parité H binaire de taille $(n - k) \times n$.
Question : $\mathcal{C}_H$ appartient-il à $\mathcal{G}_{m,\frac{n-k}{m}}$?

La longueur n, la dimension k et la distance minimum $d = 2t + 1$ d'un code de Goppa sont liées par la relation $t = \frac{n-k}{\log_2 n}$; nous considérons donc les problèmes DGPSD, DGPBD, CGPSD, CGPBD ci-dessous, versions spécialisées des problèmes DSD, DBD, CSD, CBD respectivement ; les équivalences entre les problèmes de décodage borné et de décodage par syndrome sont donc préservées et le problème DGPSD a été prouvé NP-complet par M. Finiasz dans [Fin04].

Problème DGPBD (Décodage borné décisionnel paramétré pour les codes de Goppa – Decisional Goppa parametrized bounded decoding)
Entrée : (G, x) où G est une matrice binaire $k \times n$ aléatoire, matrice de parité d'un code $\mathcal{C}_H$ de dimension k et de longueur $n = 2^m$ et x est un vecteur aléatoire de $\mathbb{F}_2^n$
Question : existe-t-il un mot d'erreur $e \in \mathbb{F}_2^n$ tel que $wt(e) \leq \frac{n-k}{\log_2 n}$ et $x + e \in \mathcal{C}_G$?

Problème DGPSD (Décodage par syndrome décisionnel paramétré pour les codes de Goppa – Decisional Goppa parametrized syndrome decoding)

Entrée : (H, s) où H est une matrice binaire $(n-k) \times n$ aléatoire, matrice de parité d'un code $\mathcal{C}_H$ de dimension k et de longueur $n = 2^m$ et s est un vecteur aléatoire de $\mathbb{F}_2^{n-k}$

Question : existe-t-il un mot $x \in \mathbb{F}_2^n$ tel que $wt(e) \leq \frac{n-k}{\log_2 n}$ et $Hx^T = s$?

Problème CGPBD (Décodage borné calculatoire paramétré pour les codes de Goppa – Computational Goppa Parametrized Bounded Decoding)

Entrée : (G, x) où G est une matrice binaire $k \times n$ aléatoire, matrice de parité d'un code $\mathcal{C}_H$ de dimension k et de longueur $n = 2^m$ et x est un vecteur aléatoire de $\mathbb{F}_2^n$

Sortie : un mot d'erreur $e \in \mathbb{F}_2^n$ tel que $wt(e) \leq \frac{n-k}{\log_2 n}$ et $x + e \in \mathcal{C}_G$.

Problème CGPSD (Décodage par syndrome calculatoire paramétré pour les codes de Goppa – Computational Goppa Parametrized Syndrome Decoding)

Entrée : (H, s) où H est une matrice binaire $(n-k) \times n$ aléatoire, matrice de parité d'un code $\mathcal{C}_H$ de dimension k et de longueur $n = 2^m$ et s est un vecteur aléatoire de $\mathbb{F}_2^{n-k}$

Sortie : un mot $x \in \mathbb{F}_2^n$ tel que $wt(e) \leq \frac{n-k}{\log_2 n}$ et $Hx^T = s$

4.1.3 Modélisation par expériences de difficulté

Afin de modéliser la difficulté d'un problème, il peut être pratique de décrire celui-ci sous la forme d'une expérience de sécurité. Cette expérience met en situation un algorithme (une machine de Turing) résolvant le problème considéré auquel un challenger fournit des entrées du problème tirées aléatoirement. La difficulté du problème est alors exprimée à l'aide de la probabilité de succès, en fonction du temps d'exécution de l'algorithme lors de cette expérience. Nous décrivons ici les expériences de sécurité et la terminologie correspondants aux problèmes que nous utilisons dans la suite de ce manuscrit pour prouver la sécurité des protocoles étudiés.

Tout d'abord, l'efficacité d'un algorithme permettant de résoudre le problème GD, appelé un *distingueur*, peut être estimée à l'aide de l'expérience $\mathsf{Exp}^r_{\mathsf{GD}(m,t)}(\mathcal{D})$ représentée figure 4.1. On définit alors l'avantage du distingueur $\mathcal{D}$ sur GD comme étant la quantité

$$\mathtt{Adv}_{\mathsf{GD}(m,t)}(\mathcal{D}) = \left| \Pr[\mathsf{Exp}^0_{\mathsf{GD}(m,t)}(\mathcal{D}) = 1] - \Pr[\mathsf{Exp}^1_{\mathsf{GD}(m,t)}(\mathcal{D}) = 1] \right|.$$

On dira que le problème est (ϵ, τ)-difficile si quelque soit le distingueur $\mathcal{D}$ s'exécutant en temps au plus τ, $\mathtt{Adv}_{GD(m,t)}(\mathcal{D}) \leq \epsilon(\tau)$.

Ensuite, les jeux $\mathsf{Exp}^r_{\mathsf{DGPBD}(m,t)}(\mathcal{D})$ et $\mathsf{Exp}^r_{\mathsf{DGPSD}(m,t)}(\mathcal{D})$ figure 4.2 permettent d'évaluer respectivement les difficultés des problèmes DGPBD et DGPSD. Au cours de ces expériences, les algorithmes considérés (appelés distingueurs) tentent de décider si le mot est décodable ou non.

Entrées : Un distingueur $\mathcal{D}$
Données : $n = 2^m$, $k \leq n$ et un bit r
Sorties : Un bit $b \in \{0, 1\}$
si $r = 0$ **alors**
  $G \xleftarrow{R} \mathcal{B}_{n,k}$;
sinon
  $G \xleftarrow{R} \mathcal{G}_{m,\frac{n-k}{m}}$;
fin
$b \leftarrow \mathcal{D}(G)$;
retourner b;

FIGURE 4.1 – Expériences de sécurité pour le problème GD : $\mathsf{Exp}^r_{\mathsf{GD}(m,t)}(\mathcal{D})$

Entrées : Un distingueur $\mathcal{D}$
Données : $n = 2^m$, $k = 2^m - mt$ et un bit r
Sorties : Un bit $b \in \{0, 1\}$
$G \xleftarrow{R} \mathcal{B}_{n,k}$;
si $r = 0$ **alors**
  $x \xleftarrow{R} \mathbb{F}_2^k$;
  $e \xleftarrow{R} \{x \in \mathbb{F}_2^n | wt(x) \leq t\}$;
  $c \leftarrow xG + e$;
sinon
  $c \xleftarrow{R} \mathbb{F}_2^n$;
fin
$b \leftarrow \mathcal{D}(G, c)$;
retourner b;

(a) $\mathsf{Exp}^r_{\mathsf{DGPBD}(m,t)}(\mathcal{D})$

Entrées : Un distingueur $\mathcal{D}$
Données : $n = 2^m$, $k = 2^m - mt$ et un bit r
Sorties : Un bit $b \in \{0, 1\}$
$H \xleftarrow{R} \mathcal{B}_{mt,2^m}$;
si $r = 0$ **alors**
  $e \xleftarrow{R} \{x \in \mathbb{F}_2^n | wt(x) \leq t\}$;
  $s = Hx^T$
sinon
  $s \xleftarrow{R} \mathbb{F}_2^{mt}$;
fin
$b \leftarrow \mathcal{D}(H, s)$;
retourner b;

(b) $\mathsf{Exp}^r_{\mathsf{DGPSD}(m,t)}(\mathcal{D})$

FIGURE 4.2 – Expériences de difficulté pour DGPBD et DGPSD

On définit alors l'*avantage du distingueur* $\mathcal{D}$ à résoudre le problème DGPxD avec $\mathsf{x} \in \{\mathsf{B}, \mathsf{S}\}$ comme étant la quantité

$$\mathsf{Adv}_{\mathsf{DGPxD}(m,t)}(\mathcal{D}) = \left| \Pr[\mathsf{Exp}^r_{\mathsf{DGPxD}(m,t)}(\mathcal{D}) = r] - \frac{1}{2} \right|.$$

On dira que le problème DGPxD est (ϵ, τ)-difficile si quelque soit le distingueur $\mathcal{D}$ s'exécutant en temps au plus τ, $\mathsf{Adv}_{\mathsf{DGPxD}(m,t)}(\mathcal{D}) \leq \epsilon(\tau)$.

Enfin, on peut modéliser les difficultés des problèmes CGPBD et CGPSD à l'aide des expériences (appelées ici jeux) présentées figure 4.3. On définit alors le succès $\mathtt{Succ}_{\mathsf{CGPxD}(m,t)}(\mathcal{D})$

Entrées : Un décodeur $\mathcal{D}ec$
Données : $n = 2^m$, $k = 2^m - mt$
$G \xleftarrow{R} \mathcal{B}_{n,k}$;
$x \xleftarrow{R} \mathbb{F}_2^n$;
$e \leftarrow \mathcal{D}ec(G, x)$;
si $wt(x) \leq \frac{n-k}{\log_2 n}$ **alors**
  $\mathcal{D}ec$ remporte le jeu
sinon
  $\mathcal{D}ec$ perd le jeu
fin

(a) Jeu CGPBD(m,t)

Entrées : Un décodeur $\mathcal{D}ec$
Données : $n = 2^m$, $k = 2^m - mt$
$H \xleftarrow{R} \mathcal{B}_{mt,2^m}$;
$s \xleftarrow{R} \mathbb{F}_2^{n-k}$;
$x \leftarrow \mathcal{D}ec(H, s)$;
si $Hx^T = s$ ***et*** $wt(x) \leq \frac{n-k}{\log_2 n}$ **alors**
  $\mathcal{D}ec$ remporte le jeu
sinon
  $\mathcal{D}ec$ perd le jeu
fin

(b) Jeu CGPSD(m,t)

FIGURE 4.3 – Jeux de difficulté pour CGPBD et CGPSD

d'un décodeur $\mathcal{D}$ à résoudre le problème CGPxD avec $\mathsf{x} \in \{\mathsf{B}, \mathsf{S}\}$ comme étant la probabilité que celui-ci remporte le jeu CGPxD. On dira que le problème CGPxD est (ϵ, τ)-difficile si quelque soit le décodeur $\mathcal{D}$ s'exécutant en temps au plus τ, $\mathtt{Succ}_{\mathsf{CGPxD}(m,t)}(\mathcal{D}) \leq \epsilon(\tau)$.

4.2 Schéma de McEliece

Le premier schéma de chiffrement utilisant la théorie des codes a été proposé en 1978 par R. J. McEliece [McE78], dans les premières années de la cryptographie à clef publique. Sa sécurité repose en grande partie sur la difficulté du problème CGPBD.

Définition 4.2.1 (Cryptosystème de McEliece) *Le cryptosystème de McEliece est défini par les quatre algorithmes suivants :*

- *McEliece.Initialiser*(1^κ)

  Déterminer les paramètres m et t nécessaires pour assurer une sécurité adéquate. Renvoyer $\mathcal{P} = (m, t)$.

- *McEliece.GénérerClef*$(\mathcal{P})$

  1. *Tirer aléatoirement un code $\mathcal{C} \xleftarrow{R} \mathcal{G}_{m,t}$.*
  2. *Calculer une matrice de parité G du code $\mathcal{C}$.*
  3. *Construire un algorithme Δ de décodage à distance bornée t pour le code $\mathcal{C}$:* $\forall x \in \mathbb{F}_2^k,\ \forall e \in W_{n,t},\ \Delta(xG + e) = (x, e)$.
  4. *Renvoyer le couple de clef publique/clef privée $(PK, SK) = (G, \Delta)$.*

– *McEliece.Chiffrer*$(\mathcal{P}, PK = G, m \in \mathbb{F}_2^k)$

1. *Tirer aléatoirement* $e \xleftarrow{R} W_{n,t}$, *une erreur de longueur* n *et* de poids t.
2. *Calculer et renvoyer* $c = mG + e$. $(c \in \mathbb{F}_2^n)$

– *McEliece.Déchiffrer*$(\mathcal{P}, SK = \Delta, c \in \mathbb{F}_2^n)$

1. *Calculer* $(x, e) \leftarrow \Delta(c)$.
2. *Renvoyer* x.

Une propriété de la fonction de chiffrement définie par R. McEliece peut être observée immédiatement : pour un code de Goppa de taille $n = 2^m$ et de dimension $k = n - mt$ l'application de chiffrement est une application de $\mathbb{F}_2^k$ vers $\mathbb{F}_2^n$, c'est-à-dire que la taille des chiffrés est plus grande que celle des messages. Le chiffrement de McEliece n'offre donc pas un taux de transmission optimal puisque celui-ci est seulement de $\frac{k}{n} = 1 - \frac{mt}{2^m}$. De plus, cette même observation nous permet de constater que la fonction de chiffrement n'est pas surjective : il existera donc des éléments de $\mathbb{F}_2^n$ qui ne correspondront à aucun chiffré. Ce dernier fait est particulièrement important puisque d'une part il rend plus délicate l'utilisation du schéma de McEliece lors de la construction d'autres protocoles et, d'autre part, il intervient dans la réduction de sécurité.

4.2.1 Sécurité réductionniste

Nous établissons maintenant la sécurité réductionniste du schéma de McEliece. Nous montrons tout d'abord que le schéma n'offre pas la sécurité sémantique puisqu'il est possible de déterminer simplement quel message a été chiffré parmi deux. Or, il existe un certain nombre de conversions (certaines génériques, d'autres spécifiques au schéma de McEliece) permettant d'obtenir la sécurité sémantique du schéma. Une rapide étude de ces constructions nous permet de déterminer le niveau de sécurité pour lequel nous établissons nos preuves par la suite.

Sécurité sémantique

On peut vérifier que le schéma de McEliece, bien qu'il soit probabiliste, ne présente pas la propriété d'indistinguabilité. Supposons que l'on dispose d'un message c_i que l'on sait être le chiffré de l'un des deux messages m_0 ou m_1 à l'aide du schéma de McEliece avec la clef publique G (la clef privée correspondante Δ est quand à elle inconnue). Il est alors possible de calculer la quantité $c_i \oplus m_0 G$. Deux cas se présentent :

– Si c_i est un chiffré de m_0 alors il existe une erreur e de poids t telle que $c_i = m_0 G + e$ et donc $c_i \oplus m_0 G = e$. Il s'ensuit que si c_i est un chiffré de m_0, alors $c_i \oplus m_0 G$ est de poids t.
– Si c_i est un chiffré de m_1 alors il existe donc une erreur e' de poids t telle que $c_i = m_1 G + e'$ et donc $c_i \oplus m_0 G = (m_0 + m_1)G + e'$. Comme $(m_0 + m_1)G$ est un mot du code de Goppa défini par G et de distance minimale $2t+1$, le poids de $c_i \oplus m_0 G$ est donc d'au moins $t+1$.

Une attaque exploitant cette observation est particulièrement simple à mettre en œuvre puisqu'il suffit de calculer un produit matriciel (nk additions bits à bits), une addition binaire de deux mots de longueurs n et de calculer le poids de celui-ci pour distinguer les chiffrés de deux messages avec une clef publique donnée. On peut donc écrire le théorème suivant :

Théorème 4.2.1 *Le schéma de McEliece n'est pas IND-CPA.*

Conversions sémantiquement sûres Dans [KI01], K. Kobara et H. Imai examinent les conversions génériques permettant d'atteindre le niveau de sécurité IND-CCA2 (voir section 2.3.1) à partir de cryptosystèmes atteignant un niveau de sécurité plus faible.

Ils montrent en particulier que ni la conversion OAEP (Optimal Asymmetric Encryption Padding) proposée par M. Bellare et P. Roggaway [BR95] ni la conversion de E. Fujisaki et T. Okamoto présentée dans [FO99a] ne s'appliquent au cryptosystème de McEliece . En effet, la première convertit toute permutation à sens unique en un cryptosystème IND-CCA2 tandis que la seconde permet de passer d'un cryptosystème IND-CPA à un schéma IND-CCA2. Or, le cryptosystème de McEliece n'est ni une permutation, ni IND-CPA (Théorème 4.2.1).

En revanche, ils montrent également que les constructions proposées par D. Pointcheval dans [Poi00] d'une part et par E. Fujisaki et T. Okamoto dans [FO99b] d'autre part s'appliquent au cryptosystème de McEliece. La première permet la conversion d'une fonction *partiellement à sens unique* — ce qui est le cas du cryptosystème de McEliece — en un cryptosystème IND-CCA2 ; la seconde permet la construction d'un schéma de chiffrement IND-CCA2 à l'aide d'une fonction à sens unique (ce qui inclut les permutations à sens unique et les fonction partiellement à sens unique).

Cependant, ces deux constructions nécessitent une forte expansion des chiffrés ; ce phénomène est encore accentué par la grande taille des blocs chiffrés à l'aide du cryptosystème de McEliece (voir section 4.2.2). C'est pourquoi K. Kobara et H. Imai ont introduit dans [KI01] une conversion spécifique au schéma de McEliece permettant d'atteindre le niveau de sécurité IND-CCA2 dans le modèle de l'oracle aléatoire tout en limitant l'expansion des chiffrés, sous l'hypothèse que la fonction de chiffrement de McEliece soit à sens unique. La première conversion du cryptosystème de McEliece dans le modèle standard permettant d'atteindre la sécurité sémantique a été proposée en 2008 par R. Nojima, H. Imai, K. Kobara et K. Morozov [NIKM08], mais celle-ci est sémantiquement sûre uniquement dans le contexte d'une attaque à clairs choisis (modèle de sécurité IND-CPA). Enfin en 2009, R. Dowsley, J. Müller-Quade et A. Nascimento ont proposé dans [DMQN09] une conversion du schéma de McEliece IND-CCA2 dans le modèle standard.

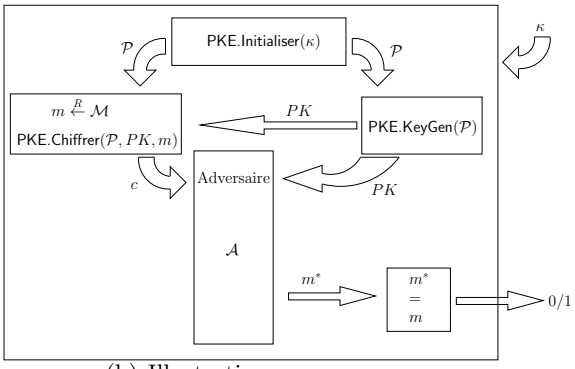
Même si elles s'appuient sur des techniques différentes, toutes ces constructions ont en commun de requérir uniquement le caractère à sens unique de la fonction de chiffrement (OW-CPA).

Sens unique

La propriété de sécurité minimale que doit vérifier tout cryptosystème à clef publique est d'être à sens unique lors d'une attaque à clairs choisis (OW-CPA) tel que nous l'avons décrit section 2.3.1. Dans ce modèle, un adversaire $\mathcal{A}$ reçoit en entrée une clef publique PK générée par le challenger en utilisant l'algorithme de génération des clefs ainsi que le chiffré c d'un message m choisi aléatoirement à l'aide de l'algorithme public et de la clef PK. Au cours de l'attaque, l'adversaire est libre de calculer le chiffré de tout message de son choix. A la fin de l'attaque, l'adversaire renvoie un message m^* ; son attaque est un succès si $m^* = m$. Cette situation peut être modélisée à l'aide d'un jeu décrit sur la figure 4.4. On appellera *succès d'un*

Entrées : Un paramètre de sécurité κ
Données : Un adversaire $\mathcal{A}$
$\mathcal{P} \leftarrow \mathsf{PKE.Initialiser}(1^\kappa)$;
$(SK, PK) \leftarrow \mathsf{PKE.GenererClefs}(\mathcal{P})$;
$m \xleftarrow{R} \mathcal{M}$;
$c \leftarrow \mathsf{PKE.Chiffrer}(\mathcal{P}, PK, m)$;
$m^* \leftarrow \mathcal{A}(PK, c)$;
si $m^* = m$ **alors**
  $\mathcal{A}$ remporte le jeu
sinon
  $\mathcal{A}$ perd le jeu
fin

(a) Présentation algorithmique, point de vue du challenger



(b) Illustration

FIGURE 4.4 – Jeu de sécurité OW-CPA(κ)

adversaire $\mathcal{A}$ lors d'une attaque à clairs choisis contre le schéma PKE la probabilité

$$\mathtt{Succ}^{\mathsf{PKE}}_{OW-CPA(\kappa)}(\mathcal{A}) = \Pr[\mathcal{A} \text{ remporte le jeu OW-CPA}(\kappa)]$$

et on dira que le schéma PKE est (ϵ, τ)-OW-CPA si pour tout adversaire $\mathcal{A}$ fonctionnant en temps τ, $\mathtt{Succ}^{\mathsf{PKE}}_{OW-CPA(\kappa)}(\mathcal{A}) \leq \epsilon(\tau)$.

Théorème 4.2.2 (Le schéma de McEliece est OW-CPA) *Si les problèmes CGPBD(m,t), DGPBD(m,t) et GD(m,t) où $(m,t) \leftarrow$ McEliece.Initialiser(1^κ) sont respectivement (ϵ_{CGPBD}, τ), (ϵ_{CGPBD}, τ) et (ϵ_{GD}, τ)-difficiles alors le cryptosystème de McEliece est (ϵ, τ)-OW-CPA avec*

$$\epsilon(\tau) = \epsilon_{CGPBD}(\tau) + \epsilon_{DGPBD}(\tau) + \epsilon_{GD}(\tau).$$

PREUVE – La preuve établie ici est une preuve par jeu selon la méthodologie introduite par V. SHOUP dans [Sho04] et présentée section 2.1.3. Cette méthodologie est utilisée ici car elle permet d'identifier clairement les différents facteurs intervenant dans la réduction de sécurité. On présente ici une séquence de jeux Jeu 0, ..., Jeu n dont le premier est le jeu OW-CPA(κ) (figure 4.5(a)) contre le schéma de McEliece et le dernier est le jeu de difficulté associé au problème CGPBD (figure 4.5(b)). On notera $\Pr[S_i]$ la probabilité que l'adversaire remporte le jeu i.

Soient $\mathcal{A}$ un adversaire contre le schéma de McEliece fonctionnant en temps τ et Jeu 0 le jeu OW-CPA(κ) adapté au schéma de McEliece. Par définition, on a

$$\Pr[S_0] = \mathsf{Succ}^{\mathsf{McEliece}}_{OW-CPA(\kappa)}(\mathcal{A}).$$

Entrées : Un paramètre de sécurité κ
Données : Un adversaire $\mathcal{A}$

1 $(m,t) \leftarrow \mathsf{McEliece.Initialiser}(1^\kappa)$;
2 $(G,\Delta) \leftarrow \mathsf{McEliece.GenererClefs}(\mathcal{G}_{m,t})$;
3 $m \xleftarrow{R} \mathbb{F}_2^k$;
4 $e \xleftarrow{R} W_{n,t}$;
5 $c = mG + e$;
6 $m^* \leftarrow \mathcal{A}(G,c)$;
7 **si** $m^* = m$ **alors**
8 | $\mathcal{A}$ remporte le jeu
9 **sinon**
10 | $\mathcal{A}$ perd le jeu
11 **fin**

(a) Jeu 0 : Jeu de sécurité OW-CPA(κ) adapté au schéma de McEliece

Entrées : Un paramètre de sécurité κ
Données : Un adversaire $\mathcal{A}$

1 $(m,t) \leftarrow \mathsf{McEliece.Initialiser}(1^\kappa)$;
2 $G \leftarrow \mathcal{B}_{2^m,2^m-mt}$;
3 $c \xleftarrow{R} \mathbb{F}_2^{2^m}$;
4 $m^* \leftarrow \mathcal{A}(G,c)$;
5 **si** $wt(c - m^*G) \leq t$ **alors**
6 | $\mathcal{A}$ remporte le jeu
7 **sinon**
8 | $\mathcal{A}$ perd le jeu
9 **fin**

(b) Jeu 3 : $\mathcal{A}$ est utilisé pour résoudre CGPBD

FIGURE 4.5 – Premier et dernier jeux de la réduction de sécurité du schéma de McEliece

Jeu 1 – L'objectif de ce premier jeu est de ne plus faire intervenir le message m, utilisé lors de la construction du chiffré, dans l'étape de vérification afin de préparer la transition du Jeu 3 au Jeu 4. Ce jeu fonctionne comme le Jeu 0 précédent excepté la condition $m^* = m$, c'est-à-dire la vérification que le message m^* produit par l'attaquant est le même que le message m utilisé pour construire le chifré $c = mG + e$ avec $e \xleftarrow{R} \{x \in \mathbb{F}_2^n | wt(x) = t\}$. Celle-ci est remplacée par la vérification de la propriété $wt(c + m^*G) \leq t$; dans ce cas l'adversaire remporte le jeu.

De façon évidente, si $m = m^*$ alors $c + m^*G = e$ et la condition de poids est vérifiée. Inversement, si $wt(c + m^*G) \leq t$ alors $c + m^*G = e$ et donc $m = m^*$. En effet, en posant $e^* = c + m^*G$, on a $e + e^* = mG + m^*G \in \mathcal{C}$. Or $wt(e + e^*) \leq 2t$, ce qui est inférieur à la distance minimale du code défini par G, et donc $e + e^* = \mathbf{0}$. De même, si $m \neq m^*$ alors $c + m^*G = mG + m^*G + e$ et donc $wt((m + m^*)G) \leq 2t$, or $(m + m^*)G \in \mathcal{C}$.

Les deux conditions utilisées dans Jeu 0 et Jeu 1 sont donc équivalentes et par suite,

$$\Pr[S_1] = \Pr[S_0].$$

Jeu 2 – Dans ce jeu, la sélection d'une matrice génératrice d'un code de Goppa binaire aléatoire en guise de clef publique est remplacée par la sélection aléatoire d'une matrice génératrice d'un code binaire quelconque de même taille.

Soit $\mathcal{D}$ le distingueur $\mathcal{D}$ présenté figure 4.6(a) et considérons l'expérience $\mathsf{Exp}^r_{\mathsf{GD}(2^m,t)}(\mathcal{D})$ présentée figure 4.1 section 4.1.3, rappelé figure 4.6(b). Si G est une matrice génératrice d'un code

Entrées : Une matrice génératrice G d'un code binaire de longueur 2^m et de dimension $2^m - mt$
Données : Un adversaire $\mathcal{A}$ contre McEliece
Sorties : Un bit b

$m \xleftarrow{R} \mathbb{F}_2^k$;
$e \xleftarrow{R} \{x \in \mathbb{F}_2^n | wt(x) = t\}$;
$c = mG + e$;
$m^* \leftarrow \mathcal{A}(G, c)$;
si $wt(c + m^*G) \leq t$ **alors**
  retourner 1
sinon
  retourner 0
fin

(a) Distingueur $\mathcal{D}$ construit Jeu 2

Entrées : Un distingueur $\mathcal{D}$
Données : $n = 2^m$, $k \leq n$ et un bit r
Sorties : Un bit $b \in \{0, 1\}$

si $r = 0$ **alors**
  $G \xleftarrow{R} \mathcal{B}_{2^m, 2^m - mt}$;
sinon
  $G \xleftarrow{R} \mathcal{G}_{m, \frac{n-k}{m}}$;
fin
$b \leftarrow \mathcal{D}(G)$;
retourner b;

(b) $\mathsf{Exp}^r_{\mathsf{GD}(m,t)}(\mathcal{D})$

FIGURE 4.6 – Distingueur $\mathcal{D}$ construit Jeu 2 et expérience de sécurité

de Goppa, alors $\mathcal{D}$ se comporte comme Jeu 1 et donc $\Pr[\mathsf{Exp}^0_{\mathsf{GD}(m,t)}(\mathcal{D}) = 1] = \Pr[S_1]$. De même si G est une matrice génératrice d'un code binaire aléatoire, alors $\mathcal{D}$ se comporte comme Jeu 2 et donc $\Pr[\mathsf{Exp}^1_{\mathsf{GD}(m,t)}(\mathcal{D}) = 1] = \Pr[S_2]$. Il s'ensuit que $\mathtt{Adv}_{\mathsf{GD}(m,t)}(\mathcal{D}) = |\Pr[S_2] - \Pr[S_1]|$. Or,

par hypothèse, GPSD est $(\epsilon_{\mathsf{GD}}, \tau_{\mathsf{GD}})$-difficile et donc

$$|\Pr[S_2] - \Pr[S_1]| \leq \epsilon_{\mathsf{GD}}(\tau).$$

Jeu 3 – Dans ce jeu, le calcul du chiffré c, à partir d'un message et d'une erreur de poids t tous deux tirés aléatoirement, est remplacé par le tirage aléatoire de c dans $\mathbb{F}_2^n$.

Soit $\mathcal{D}$ le distingueur présenté figure 4.7(a). Considérons l'expérience 4.2(a) présentée section 4.1.3 et rappelée figure 4.7(b). Si c est un mot construit en tirant aléatoirement $m \xleftarrow{R} \mathbb{F}_2^k$

Entrées : Une matrice génératrice G d'un code binaire de longueur 2^m et de dimension $2^m - mt$, un mot $c \in \mathbb{F}_2^n$
Données : Un adversaire $\mathcal{A}$ contre McEliece
Sorties : Un bit b

$m^* \leftarrow \mathcal{A}(G, c)$;
si $wt(c + m^*G) \leq t$ **alors**
  retourner 1
sinon
  retourner 0
fin

(a) Distingueur $\mathcal{D}$ construit Jeu 3)

Entrées : Un distingueur $\mathcal{D}$
Données : $n = 2^m$, $k = 2^m - mt$ et un bit r
Sorties : Un bit $b \in \{0, 1\}$

$G \xleftarrow{R} \mathcal{B}_{2^m, 2^m - mt}$;
si $r = 0$ **alors**
  $x \xleftarrow{R} \mathbb{F}_2^k$;
  $e \xleftarrow{R} \{x \in \mathbb{F}_2^n | wt(x) \leq t\}$;
  $c \leftarrow xG + e$;
sinon
  $c \xleftarrow{R} \mathbb{F}_2^n$;
fin
$b \leftarrow \mathcal{D}(G, c)$;
retourner b;

(b) $\mathsf{Exp}^r_{\mathsf{DGPBD}(m,t)}(\mathcal{D})$

FIGURE 4.7 – Distingueur $\mathcal{D}$ construit Jeu 3 et expérience de sécurité

et $e \xleftarrow{R} \{x \in \mathbb{F}_2^n | wt(x) = t\}$, alors $\mathcal{D}$ se comporte comme Jeu 2 et donc $\Pr[\mathsf{Exp}^0_{GPBD_d(m,t)}(\mathcal{D}) = 1] = \Pr[S_2]$. De même, si c est tiré aléatoirement dans $\mathbb{F}_2^n$ alors $\mathcal{D}$ se comporte comme Jeu 3 et $\Pr[\mathsf{Exp}^1_{\mathsf{DGPBD}(m,t)}(\mathcal{D}) = 1] = \Pr[S_3]$. Comme par hypothèse DGPBD est $(\epsilon_{\mathrm{GPBD_d}}, \tau_{\mathrm{GPBD_d}})$-difficile, on a

$$|\Pr[S_3] - \Pr[S_2]| \leq \epsilon_{\mathsf{DGPBD}}(\tau)$$

De plus ce jeu est exactement le jeu de difficulté associé à CGPBD puisque $e^* = c + m^*G$ est de poids t et que $c + e^* = m^*G$ est nécessairement un mot du code défini par G. Comme par hypothèse CGPBD est $(\epsilon_{\mathsf{CGPBD}}, \tau_{\mathsf{CGPBD}})$-difficile, on a

$$\Pr[S_3] \leq \epsilon_{\mathsf{CGPBD}}(\tau)$$

Conclusion – On obtient de la séquence de jeu précédente les inégalités suivantes :

1. $\Pr[S_0] = \mathtt{Succ}^{\mathsf{McEliece}}_{OW-CPA(\kappa)}(\mathcal{A})$,
2. $\Pr[S_1] = \Pr[S_0]$,
3. $|\Pr[S_2] - \Pr[S_1]| \leq \epsilon_{\mathsf{GD}}(\tau)$,
4. $|\Pr[S_3] - \Pr[S_2]| \leq \epsilon_{\mathsf{DGPBD}}(\tau)$,
5. $\Pr[S_3] \leq \epsilon_{\mathsf{CGPBD}}(\tau)$.

Par inégalité triangulaire, on obtient donc

$$\mathtt{Succ}^{\mathsf{McEliece}}_{OW-CPA(\kappa)}(\mathcal{A}) \leq \epsilon_{\mathsf{CGPBD}}(\tau) + \epsilon_{\mathsf{DGPBD}}(\tau) + \epsilon_{\mathsf{GD}}(\tau)$$

$\diamondsuit$

Il est intéressant de remarquer l'influence de DGPBD (la version décisionnelle du problème CGPBD) dans la réduction. Celle-ci est due au fait que le problème CGPBD est défini pour un mot aléatoire de l'espace $\mathbb{F}_2^n$ alors que les chiffrés sont, par définition, des mots de $\mathbb{F}_2^n$ situés à distance au plus t d'un mot du code définit par la clef publique. De plus, la proportions de ces mots de $\mathbb{F}_2^n$ à distance au plus t d'un mot du code est en général faible.

Il est donc interessant de relier la sécurité du cryptosystème de McEliece à une version de CGPBD ne considérant que des mots décodables. Nous appellons ce problème RCGPBD, pour *Restricted Computational Goppa Parametrized Bounded Decoding* ; la figure 4.8 représente le jeu de sécurité $\mathsf{RCGPBD}(m,t)$ associé. Concrètement, les prescriptions de sécurité pour le cryptosystème de McEliece se font toujours en considérant la difficulté du problème RCGPBD, sans tenir compte de la possibilité qu'un mot ne soit pas décodable [CC94, BLP08, FS09]. Il est immédiat de constater que ce problème se réduit à CGPBD et DGPBD.

Problème RCGPBD (Décodage borné restreint paramétré pour les codes de Goppa – Restricted Goppa parametrized bounded decoding)

Entrée : (G, x) où G est une matrice binaire $k \times n$ aléatoire, matrice de génératrice d'un code $\mathcal{C}_H$ de longueur $n = 2^m$ et de dimension $k = 2^m - mt$ et $x = yG + e$ où y est un vecteur aléatoire de $\mathbb{F}_2^k$ et e une erreur de poids inférieur à t

Sortie : un mot d'erreur $e \in \mathbb{F}_2^n$ tel que $wt(e) \leq t$ et $x + e \in \mathcal{C}_G$.

On établit alors le théorème suivant :

Théorème 4.2.3 (Le schéma de McEliece est OW-CPA) *Si les problèmes $\mathsf{RCGPBD}(m,t)$ et $\mathsf{GD}(m,t)$ où $(m,t) \leftarrow \mathsf{McEliece.Initialiser}(1^\kappa)$ sont respectivement $(\epsilon_{\mathsf{RCGPBD}}, \tau)$, et $(\epsilon_{\mathsf{GD}}, \tau)$-difficiles alors le cryptosystème de McEliece est (ϵ, τ)-OW-CPA avec*

$$\epsilon(\tau) = \epsilon_{\mathsf{RCGPBD}}(\tau) + \epsilon_{\mathsf{GD}}(\tau).$$

Entrées : m et t
Données : Un décodeur $\mathcal{D}ec$
$G \xleftarrow{R} \mathcal{B}_{n=2^m,k=2^m-mt}$;
$y \xleftarrow{R} \mathbb{F}_2^k$ $e \xleftarrow{R} \{x \in \mathbb{F}_2^n | wt(x) \leq t\}$;
$c \leftarrow yG + e$;
$e^* \leftarrow \mathcal{D}ec(H, s)$;
si $c + e^* \in \mathcal{C}$ ***et*** $wt(e) \leq t$ **alors**
  $\mathcal{D}ec$ remporte le jeu
sinon
  $\mathcal{D}ec$ perd le jeu
fin

FIGURE 4.8 – Jeu de difficulté $\mathsf{RCGPBD}(m, t)$

PREUVE (Aperçu) – La preuve de ce théorème est identique à celle du théorème 4.2.2 à l'exception du dernier jeu qui n'est alors plus nécessaire, le Jeu 2 étant directement le jeu $\mathsf{RGPBD}(m, t)$. ♢

4.2.2 Performances

Taille des clefs et des chiffrés

Les paramètres $m = 10$ et $t = 50$ initialement préconisés par R. MCELIECE sont clairement insuffisants comme le montrent les attaques de A. CANTEAUT et F. CHABAUD [CC98] et de D. BERNSTEIN, T. LANGE, et C. PETERS de complexités respectives $2^{64,2}$ et $2^{60,5}$.

Dans [FS09], M. FINIASZ et N. SENDRIER utilisent les bornes qu'ils décrivent pour évaluer la sécurité du cryptosystème de McEliece instancié à l'aide de codes de Goppa pour des jeux de paramètres classiques. Un code de Goppa de longueur $n = 2^m = 2^{11}$ et de capacité de correction $t = 32$ offrira une sécurité de $2^{59,9}$; un code de Goppa de longueur $n = 2^m = 2^{12}$ et de capacité de correction $t = 41$ offrira une sécurité de $2^{128,5}$.

La clé publique est la matrice de parité d'un code de Goppa de longueur $n = 2^m$ et de capacité de correction t (de dimension $k = n - mt$). Celle-ci nécessite le stockage d'une matrice de taille $k \times n$ soit de $2^{2m} - mt2^m$ bits. La clé privée est quant à elle la connaissance d'un algorithme de décodage efficace pour ce même code. Celui-ci est fourni par la connaissance de la structure interne du code, c'est-à-dire de son support (n éléments de $\mathbb{F}_2^m$) et de son polynôme générateur (un polynôme de degré t à valeurs dans $\mathbb{F}_2^m$) soit $2^m + t$ éléments de m bits. Les chiffrés, qui sont des éléments de $\mathbb{F}_2^n$, nécessitent donc n bits.

Ces tailles sont grandes d'un point de vue cryptographique puisqu'une sécurité de 2^{80}

nécessite des clefs publiques de 424 Ko, des clefs privées d'environ 2,79 Ko ; celles-ci permettent de chiffrer des blocs de 1696 bits sur 2048 bits. A titre de comparaison, une sécurité de 2^{80} est atteinte en utilisant le chiffrement RSA [RSA78] avec des clefs (publiques et privées) de 1024 bits sur des blocs de même taille.

Efficacité calculatoire

L'opération de chiffrement réalisée dans le cryptosystème de McEliece consiste en

- une multiplication d'un message de k bits par une matrice binaire de taille $k \times n$ soit k additions de n bits,
- la sélection d'une erreur de poids t,
- son addition avec le produit précédemment obtenu, soit une nouvelle addition de n bits.

Si le cryptosystème de McEliece utilise des clefs de grande taille, le chiffrement ne nécessite donc aucune opération réellement coûteuse. Le déchiffrement cependant n'est pas aussi aisé, même si son coût reste raisonnable. En effet, celui-ci nécessite d'effectuer

- le produit d'un mot de longueur n par une matrice de taille $n \times n$ soit n additions de n bits,
- le décodage d'un mot de longueur n dans le code $\mathcal{C}_0$ en un mot de longueur k,
- le produit du résultat de ce décodage par une matrice de taille $k \times k$ soit k additions de k bits.

Les tailles et les coûts des algorithmes sont explicités dans la table 4.2 page 70.

4.3 Variante de Niederreiter

En 1986, H. Niederreiter a proposé une variante duale du cryptosystème de McEliece [Nie86]. Il utilise comme clef publique une matrice de parité, les message sont alors des motifs d'erreurs de poids fixé et les chiffrés sont des syndromes. À l'origine, Niederreiter proposait d'utiliser des codes de Reed-Solomon généralisés, mais V. Sidelnikov et S. Shestakov ont montré en 1992 que l'utilisation de ces codes n'était pas sûre [SS92]. En revanche, il est montré dans [LDW94] que le cryptosystème de Niederreiter utilisé avec le même code de Goppa que celui utilisé dans le cryptosystème de McEliece a la même sécurité que ce dernier. C'est cette version que nous présentons ici. Sa sécurité repose alors sur les problèmes CGPSD et GD.

Définition 4.3.1 (Cryptosystème de Niederreiter) *Le cryptosystème de Niederreiter est défini par les quatre algorithmes suivants :*

- *Niederreiter.Initialiser*(1^κ)

  Déterminer les paramètres m et t nécessaires pour assurer une sécurité adéquate. Renvoyer $\mathcal{P} = (m, t)$

  Pour faciliter la lecture de la suite, on note $n = 2^m$, $k = n - mt$ et $\mathcal{M} = W_{n,t}$.

- *Niederreiter.GénérerClef*$(\mathcal{P})$

  1. *Tirer aléatoirement un code $\mathcal{C} \xleftarrow{R} \mathcal{G}_{m,t}$.*
  2. *Calculer une matrice de parité H du code $\mathcal{C}$.*
  3. *Construire un algorithme Δ de décodage par syndrome à distance t pour le code $\mathcal{C}$: $\forall e \in W_{n,t},\ \Delta(H_0 e^T) = e$.*
  4. *Renvoyer le couple de clef publique/clef privée $(PK, SK) = (H, \Delta)$.*

- *Niederreiter.Chiffrer*$(\mathcal{P}, PK = H, m \in W_{n,t})$

  Calculer et renvoyer $c = Hm^T$. ($c \in \mathbb{F}_2^{n-k}$)

- *Niederreiter.Déchiffrer*$(\mathcal{P}, SK = \Delta, c \in \mathbb{F}_2^{n-k})$

  Calculer et renvoyer $x = \Delta(c)$.

Ce schéma nécessite donc le codage de message en mots de poids constant t. Ceux-ci sont au nombre de $\binom{n}{t}$ et peuvent être associés à un indice. Une méthode permettant de réaliser cela est donnée dans [CFS01]. Dans cette méthode, l'indice d'un mot e de poids t dont les positions des bits non nuls sont $(i_1, i_2, \ldots, i_n)$ est

$$I_e = \binom{i_1}{1} + \binom{i_2}{2} + \cdots + \binom{i_t}{t}.$$

En pratique, ce calcul n'est pas aussi simple qu'il n'y parait puisqu'il fait nécessairement appel à des calculs sur des grands entiers. Pour calculer l'erreur de poids t associée à l'indice I_e deux stratégies peuvent être adoptées.

La méthode la plus simple consiste à d'abord chercher le plus grand indice i_t tel que $\binom{i_t}{t} < I_e$, puis à chercher i_{t-1} tel que $\binom{i_{t-1}}{t-1} < I_e - \binom{i_t}{t}$ et ainsi de suite. Comme souligné dans [Fin04], cette procédure peut être améliorée pour ne pas avoir à effectuer de calcul direct de coefficients binomiaux mais uniquement des divisions de grands entiers [Cov73, Gol66].

Il est également possible de précalculer tous les coefficients binomiaux que l'algorithme risque de rencontrer (c'est-à-dire les $nt = 2^m t$ binomiales $\binom{i}{j}$ pour $i = 0, \ldots, n-1$ et $j = 1, \ldots, t$) et d'effectuer une recherche par dichotomie. Il est ainsi possible de n'utiliser aucune division ou multiplication lors du codage d'un message ou des précalculs (les binomiales peuvent être calculées à l'aide du triangle de Pascal). Cette méthode est donc certainement plus efficace lors du chiffrement de nombreux messages mais nécessite une quantité de mémoire importante.

4.3.1 Sécurité réductionniste

En 1994, Y.X. Li, R.H. Deng et X.M. Wang ont montré dans [LDW94] l'équivalence entre le cryptosystème de Niederreiter et celui de McEliece instancié à l'aide des codes de Goppa. Les théorèmes 4.2.2, 4.2.3 et 4.2.1 s'appliquent donc trivialement au chiffrement de Niederreiter. Ainsi ce dernier atteint le niveau de sécurité OW-CPA sans toutefois atteindre le niveau IND-CPA. Remarquons que l'attaque sur la sécurité sémantique du schéma de Niederreiter est encore plus simple que dans le cas du cryptosystème de McEliece puisqu'il suffit dans ce cas de comparer le chiffré c_r avec le syndrome $s_1 = Hm_1^T$.

Nous avons donc une quantification de la sécurité OW-CPA du schéma de Niederreiter à l'aide de la difficulté des problèmes CGPBD, DGPBD et GD ou de celle des problèmes RCGPBD et GD. Cependant, une réduction de sécurité vers CGPSD, DGPSD et GD semble plus naturelle. Nous établissons donc le théorème suivant :

Théorème 4.3.1 (Le schéma de Niederreiter est OW-CPA) *Si les problèmes* $\mathsf{CGPSD}(m,t)$, $\mathsf{DGPSD}(m,t)$ *et* $\mathsf{GD}(m,t)$ *où* $\mathcal{G}_{m,t} \leftarrow \mathsf{Niederreiter.Initialiser}(1^\kappa)$ *sont respectivement* $(\epsilon_{\mathsf{CGPSD}}, \tau)$, $(\epsilon_{\mathsf{DGPSD}}, \tau)$ *et* $(\epsilon_{\mathsf{GD}}, \tau)$*-difficiles alors le cryptosystème de Niederreiter est* (ϵ, τ)*-OW-CPA avec*

$$\epsilon(\tau) = \epsilon_{\mathsf{CGPSD}}(\tau) + \epsilon_{\mathsf{DGPSD}}(\tau) + \epsilon_{\mathsf{GD}}(\tau).$$

Preuve (Aperçu) – La preuve peut également être établie à l'aide d'une séquence de jeux similaire à celle établie dans la preuve du théorème 4.2.2, nous ne détaillons donc pas ici les différentes transitions. Le jeu de départ Jeu 0 est le jeu OW-CPA(κ) adapté au schéma de Niederreiter représenté figure 4.9(a).

Jeu 1 – Ce premier jeu remplace la condition de vérification $m^* = m$ par les conditions $wt(m^*) \leq t$ et $Hm^{*T} = c$. On a $\Pr[S_1] = \Pr[0]$.

Jeu 2 – La sélection d'une matrice de Goppa binaire aléatoire tenant lieu de clef publique est remplacée par la sélection d'une matrice de parité d'un code binaire aléatoire de même taille. On obtient alors $|\Pr[S_2] - \Pr[1]| \leq \epsilon_{\mathsf{GD}}(\tau)$.

Jeu 3 – Ce jeu remplace le calcul du syndrome d'un mot aléatoire de poids t par la sélection aléatoire d'un élément de $\mathbb{F}_2^{mt}$. On a alors $|\Pr[S_3] - \Pr[2]| \leq \epsilon_{\mathsf{DGPSD}}(\tau)$. Ce jeu est également le jeu de difficulté $\mathsf{CGPSD}(m,t)$ et ainsi $\Pr[S_3] = \epsilon_{\mathsf{CGPSD}}(\tau)$.

De cette série de jeu on obtient donc par inégalité triangulaire la relation donnée dans l'énoncé du théorème. ♢

On constate donc ici également l'influence de la version décisionnelle du problème de décodage considéré. On devra donc de nouveau considérer une version restreinte de ce problème :

Entrées : Un paramètre de sécurité κ

Données : Un adversaire $\mathcal{A}$

1 $\mathcal{G}_{m,t} \leftarrow$ Niederreiter.Initialiser(1^κ);
2 $(H, \Delta) \leftarrow$ Niederreiter.GenererClefs$(\mathcal{G}_{m,t})$;
3 $m \xleftarrow{R} W_{2^m,t}$;
4 $c = Hm^T$;
5 $m^* \leftarrow \mathcal{A}(H, c)$;
6 **si** $m^* = m$ **alors**
7 $\quad$ $\mathcal{A}$ remporte le jeu
8 **sinon**
9 $\quad$ $\mathcal{A}$ perd le jeu
10 **fin**

(a) Jeu de sécurité OW-CPA(κ) adapté au schéma de Niederreiter instancié par les codes de Goppa

Entrées : Un paramètre de sécurité κ

Données : Un adversaire $\mathcal{A}$

1 $\mathcal{G}_{m,t} \leftarrow$ Niederreiter.Initialiser(1^κ);
2 $H \leftarrow \mathcal{B}_{mt,2^m}$;
3 $c \xleftarrow{R} \mathbb{F}_2^{mt}$;
4 $m^* \leftarrow \mathcal{A}(H, c)$;
5 **si** $wt(m^*) \leq t$ *et* $m^* H^T = c$ **alors**
6 $\quad$ $\mathcal{A}$ remporte le jeu
7 **sinon**
8 $\quad$ $\mathcal{A}$ perd le jeu
9 **fin**

(b) Jeu 3 : $\mathcal{A}$ est utilisé pour résoudre CGPSD

FIGURE 4.9 – Premier et dernier jeux de la réduction de sécurité du schéma de niederreiter

Problème RCGPSD (Décodage borné calculatoire restreint paramétré pour les codes de Goppa – Restricted Computational Goppa parametrized bounded decoding)

Entrée : (H, s) où H est une matrice binaire $k \times n$ aléatoire, matrice de parité d'un code $\mathcal{C}_H$ de longueur $n = 2^m$ et de dimension $k = 2^m - mt$ et $s = Hx^T$ où x est un vecteur aléatoire de $\mathbb{F}_2^k$ de poids inférieur à t

Sortie : un mot d'erreur $e \in \mathbb{F}_2^n$ tel que He^T et $wt(e) \leq t$.

On peut alors établir le théorème suivant :

Théorème 4.3.2 (Le schéma de Niederreiter est OW-CPA) *Si les problèmes RCGPSD(m, t) et GD(m, t) où $\mathcal{G}_{m,t} \leftarrow$ Niederreiter.Initialiser(1^κ) sont respectivement $(\epsilon_{RCGPSD}, \tau)$ et (ϵ_{GD}, τ)-difficiles alors le cryptosystème de McEliece est (ϵ, τ)-OW-CPA avec*

$$\epsilon(\tau) = \epsilon_{RCGPSD}(\tau) + \epsilon_{GD}(\tau).$$

PREUVE (Aperçu) – La preuve de ce théorème est sensiblement identique à celle du théorème 4.3.1 à l'exception du dernier jeu qui n'est alors plus nécessaire, Jeu 2 étant directement le jeu RCGPSD(m, t). ♢

Entrées : m et t
Données : Un décodeur $\mathcal{D}ec$
$H \xleftarrow{R} \mathcal{B}_{n=2^m,k=2^m-mt}$;
$x \xleftarrow{R} \{x \in \mathbb{F}_2^n | wt(x) \leq t\}$;
$s \leftarrow Hx^T$;
$e^* \leftarrow \mathcal{D}ec(H, s)$;
si $He^{*T} = s$ ***et*** $wt(e) \leq t$ **alors**
  $\mathcal{D}ec$ remporte le jeu
sinon
  $\mathcal{D}ec$ perd le jeu
fin

FIGURE 4.10 – Jeu de difficulté $\mathsf{RCGPSD}(m, t)$

4.3.2 Performances

Les problèmes GPSD et GPBD ont été montrés équivalents par Y.X. LI, R.H. DENG et X.M. WANG dans [LDW94]. Par conséquent, la taille des codes à utiliser pour assurer la sécurité du cryptosystème de Niederreiter est la même que pour le schéma de McEliece. Cependant, le code public est décrit par une matrice de parité et non par une matrice génératrice. Pour un code de taille $n = 2^m$ et de dimension $k = 2^m - mt$, la matrice publique du cryptosystème de Niederreiter est donc de taille $(n - k) \times n$ (soit $mt2^m$ bits) alors que celle du cryptosystème de McEliece est de taille $k \times n$ (soit $2^{2m} - mt2^m$ bits).

Le cryptosystème de Niederreiter chiffre des messages contenus dans une erreur de poids t. Ceux-ci sont au nombre de $\binom{n}{t} = \binom{2^m}{t}$ et peuvent donc transmettre $\log_2 \binom{n}{t}$ bits d'information. Les chiffrés sont les syndromes associés à cette erreur, c'est-à-dire des vecteurs de $n - k = mt$ bits. Ceux-ci sont donc bien plus courts que les chiffrés produits par le schéma de McEliece (qui sont des vecteurs de $n = 2^m$ bits) ; mais ils permettent également de transmettre moins d'information. Le taux de transmission de ces deux schémas est donc un critère de comparaison plus pertinent.

Le chiffrement en tant que tel est quant à lui beaucoup moins coûteux puisqu'il suffit de réaliser la somme de t vecteurs de $n - k$ bits. Cependant, le codage des messages en mots de poids constant est souvent l'étape la plus coûteuse de l'algorithme. Le déchiffrement a pour sa part un coût très similaire, avec un léger gain sur le produit de matrice puisque les matrices manipulées sont plus petites. En comparaison avec le cryptosystème de McEliece, ce schéma possède donc l'avantage d'avoir des blocs beaucoup plus petits et un chiffrement bien plus rapide sans toutefois perdre beaucoup sur les autres points. Le tableau 4.2 page 70 reprend les

différentes tailles et coûts algorithmiques du schéma.

4.4 Variante de Sendrier & Biswas

En 2008 N. Sendrier et B. Biswas ont proposé dans [BS08] une nouvelle variante du cryptosystème de McEliece qu'ils qualifient d'*hybride*. Cette variante introduit deux modifications du schéma : tout d'abord ils proposent d'utiliser le cryptosystème de McEliece tout en introduisant une partie de l'information à transmettre au sein même de l'erreur qui est ajoutée au mot de code contenant l'essentiel de l'information du message. Ceci permet d'accroître le taux de transmission du schéma de McEliece. Ensuite, ils proposent de réduire la taille des clefs en utilisant un code public décrit par une matrice génératrice sous forme systématique. Ce schéma est décrit en utilisant des codes de Goppa.

Définition 4.4.1 (Cryptosystème de McEliece hybride) *Soit une fonction injective* $\varphi : \{0,1\}^\ell \longrightarrow \mathcal{W}_{n,t} = \{e \in \mathbb{F}^n | wt(e) \leq t\}$ *telle que* φ *et* φ^{-1} *soient facilement calculables et* $\ell = \left\lfloor \binom{n}{t} \right\rfloor$. *Le schéma de chiffrement de McElice hybride est alors défini par les quatre algorithmes suivants :*

- *Hybride.Initialiser*(1^κ)

  1. *Déterminer les paramètres* m *et* t *nécessaires pour assurer une sécurité adéquate.*
  2. *Construire une fonction* φ *injective* $\varphi : \{0,1\}^\ell \longrightarrow \mathcal{W}_{n,t} = \{e \in \mathbb{F}^n | wt(e) \leq t\}$ *telle que* φ *et* φ^{-1} *soient facilement calculables et* $\ell = \left\lfloor \binom{n}{t} \right\rfloor$.
  3. *Renvoyer* $\mathcal{P} = (m, t, \varphi)$.

  Pour faciliter la lecture de la suite, on note $n = 2^m$ *et* $k = n - mt$.

- *Hybride.GénérerClef*$(\mathcal{P})$

  1. *Tirer aléatoirement un code de Goppa* $\Gamma(\mathcal{L}, g) \xleftarrow{R} \mathcal{G}_{m,t}$.
  2. *Calculer une matrice* $k \times (n-k)$ *binaire* R *telle que* $(I_k|R)$ *soit une matrice génératrice de* $\Gamma(\mathcal{L}, g)$.
  3. *Construire un* t*-décodeur* $\Delta_{\mathcal{L},g}$ *pour* $\Gamma(\mathcal{L}, g)$ *:* $\forall x \in \mathbb{F}_2^k,\ \forall e \in W_{n,t},\ \Delta(xG+e) = (x, e)$.
  4. *Renvoyer le couple de clef publique/clef privée* $(PK, SK) = (R, \Delta_{L,g})$.

- *Hybride.Chiffrer*$(\mathcal{P}, PK = G, m = (x, y) \in \mathbb{F}_2^k \times \mathbb{F}_2^\ell)$.

  Calculer et renvoyer $c = (x\|xR) + \varphi(y)$. $(c \in \mathbb{F}_2^n)$

- *Hybride.Déchiffrer*$(\mathcal{P}, SK = \Delta_{\mathcal{L},g}, c \in \mathbb{F}_2^n)$

  1. *Calculer* $(x, e) \leftarrow \Delta_{\mathcal{L},g}(c)$.
  2. *Renvoyer* $(x, \varphi^{-1}(e))$.

4.4.1 Sécurité réductionniste

Il est montré dans [BS08] que le cryptosystème de McEliece est sûr sous les hypothèses que CGPBD et GD soient tous deux difficiles. Cependant, afin de pouvoir comparer les cryptosystèmes, il peut être intéressant d'établir une réduction du cryptosystème hybride à celle du schéma de McEliece original. Nous établissons le théorème suivant :

Théorème 4.4.1 (Le schéma de McEliece hybride est OW-CPA) *Si le schéma de McEliece est (ϵ, τ)-OW-CPA et φ est une fonction injective à valeurs uniformément distribuées, alors le schéma de McEliece hybride est (ϵ', τ')-OW-CPA avec*

$$\epsilon'(\tau') \leq \epsilon(\tau + O(\frac{n^3}{\lambda})) + 1 - \frac{|\mathcal{D}_\varphi|}{\binom{n}{t}}$$

Preuve – Nous établissons une séquence de jeux dont le premier est le jeu OW-CPA(κ) adapté au schéma de McEliece hybride (présenté figure 4.11(a)) et dont le dernier (présenté figure 4.11(b)) est le jeu OW-CPA(κ) adapté au schéma de McEliece pour un nouvel adversaire que nous définissons au cours de la preuve.

Soient $\mathcal{A}$ un adversaire contre le schéma de McEliece hybride et Jeu 0 le jeu OW-CPA(κ) adapté au schéma de McEliece hybride. Par définition, on a

$$\Pr[0] = \mathsf{Succ}^{\mathsf{Hybride}}_{OW-CPA(\kappa)}(\mathcal{A})$$

Jeu 1 – L'objectif de ce jeu est de remplacer l'algorithme de génération de la clef publique afin d'y introduire l'algorithme de génération des clefs du schéma de McEliece original.

Pour cela on introduit un oracle de systématisation Syst qui prend en entrée une matrice G de taille $k \times n$ et renvoie une matrice de permutation $n \times n$ P telle que $GP = (U|V)$ avec U non-singulière. Comme toute matrice génératrice admet un code équivalent sous forme systématique, l'oracle Syst fonctionne donc en temps $O(n^3/\lambda)$ où λ est la proportion des matrices non-singulières de taille $k \times k$.

Cet oracle est alors interrogé sur une matrice G qui est la clef publique fournie par McEliece.GénérerClef($\mathcal{G}_{m,t}$). Il renvoie une matrice de permutation P telle que $GP = (U|V)$. On pose $R = U^{-1}V$ qui est alors fourni à l'adversaire ainsi qu'à l'oracle de chiffrement en guise de clef publique et qui est également utilisé lors de la génération du challenge. Le Jeu 1 résultant est présenté figure 4.12.

Le code $\mathcal{C}$ défini par la matrice $GP = (U|V)$ est un code équivalent au code engendré par G, il est donc lui-même un code de Goppa. De plus, la matrice $(I_k|R) = (I_k|U^{-1}V)$ est une matrice génératrice de $\mathcal{C}$ sous forme systématique et R est donc une clef aléatoire valide pour le cryptosystème de McEliece hybride. On a donc

$$\Pr[S_1] = \Pr[S_0]$$

Entrées : Un paramètre de sécurité κ
Données : Un adversaire $\mathcal{A}$

$(m, t, \varphi) \leftarrow \mathsf{Hybride.Initialiser}(1^\kappa)$;
$(R, \Delta) \leftarrow \mathsf{Hybride.GenererClefs}(m, t)$;
$m \xleftarrow{R} \mathbb{F}_2^{2^m - mt}$;
$r \xleftarrow{R} \mathbb{F}_2^{\ell}$;
$c = (m \| mR) + \varphi(r)$;
$(m^*, r^*) \leftarrow \mathcal{A}(R, c)$;
si $m^* = m$ *et* $r^* = r$ **alors**
  $\mathcal{A}$ remporte le jeu
sinon
  $\mathcal{A}$ perd le jeu
fin

(a) Jeu de sécurité OW-CPA(κ) adapté au schéma de McEliece hybride

Entrées : Un paramètre de sécurité κ
Données : Un adversaire $\mathcal{A}$

$(m, t, \varphi) \leftarrow \mathsf{Hybride.Initialiser}(1^\kappa)$;
$(G, \Delta) \leftarrow \mathsf{McEliece.GenererClefs}(m, t)$;
$m \xleftarrow{R} \mathbb{F}_2^{2^m - mt}$;
$e \xleftarrow{R} \mathcal{W}_{2^m, t}$;
$c' = mG + e$;
$P \leftarrow \mathsf{Syst}(G)$;
$(U|V) \leftarrow GP$;
$R \leftarrow U^{-1}V$;
$c \leftarrow c'P$;
$(m', r') \leftarrow \mathcal{A}'(R, c)$;
$m^* = m'U$;
$r^* = \varphi^{-1}(\varphi(r')P^{-1})$;
si $m^* = m$ *et* $\varphi(r^*) = e$ **alors**
  $\mathcal{A}'$ remporte le jeu
sinon
  $\mathcal{A}'$ perd le jeu
fin

(b) Jeu 3 : $\mathcal{A}$ est utilisé pour attaquer le cryptosystème de McEliece

FIGURE 4.11 – Premier et dernier jeux de la réduction de sécurité du schéma de McEliece hybride

et

$$T_1 = T_0 + O(\frac{n^3}{\lambda})$$

Jeu 2 – Ce jeu substitue le tirage aléatoire d'une erreur e de poids t au calcul de $\varphi(r)$. Il est alors nécessaire d'adapter la condition de succès en vérifiant que si l'adversaire renvoie un couple (m^*, r^*), on ait $m = m^*$ et $\varphi(r^*) = e$. La simulation échoue donc lorsque e n'appartient pas au domaine D_φ de φ, c'est-à-dire avec probabilité $1 - \frac{|\mathcal{D}_\varphi|}{\binom{n}{t}}$, en supposant que la fonction φ renvoie des valeurs uniformément distribuées dans D_φ.

D'après le lemme de la différence (Lemme 2.1.1 page 24), on a donc :

$$|\Pr[S_2] - \Pr[S_1]| \leq 1 - \frac{|\mathcal{D}_\varphi|}{\binom{n}{t}}$$

Entrées : Un paramètre de sécurité κ
Données : Un adversaire $\mathcal{A}$
$(m, t, \varphi) \leftarrow \mathsf{Hybride.Initialiser}(1^\kappa)$;
$(G, \Delta) \leftarrow \mathsf{McEliece.GenererClefs}(\mathcal{G}_{m,t})$;
$P \leftarrow \mathsf{Syst}(G)$;
$(U|V) \leftarrow GP$;
$R \leftarrow U^{-1}V$;
$m \overset{R}{\leftarrow} \mathbb{F}_2^k$;
$r \overset{R}{\leftarrow} \mathbb{F}_2^\ell$;
$c = (m\|mR) + \varphi(r)$;
$(m^*, r^*) \leftarrow \mathcal{A}(R, c)$;
si $m^* = m$ *et* $r^* = r$ **alors**
  $\mathcal{A}$ remporte le jeu
sinon
  $\mathcal{A}$ perd le jeu
fin

FIGURE 4.12 – Jeu 1

Jeu 3 – Ce jeu remplace l'encodage de m à l'aide de la matrice R par l'encodage de m en c' à l'aide de la matrice G lors de la génération du challenge, comme cela est fait dans le schéma de McEliece. Ce mot c' est alors permuté pour obtenir le chiffré c qui sera transmis à l'adversaire. On a donc $c = mGP + eP = m(U|V) + eP = mU(I_k|R) + eP$. Une erreur de poids t restant un mot de poids t par permutation, c est donc un chiffré valide de mU par le schéma de McEliece hybride. On a donc

$$\Pr[S_3] = \Pr[S_2]$$

En cas de succès, l'adversaire $\mathcal{A}$ renvoie alors un couple (m', r'). Ce jeu calcule alors $m^* = m'U^{-1}$ et $r^* = \varphi^{-1}(\varphi(r')P^{-1})$ avant l'étape de vérification. On obtient alors le jeu final présenté figure 4.11(b).

De façon évidente, si l'adversaire remporte ce jeu, alors le nouvel adversaire $\{\mathcal{A}, \mathsf{Syst}\}$ formé par $\mathcal{A}$ et l'oracle de systématisation et les étapes de transformation des challenges et des réponses fournit une réponse valide dans le jeu CGPBD. On a donc

$$\Pr[S_3] = \mathsf{Succ}^{\mathsf{McEliece}}_{OW-CPA(\kappa)}(\{\mathcal{A}, \mathsf{Syst}\}) \leq \epsilon(\tau + O(\frac{n^3}{\lambda}))$$

Conclusion – On a donc

- $\Pr S_1 = \Pr S_0 = \mathtt{Succ}^{\mathsf{Hybride}}_{OW-CPA(\kappa)}(\mathcal{A})$,
- $|\Pr[S_3] - \Pr[S_2]| \leq 1 - \frac{|\mathcal{D}_\varphi|}{\binom{n}{t}}$
- et $\Pr[S_3] = \mathtt{Succ}^{\mathsf{McEliece}}_{OW-CPA(\kappa)}(\{\mathcal{A}, \mathsf{Syst}\}) \leq \epsilon(\tau + O(\frac{n^3}{\lambda}))$,

c'est à dire

$$\mathtt{Succ}^{\mathsf{Hybride}}_{OW-CPA(\kappa)}(\mathcal{A}) \leq \epsilon(\tau + O(\frac{n^3}{\lambda})) + 1 - \frac{|\mathcal{D}_\varphi|}{\binom{n}{t}},$$

ce qui conclue la preuve. $\diamondsuit$

4.4.2 Performances

Les tables 4.1 et 4.2 page 70 permettent de voir que le cryptosystème hybride présente l'avantage, par rapport aux cryptosystèmes de McEliece et de Niederreiter, d'accroître de façon significative le taux de transmission. La réduction de sécurité précédente nous montre que dans le cas d'un codage en mots d'erreurs de poids constant bijectif, l'utilisation du cryptosystème hybride ne diminue pas la sécurité du cryptosystème. il est alors possible d'utiliser des clefs de même taille que celles utilisées pour le cryptosystème de McEliece. Le schéma hybride présentera donc essentiellement les mêmes performances que le cryptosystème de McEliece tout en offrant un meilleur taux de transmission.

Cependant, le codage de l'information en mots d'erreurs de poids constant bijectif peut ralentir de façon significative le chiffrement. Dans ce cas, un codage surjectif peut être utilisé ; la sécurité du schéma est alors légèrement dégradée. Le choix de la fonction de codage surjective est donc un point clé de la solidité de la construction.

4.5 Bilan

Nous avons établi au cours de ce chapitre le niveau de sécurité de trois schémas de chiffrement utilisant les codes de Goppa. Nous avons montré explicitement le lien entre leur sécurité OW-CPA et deux propriétés spécifiques à cette famille de codes : la forte ressemblance entre un code de Goppa et un code aléatoire de même taille, et la difficulté du décodage dans ce dernier. Notre analyse nous conduit à introduire une nouvelle catégorie de problèmes de décodage, plus proche de ceux étudiés en pratique : les problèmes de décodages restreints aux seules instances admettant une solution.

Ces trois schémas résultent de différents compromis. Le chiffrement de McEliece présente un meilleur taux de transmission que le schéma de Niederreiter, mais nécessite des clefs publiques plus imposantes et des blocs chiffrés de plus grande taille. Le schéma de Sendrier & Biswas améliore pour sa part sensiblement le taux de transmission, mais les blocs de message clairs sont de taille légèrement supérieurs à ceux utilisés dans le schéma de McEliece pour une même taille

de clefs. De plus, il peut être nécessaire d'affaiblir (raisonnablement) la sécurité du système pour conserver une vitesse de chiffrement intéressante.

Ces trois schémas partagent également un même inconvénient : la grande taille des clefs qu'ils utilisent. Avant d'envisager une réelle utilisation pratique, il reste donc un problème à résoudre : trouver une famille de codes admettant une représentation plus compacte pour remplacer de façon sûre les codes de Goppa, et admettant si possible une preuve de sécurité réductionniste. Nous examinons dans le chapitre suivant différentes tentatives dans ce sens.

	McEliece		Niederreiter		Hybride	
Sécurité	$2^{86,8}$	$2^{128,5}$	$2^{86,8}$	$2^{128,5}$	$2^{86,8}$	$2^{128,5}$
Paramètres (m,t)	$(11,32)$	$(12,41)$	$(11,32)$	$(12,41)$	$(11,32)$	$(12,41)$
matrice publique (clef publique)	424 Ko	1802 Ko	88 Ko	246 Ko	424 Ko	1802 Ko
messages	1696 bits	3604 bits	233 bits	327 bits	1929 bits	3931 bits
chiffrés	2048 bits	4096 bits	352 bits	492 bits	2048 bits	4096 bits
taux de transmission (arrondi à 10^{-2})	$0,83$	$0,88$	$0,66$	$0,66$	0.94	0.96

TABLE 4.1 – Taille des clefs et taux de transmission des principaux cryptosystèmes fondés sur les codes correcteurs d'erreurs pour des mêmes codes de Goppa de $\mathcal{G}_{m,t}$.

	McEliece	Niederreiter	Hybride
matrice publique (clef publique)	$k \times n$	$(n-k) \times n$	$k \times n$
messages	k bits	$\lfloor \log_2 \binom{n}{t} \rfloor$ bits	$k + \lfloor \log_2 \binom{n}{t} \rfloor$ bits
chiffrés	n bits	$n-k$ bits	n bits
taux de transmission	$\frac{k}{n}$	$\frac{\lfloor \log_2 \binom{n}{t} \rfloor}{n-k}$	$\frac{n+\lfloor \log_2 \binom{n}{t} \rfloor}{n-k}$
chiffrement (en nombre d'opérations binaires)	$n(k+1)$	kt + codage en mots de poids constant	$n(k+1)$ + codage en mots de poids constant
déchiffrement (en nombre d'opérations binaires)	$n^2 + k^2 + t^2(\log_2 n)^3$	$n^2 + (n-k)^2 + t^2(\log_2 n)^3$	$n^2 + k^2 + t^2(\log_2 n)^3$

TABLE 4.2 – Comparaison entre les principaux cryptosystèmes fondés sur les codes correcteurs d'erreurs pour un même code de Goppa de taille $n = 2^m$ et de dimension $k = 2^m - mt$.

Chapitre 5

Cryptanalyses de variantes de McEliece

Ce chapitre a fait l'objet des publications [OTD08a] et [OTD08b], en collaboration avec A. Otmani et J.P. Tillich.

Sommaire

5.1 Réduction de la taille des clefs

Nous avons vu au chapitre précédent que les cryptosystèmes fondés sur la théorie des codes correcteurs d'erreurs nécessitait des clefs de grande taille, de l'ordre de plusieurs centaines de Ko. Cette situation est un frein majeur à l'utilisation pratique de ces cryptosystèmes. La recherche de nouvelles familles de codes permettant d'instancier le cryptosystème de McEliece de façon sûre est un domaine actif.

Pour cela, deux contraintes doivent être respectées par la famille de codes choisie : tout d'abord, il faut que la famille considérée admette un algorithme de décodage pour des paramètres correspondant à un code aléatoire dont le décodage est difficile ; ensuite, il est nécessaire que la structure du code privé puisse être efficacement masquée dans la matrice publique. Si le premier critère est facile à satisfaire, le second en revanche n'est pas toujours évident. Par exemple, V. Sidelnikov et S. Shestakov ont montré en 1992 [SS92] qu'il était possible de retrouver efficacement la structure de codes de Reed-Solomon généralisés, ce qui invalidait la proposition de H. Niederreiter d'utiliser ceux-ci pour instancier sa variante ; N. Sendrier a prouvé qu'une permutation masquant la structure d'un code concaténé peut être extrait d'une clef publique ainsi formée [Sen98] et L. Minder et A. Shokrollahi [MS07] ont montré qu'il existait une attaque structurelle contre un cryptosystème utilisant des codes de Reed-Muller [Sid94].

En 2000, C. Monico, J. Rosenthal et A. Shokrollahi, ont étudié la possibilité de remplacer l'utilisation des codes de Goppa par des codes LDPC [MRS00]. Ces derniers sont définis par une matrice de parité très creuse (c'est-à-dire dont l'essentiel des positions sont à 0) et admettent donc une représentation compacte. Malheureusement, ils mettent en avant les faiblesses du schéma ainsi obtenu. En 2005, P. Gaborit a proposé l'utilisation de codes quasi-cycliques [Gab05], c'est-à-dire définis par une matrice génératrice dont les lignes sont obtenues par une permutation simple de la première ligne. Ces codes admettent donc également une représentation courte. En 2007, M. Baldi et G. Chiaraluce proposèrent de combiner les deux idées en utilisant des codes LDPC quasi-cycliques [BC07]. Nous présentons dans ce chapitre une cryptanalyse de chacun de ces deux derniers cryptosystèmes. Celles-ci ont été réalisées en collaboration avec A. Otmani et J.P. Tillich ; elles ont fait l'objet des publications [OTD08a] et [OTD08b].

5.2 Notations et définitions

Nous introduisons tout d'abord les notations et définitions nécessaires pour présenter les variantes du cryptosystème de McEliece proposées dans [Gab05] et [BC07], ainsi que leurs cryptanalyses.

5.2.1 Matrices circulantes

Une *matrice circulante* M est une matrice $p \times p$ obtenue par décalage circulaire à droite de la ligne précédente, de sorte que la matrice M soit entièrement décrite par sa première ligne $\mathbf{m} = (m_0, m_1, m_{p-1})$. Notons que M peut être également obtenue par décalage circulaire vers le bas de la première colonne.

$$M = \begin{pmatrix} m_0 & m_1 & \cdots & m_{p-1} \\ m_{p-1} & m_0 & \cdots & m_{p-2} \\ \vdots & \vdots & \ddots & \vdots \\ m_1 & m_2 & \cdots & m_0 \end{pmatrix}$$

Notons $\mathbb{F}_2[x]$ l'ensemble des polynômes univariés à coefficients dans $\mathbb{F}_2$. Tout vecteur $\mathbf{v} = (v_0, v_1, \ldots, v_{p-1})$ de $\mathbb{F}_2^p$ peut être identifié au polynôme $\mathbf{v}(x) = v_0 + v_1 x \cdots + v_{p-1}x^{p-1}$; le polynôme d'intersection de deux polynômes $\mathbf{u}(x)$ et $\mathbf{v}(x)$ est $\mathbf{u}(x) \star \mathbf{v}(x) = \sum u_i v_i x^i$.

Ainsi, il est possible de caractériser la i^{eme} ligne d'une matrice circulante M par le polynôme

$$x^i \cdot \mathbf{m}(x) \mod (x^p - 1)$$

et le produit $\mathbf{b} \times M$ d'un vecteur binaire $\mathbf{b} = (b_0, b_1, \ldots, b_{p-1})$ par la matrice M est exactement le vecteur binaire représenté par le polynôme $\mathbf{b}(x) \cdot \mathbf{m}(x) \mod (x^p - 1)$. Cette propriété s'étend naturellement au produit de deux matrices circulantes M et N de même taille $p \times p$: la première ligne de $M \times N$ est $\mathbf{m}(x) \cdot \mathbf{n}(x) \mod (x^p - 1)$ et sa i^{eme} ligne est représentée par

$$\big(x^i \cdot \mathbf{m}(x)\big) \cdot \mathbf{n}(x) \mod (x^p - 1) = x^i \cdot (\mathbf{m}(x) \cdot \mathbf{n}(x)) \mod (x^p - 1).$$

On a alors le résultat suivant :

Proposition 5.2.1 *Soit $\mathfrak{C}_p$ l'ensemble des matrices binaires circulantes de taille $p \times p$; il existe alors un isomorphisme entre les anneaux $(\mathfrak{C}_p, +, \times)$ et $(\mathbb{F}_2[x]/(x^p - 1), +, \cdot))$:*

$$(\mathfrak{C}_p, +, \times) \simeq (\mathbb{F}_2[x]/(x^p - 1), +, \cdot)\,.$$

Remarque 5.2.1 *La première colonne d'une matrice circulante M définie par $\mathbf{m}(x)$ correspond au polynome $\mathbf{m}^\star(x) = x^p \cdot \mathbf{m}(\frac{1}{x}) \mod (x^p - 1)$.*

Remarque 5.2.2 *La matrice identité $p \times p$ est une matrice circulante représentée par le polynôme constant 1.*

La proposition 5.2.1 peut être utilisée pour caractériser de façon simple les matrices circulantes inversibles :

Proposition 5.2.2 *Une matrice circulante $M \in \mathfrak{C}_p$ définie par $\mathbf{m}(x)$ est inversible si et seulement si $\mathbf{m}(x)$ est premier avec $x^p - 1$.*

PREUVE – Pour établir la proposition, il est uniquement nécessaire de prouver que l'inverse d'une matrice circulante M est également circulante. Supposons qu'il existe une matrice N telle que $N \times M = M \times N = \mathcal{I}_p$. Soit $\mathbf{n} = (n_0, \ldots, n_{p-1})$ la première ligne de N ; le produit $\mathbf{n} \times M$ peut alors être vu comme le polynôme $\mathbf{n}(x) \cdot \mathbf{m}(x) \mod (x^p - 1)$, lequel est égal à 1 par hypothèse. Par conséquent, pour tout i tel que $0 \leq i < p$, on a également $\left(x^i \cdot \mathbf{n}(x)\right) \cdot \mathbf{m}(x) = x^i \mod (x^p - 1)$, ce qui prouve que la matrice circulante définie par $\mathbf{n}(x)$ est l'inverse de M. Il s'ensuit que N est circulante. $\diamondsuit$

Une matrice G de taille $k \times n$ avec $k = k_0 p$ et $n = n_0 p$ $(k_0, n_0 > 0)$ est circulante par blocs de taille p s'il existe $k_0 n_0$ matrices circulantes $G_{i,j} \in \mathfrak{C}_p$ telles que

$$G = \begin{pmatrix} G_{1,1} & \ldots & G_{1,n_0} \\ \vdots & & \vdots \\ G_{k_0,1} & \ldots & G_{k_0,n_0} \end{pmatrix}.$$

Il est immédiat de voir que l'ensemble des matrices circulantes par bloc est stable par addition et produit de matrices. Il est par conséquent naturel d'établir une identification entre une matrice circulante G et une matrice de polynômes $\mathbf{G}(x)$ de taille $k_0 \times n_0$ à coefficients dans $\mathbb{F}_2[x]/(x^p - 1)$. Celle-ci est obtenue par l'application qui associe à chaque bloc $G_{i,j}$ de G le polynôme $\mathbf{g}_{i,j}(x)$ qui le définit.

Proposition 5.2.3 *Soient $\mathfrak{B}^p_{k_0,n_0}$ l'ensemble des matrices de taille $k_0 p \times n_0 p$ circulantes par blocs de tailles p, $R_p = \mathbb{F}_2[x]/(x^p - 1)$ et $\mathfrak{M}_{k_0,n_0}(R_p)$ l'ensemble des matrices de taille $k_0 p \times n_0 p$ à coefficients dans R_p. Il existe un isomorphisme d'anneaux entre $\mathfrak{B}^p_{k_0,n_0}$ et $\mathfrak{M}_{k_0,n_0}(R_p)$:*

$$\begin{array}{rcl} \mathfrak{B}^p_{k_0,n_0} & \simeq & \mathfrak{M}_{k_0,n_0}(R_p) \\ G & \mapsto & \mathbf{G}(x) \end{array}$$

En particulier, une matrice G circulante par blocs de taille p est inversible si et seulement si $\det(\mathbf{G})(x)$ est premier avec $x^p - 1$ et son inverse est également une matrice circulante par bloc.

5.2.2 Codes cycliques et quasi-cycliques

Un *code cyclique* $\mathcal{C}$ de longueur n est un idéal de l'anneau $\mathbb{F}_2[x]/(x^p - 1)$. Il est caractérisé par un polynôme unique $\mathbf{g}(x)$ diviseur de $x^p - 1$. Soit r le degré de $\mathbf{g}$. Tout mot de code $\mathbf{c}(x)$ est obtenu par un produit dans $\mathbb{F}_2[x]$ de la forme

$$\mathbf{c}(x) = \mathbf{m}(x) \cdot \mathbf{g}(x)$$

où $\mathbf{m}(x)$ est un polynôme de $\mathbb{F}_2[x]$ de degré $n - 1 - r$. Le code $\mathcal{C}$ ainsi définit est un code linéaire de dimension $k = n - r$. Le polynôme $\mathbf{g}(x)$ est appelé *polynôme générateur* de $\mathcal{C}$ et on note $\mathcal{C} =< \mathbf{g}(x) >$.

Un code $\mathcal{C}$ est *quasi-cyclique* d'indice p s'il admet une matrice $n \times p$ génératrice G circulante par blocs de taille p. On admet que toutes les matrices $G_{i,j}$ sont de taille $p \times p$ et par suite que $n = n_0 p$ et que $k = k_0 p$. Les codes cycliques de longueur n sont donc des codes quasi-cycliques d'indice n dont la matrice génératrice est associée à son polynôme générateur.

Une méthode utile, développée par P. Gaborit dans [Gab05], permet d'obtenir des codes quasi-cycliques d'indice p et de longueur $n = n_0 p$. Elle consiste à considérer un code cyclique $\mathcal{C} =< \mathbf{g}(x) >$ puis à construire le sous-code $S_{n_0}(\mathbf{c})$ engendré par un mot de code $\mathbf{c}(x) \in \mathcal{C}$ et ses $p-1$ décalages à droite modulo $x^n - 1$ de n_0 positions $x^{n_0} \cdot \mathbf{c}(x), x^{2n_0} \cdot \mathbf{c}(x) \ldots, x^{(p-1)n_0} \cdot \mathbf{c}(x)$. Cependant, le code $S_{n_0}(\mathbf{c})$ ainsi défini n'admet pas de matrice génératrice circulante par blocs de taille p ; en fait il faut considérer le code équivalent à $\mathcal{C}$ obtenu par la permutation π qui envoie tout $an_0 + b$ vers $bp + a$ avec $1 \leq a \leq p-1$ et $0 \leq b \leq n_0 - 1$. Ceci signifie qu'à la permutation près, tout mot de code $\mathbf{c}(x)$ d'un code cyclique $\mathcal{C}$ peut être vu comme un vecteur $\mathbf{c} = (\mathbf{c}_0, \ldots, \mathbf{c}_{n_0-1})$ dans lequel chaque $\mathbf{c}_i$ appartient à $\mathbb{F}_2^p \simeq \mathbb{F}_2[x]/(x^p - 1)$ et tel qu'un vecteur $\mathbf{c}' = (\mathbf{c}'_0, \ldots, \mathbf{c}'_{n_0-1})$ où $\mathbf{c}'_i = x \cdot \mathbf{c}_i \mod (x^p - 1)$ est également un mot du code $S_{n_0}(\mathbf{c})$.

5.2.3 Codes BCH

Parmi les codes cycliques les plus utilisés figurent les codes BCH, ainsi nommés d'après R. Bose et D. Ray-Chaudhuri d'une part et A. Hocquenghem d'autre part qui les introduisirent indépendamment et respectivement dans [BRC60] et [Hoc59]. Il existe aujourd'hui de nombreux algorithmes de décodage des codes BCH ; les plus connus sont l'algorithme de Peterson [Pet60, GZ61] et l'algorithme de Berlekamp-Massey [Mas69].

Comme tout code cyclique, les codes BCH peuvent être décrits par un polynôme générateur. Un code BCH sur $\mathbb{F}_q$ de longueur n, premier avec q, et de distance minimale supérieure à $2t+1$ est défini comme suit :

- Déterminer le plus petit m tel que $\mathbb{F}_{q^m}$ possède une racine n^{eme} de l'unité β ;
- Sélectionner un entier positif b ;
- Écrire une liste de $2t$ puissances de β consécutives :

$$\beta^b, \beta^{b+1}, \ldots, \beta^{b+2t-1} \ ;$$

- Déterminer le polynôme minimal dans $\mathbb{F}_q$ de chacune des puissances précédentes de β ; ceux-ci ne sont pas forcement tous distincts, du fait de la conjugaison ;
- Le polynôme générateur $\mathbf{g}(x)$ est le plus petit commun multiple (PPCM) des polynômes minimaux de l'étape précédente.

Le code $\mathcal{C}$ ainsi obtenu est un code cyclique de longueur n et de dimension $n - \deg(\mathbf{g}(x))$ et, comme les polynômes minimaux sont choisis dans $\mathbb{F}_q[x]$, $\mathbf{g}(x)$ a ses coefficients dans $\mathbb{F}_q$ et $\mathcal{C}$ est défini sur $\mathbb{F}_q$.

Si, lors de la procédure précédente b est choisi égal à 1, le code $\mathcal{C}$ obtenu est dit BCH *au sens étroit du terme* (narrow-sense en anglais) ; si $n = q^m - 1$, $\mathcal{C}$ est dit *primitif*.

Remarque 5.2.3 *Obtenir par construction un code BCH primitif de dimension fixée impose une contrainte forte sur* $\deg(\mathbf{g}(x))$*, et le nombre de codes qu'il est ainsi possible d'obtenir est clairement borné par le nombre de polynômes primitifs de degré* m*.*

5.2.4 Codes LDPC

Les codes LDPC (pour *Low Density Parity Check*) sont des codes linéaires définis par des matrices de parité binaires creuses (*i*.e. contenant peu de positions à 1). Tout d'abord décrits par R. Gallager au début des années 60 [Gal63], les codes LDPC sont restés largement ignorés jusqu'à leur redécouverte sous une forme légèrement différente en 1995 par D. McKay et R. Neal [MN95]. Outre leur représentation compacte, ces codes présentent l'avantage de pouvoir être décodés de façon efficace (linéairement en la longueur du code) par décodage itératif (voir [JH04], chapitre 13, par exemple).

5.3 Variante du cryptosystème de McEliece utilisant des sous-codes d'un code BCH

5.3.1 Description

Soit $\mathcal{C}_0$ un code cyclique défini sur $\mathbb{F}_2$ de longueur $n = n_0 p$, où n_0 et p sont deux entiers positifs, et de dimension k. Nous supposons que $\mathcal{C}_0$ admet une matrice génératrice de taille $k' = pk_0 \geq k$; par souci de simplicité on écrit $k = k'$. Soient $\mathbf{c}_1(x), \mathbf{c}_2(x), \ldots, \mathbf{c}_{k_0-1}(x)$, k_0 mots aléatoires de $\mathcal{C}_0$. Considérons le code $\mathcal{C}$ défini par

$$\mathcal{C} = S_{n_0}(\mathbf{c}_1) + S_{n_0}(\mathbf{c}_2) + \cdots + S_{n_0}(\mathbf{c}_{k_0-1}),$$

et que nous considérons être de dimension $k-p = p(k_0-1)$. Nous avons vu à la section précédente qu'à la permutation près, tout vecteur $\mathbf{c}_i(x)$ de $\mathcal{C}$ où $1 \leq i \leq k_0 - 1$ peut être vu comme un vecteur $(\mathbf{c}_{i,0}, \mathbf{c}_{i,12}, \ldots, \mathbf{c}_{i,n_0-1})$ dans lequel chaque $\mathbf{c}_{i,j}$ peut également être vu comme un élément de $\mathbb{F}_2[x]/(x^p - 1)$. Ainsi, $\mathcal{C}$ est un code quasi-cyclique d'indice p dont la matrice génératrice sous forme circulante par blocs de taille p est

$$\mathbf{G}(x) = \begin{pmatrix} \mathbf{c}_{1,1}(x) & \cdots & \mathbf{c}_{1,n_0}(x) \\ \vdots & & \vdots \\ \mathbf{c}_{k_0-1,1}(x) & \cdots & \mathbf{c}_{k_0-1,n_0}(x) \end{pmatrix} \tag{5.1}$$

Dans [Gab05], P. Gaborit propose de générer ainsi un sous-code aléatoire $\mathcal{C}$ de dimension $p(k_0 - 1)$ d'un code BCH primitif $\mathcal{C}_0$ (public) puis de masquer sa structure, tout en préservant

la quasi-cyclicité du code, à l'aide d'une permutation secrète π du groupe symétrique d'ordre n_0. Si $\mathcal{C}$ admet une matrice génératrice $\mathbf{G}(x)$ circulante par blocs de taille p sous la forme (5.1) on notera $\mathcal{C}^\pi$ le code défini par la matrice génératrice

$$\mathbf{G}^\pi(x) = \begin{pmatrix} \mathbf{c}_{1,\pi(1)}(x) & \cdots & \mathbf{c}_{1,\pi(n_0)}(x) \\ \vdots & & \vdots \\ \mathbf{c}_{k_0-1,\pi(1)}(x) & \cdots & \mathbf{c}_{k_0-1,\pi(n_0)}(x) \end{pmatrix}.$$

La variante du cryptosystème de McEliece que propose P. Gaborit consiste à fournir $\mathbf{G}^\pi(x)$ en guise de clef publique ; l'algorithme de décodage utilisé comme clef secrète est quand à lui fourni par la connaissance de $\mathbf{G}(x)$ et de π. Le chiffrement et le déchiffrement sont alors respectivement les opérations d'encodage avec erreur du message et de décodage des chiffrés dans $\mathcal{C}^\pi$.

Le code cyclique $\mathcal{C}_0$ préconisé dans [Gab05] est un code BCH de longueur $2^m - 1$ et de dimension $n - tm$ où $t > 0$. Deux jeux de paramètres sont fournis, correspondant respectivement à des sécurités en 2^{100} et 2^{80} :

- Jeu de paramètres A : $m = 12$, $t = 26$, $n_0 = 45$ et $k_0 = 43$.
- Jeu de paramètres B : $m = 11$, $t = 31$, $n_0 = 23$ et $k_0 = 21$.

On peut remarquer que dans les deux cas, $p > n_0$; cette propriété sera utile lors de la cryptanalyse du schéma.

5.3.2 Cryptanalyse

Nous présentons maintenant une cryptanalyse du schéma proposé dans [Gab05] et présenté à la section précédente. Celle-ci est de type structurelle, c'est-à-dire qu'elle s'attache directement à retrouver la structure du code privé à partir du code public ; il s'agit donc d'une cryptanalyse totale du système. Celle-ci exploite trois observations :

1. Le code $\mathcal{C}_0$ admet une matrice de parité H_0 que l'on peut supposer connue : il existe peu de codes BCH primitifs pour un jeu donné de paramètres (n, m, t) ; par exemple, pour le jeu de paramètres B, ils sont 176. Il est donc possible de tous les essayer.
2. Le code privé $\mathcal{C}$ est un sous-code de $\mathcal{C}_0$ et ainsi, tout mot $\mathbf{c}$ de $\mathcal{C}$ satisfait l'équation de parité
$$H_0 \times \mathbf{c}^T = 0. \tag{5.2}$$
3. La permutation des colonnes de la matrice génératrice sous forme polynomiale $\mathbf{G}(x)$ par π peut être traduite par un produit matriciel par la matrice de permutation $\mathbf{\Pi}$ de taille $n_0 \times n_0$ associée à π. Cette matrice $\mathbf{\Pi}$ peut être également vue comme une matrice de polynômes $\mathbf{\Pi}(x) \in \mathfrak{B}^p_{n_0,n_0}$ dans laquelle une entrée à 0 (resp. 1) correspond au polynôme constant à 0 (resp. 1) ; on a donc :
$$\mathbf{G}^\pi(x) = \mathbf{G}(x) \times \mathbf{\Pi}(x). \tag{5.3}$$

Remarquons que cette dernière équation peut être récrite comme une égalité entre matrices binaires circulantes par blocs de taille $p \times p$:

$$G^\pi = G \times \Pi \tag{5.4}$$

où G^π est la matrice génératrice publique de taille $(k-p) \times n$ et $\Pi = \mathbf{\Pi} \otimes \mathcal{I}_p$. Retrouver Π revient donc en réalité à résoudre un système linéaire à n_0^2 inconnues représentant les entrées de Π^{-1} telles que

$$H_0 \times (G^\pi \times \Pi^{-1})^T = 0, \tag{5.5}$$

c'est-à-dire que toute ligne de la matrice G^π une fois permutée par Π^{-1} doit satisfaire l'équation de parité (5.2). Ce système est linéaire puisque Π^{-1} peut être récrit comme $\mathbf{Pi}^{-1} \otimes \mathcal{I}_p$. En d'autres mots, chaque ligne de G^π fournit $n-k$ équations linéaires binaires vérifiées par Π^{-1}. Ainsi, l'équation (5.5) procure $(k-p)(n-k)$ équations que doivent satisfaire n_0^2 inconnues.

La cryptanalyse du cryptosystème proposé dans [Gab05] requiert donc de résoudre un système surdéterminé constitué de $p^2(k_0-1)(n_0-k_0)$ équations et n_0^2 inconnues puisque, comme nous l'avions déjà remarqué, $p > n_0$. Par exemple, avec le jeu de paramètres B nous obtenons 529 inconnues satisfaisant 316 840 équations et le jeu de paramètres A nous donne 2025 inconnues devant satisfaire 695 604 équations. Beaucoup de ces équations sont évidemment linéairement dépendantes et le succès de cette méthode dépend donc fortement de la taille de l'espace vectoriel des solutions. Une implémentation en Magma [BCP97] a toujours renvoyé, pour les deux jeux de paramètres, un espace vectoriel de dimension 1. Ceci nous révèle donc la permutation secrète.

5.4 Variante utilisant des codes quasi-cycliques LDPC

5.4.1 Description

Posons $n = pn_0$ et $k = p(n_0-1)$ et considérons une matrice de parité H de la forme

$$H = \begin{pmatrix} H_1 & \cdots & H_{n_0} \end{pmatrix} \tag{5.6}$$

dans laquelle chaque H_i est une matrice *creuse* circulante de taille $p \times p$ et chaque colonne de H est de poids fixé d_v. Sans perte de généralité, H_{n_0} est considérée être de rang maximum. On suppose également que l'on a une bonne estimation du nombre t d'erreurs pouvant être décodées par décodage itératif dans le code défini par H.

Dans [BC07], M. Baldi et G. Chiaraluce proposent d'instancier le cryptosystème de McEliece en utilisant en guise de code privé un code $\mathcal{C}'$ défini par une matrice de parité H de la forme (5.6). Pour cela, ils tirent aléatoirement deux matrices S et Q de tailles respectives $k \times k$ et $n \times n$. S (resp. Q) est sélectionnée de façon à ce que chacune de ses lignes et de ses

colonnes soient de poids s (resp. m). Pour obtenir la clef publique, une matrice génératrice sous forme systématique G' du code $\mathcal{C}'$ est tout d'abord calculée. La clef publique est alors la matrice $G = S^{-1} \times G' \times Q^{-1}$. Le chiffrement d'un message $\mathbf{x} \in \mathbb{F}_2^k$ consiste à tirer aléatoirement une erreur $\mathbf{e}$ de longueur n et de poids $t' \leq t/m$ puis à calculer $\mathbf{c} = \mathbf{x}G + \mathbf{e}$. En supposant que la connaissance de H fournisse un t-décodeur itératif Δ dans le code $\mathcal{C}'$, la connaissance de H, S et Q fournit un algorithme de décodage dans le code public $\mathcal{C}$. En effet, pour déchiffrer le message $\mathbf{c}$, il suffit de décoder itérativement $\mathbf{c} \times Q = \mathbf{x} \times S^{-1} \times G' + \mathbf{e} \times Q$ à l'aide de Δ pour obtenir $\mathbf{z} = \mathbf{x} \times S^{-1}$ puis de calculer $\mathbf{x} = \mathbf{z} \times S$. La clef privée correspondante est donc le triplet (H, S, Q). Le point crucial rendant le cryptosystème valide est que $\mathbf{e} \times Q$ est une erreur décodable puisque sont poids est inférieur ou égal à $t'm = t$.

5.4.2 Quelques remarques sur le choix des paramètres

Les auteurs de [BC07] suggèrent de choisir une matrice Q sous forme diagonale. Ils suggèrent aussi les paramètres $p = 4032$, $n_0 = 4$, $d_v = 13$, $m = 7$ et $t = 190$ ($t' = 27$). Finalement, ils proposent de choisir les colonnes et les lignes de chaque bloc de S de poids m, de sorte que $s = m(n_0 - 1)$. Malheureusement ce choix ne permet pas d'obtenir une matrice S inversible. Ceci provient du fait que dans ce cas $x - 1$ divise toujours $\det(\mathbf{S})(x)$ qui ne peut donc pas être premier avec $x^p - 1$ et par suite, $\mathbf{S}(x)$ n'est pas inversible. Il est possible de s'en convaincre en considérant les arguments suivants.

Lemme 5.4.1 *Soit* $\mathbf{S}(x) = (\mathbf{s}_{i,j}(x)) \in \mathfrak{M}_{n_0-1,n_0-1}(R_p)$ *et définissons la matrice binaire* $\tilde{S} = (\tilde{s}_{i,j})$ *par* $\tilde{s}_{i,j} = wt(\mathbf{s}_{i,j}) \mod 2$. *On a alors*

$$\det(\tilde{S}) = wt(\det(\mathbf{S})(x)) \mod 2$$

Preuve – Ce lemme provient du fait que $wt(\mathbf{u} + \mathbf{v}) = wt(\mathbf{u}) + wt(\mathbf{v}) - 2wt(\mathbf{u} \star \mathbf{v})$ pour tout $\mathbf{u}(x)$ et tout $\mathbf{v}(x)$ dans $\mathbb{F}_2[x]$, ce qui implique que

$$\begin{cases} wt(\mathbf{u} + \mathbf{v}) &= wt(\mathbf{u}) + wt(\mathbf{v}) & \mod 2 \\ wt(\mathbf{u} \cdot \mathbf{v}) &= wt(\mathbf{u}) \cdot wt(\mathbf{v}) & \mod 2 \end{cases}$$

♢

Proposition 5.4.1 *Pour tout* $\mathbf{S}(x) \in \mathfrak{M}_{,3,3}(R_p)$ *tel que chaque* $\mathbf{s}_{i,j}$ *est de poids* m, $x - 1$ *divise* $\det(\mathbf{S})(x)$.

Preuve – En utilisant les même notations que dans le lemme précédent, on sait que $\det(\tilde{S}) = 0$ puisque $\tilde{S}$ est la matrice toute à 1. D'après le lemme précédent, il s'ensuit que $\det(\mathbf{S})(x)$ a un support de poids pair. Ceci implique que $x - 1$ divise $\det(\mathbf{S})(x)$. ♢

Afin d'éviter cette situation, nous introduisons dans $\mathbf{S}$ quelques polynômes de poids différents de m de façon à ce que $\det \tilde{S} = 1$. Un choix possible est le suivant : tout d'abord nous choisissons la matrice non singulière

$$\tilde{S} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

puis lorsque $\tilde{s}_{i,j} = 1$, nous choisissons l'entrée correspondante $\mathbf{s}_{i,j}$ de poids m et lorsque $\tilde{s}_{i,j} = 1$, nous choisissons l'entrée correspondante $\mathbf{s}_{i,j}$ de poids $m - 1$.

De plus, une attaque par décodage sur le schéma proposé par M. Baldi et G. Chiaraluce consiste à rechercher des mots de poids inférieur à $t = 190$ dans un code de longueur $n = 16128$ et de dimension $k = 12096$. Or, celle-ci a un coût d'environ $2^{78,5}$ à l'aide de l'algorithme de Canteaut-Chabaud [CC98]. Elle a donc un coût inférieur à la sécurité en 2^{80} recherchée.

5.4.3 Cryptanalyse

Le but de notre attaque est de retrouver le code privé $\mathcal{C}'$, défini à l'aide d'une matrice de parité H de la forme (5.6). On sait que les matrices S et Q sont définies de façon équivalente par les polynômes $\mathbf{s}_{i,j}(x)$ et $\mathbf{q}_{i,j}(x)$ respectivement, et que la matrice Q est choisie pour être diagonale, c'est-à-dire que $\mathbf{q}_{i,j}(x) = 0$ si $i \neq j$. Par souci de simplicité, nous notons $\mathbf{q}_i(x) = \mathbf{q}_{i,i}(x)$. De plus, chaque polynôme $\mathbf{q}_i(x)$ est inversible modulo $x^p - 1$ puisque Q est inversible.

Il est également immédiat de remarquer que la matrice génératrice privée G' est

$$G' = \left(\begin{array}{c|c} \mathcal{I}_k & \begin{matrix} \left(H_{n_0}^{-1} H_1\right)^T \\ \vdots \\ \left(H_{n_0}^{-1} H_{n_0-1}\right)^T \end{matrix} \end{array}\right)$$

En d'autres mots, la matrice génératrice G du code public est

$$\begin{aligned} G &= S^{-1} \times \left(\begin{array}{c|c} \mathcal{I}_k & \begin{matrix} \left(H_{n_0}^{-1} H_1\right)^T \\ \vdots \\ \left(H_{n_0}^{-1} H_{n_0-1}\right)^T \end{matrix} \end{array}\right) \times \begin{pmatrix} Q_1^{-1} & & 0 \\ & \ddots & \\ 0 & & Q_{n_0}^{-1} \end{pmatrix} \\ &= S^{-1} \times \left(\begin{array}{ccc|c} Q_1^{-1} & & 0 & \left(H_{n_0}^{-1} H_1\right)^T Q_{n_0}^{-1} \\ & \ddots & & \vdots \\ 0 & & Q_{n_0-1}^{-1} & \left(H_{n_0}^{-1} H_{n_0-1}\right)^T Q_{n_0}^{-1} \end{array}\right). \end{aligned}$$

Si on note $G_{\leq k}$ la matrice constituée des k premières colonnes de G, on a donc

$$G_{\leq k} = S^{-1} \times \begin{pmatrix} Q_1^{-1} & & 0 \\ & \ddots & \\ 0 & & Q_{n_0-1}^{-1} \end{pmatrix},$$

ce qui implique que $G_{\leq k}^{-1}$ est une matrice circulante par blocs de taille p définis par des polynômes $\mathbf{g}_{i,j}(x)$ vérifiant les équations

$$\mathbf{g}_{i,j}(x) = \mathbf{q}_i(x) \cdot \mathbf{s}_{i,j}(x) \mod (x^p - 1). \tag{5.7}$$

Il est immédiat de constater que le poids de l'un de ces polynômes $\mathbf{g}_{i,j}(x)$ est au plus m^2 ; en réalité on peut observer qu'avec une bonne probabilité son poids est exactement m^2, du fait du poids très faible des polynômes. Posons $\mathbf{q}_i(x) = x^{e_1} + \cdots + x^{e_m}$ et $\mathbf{s}_{i,j}(x) = x^{\ell_1} + \cdots + x^{\ell_m}$ avec $0 \leq e_a, \ell_a \leq p-1$ pour tout $0 \leq a \leq m$. Fixons $\mathbf{q}_i(x)$ et supposons que les monômes x^{ℓ_a} de $\mathbf{s}_{i,j}(x)$ sont choisis indépendamment et uniformément. Nous souhaitons estimer la probabilité que le support de $\mathbf{g}_{i,j}(x)$ contienne le support d'au moins un décalage $x^{\ell_a} \cdot \mathbf{q}_i(x)$, ainsi que la probabilité que le poids de $\mathbf{g}_{i,j}(x)$ soit exactement m^2.

Lemme 5.4.2 *Soient w entier $\ell_1, \ldots, \ell_w$ deux à deux distincts tels que $0 \leq \ell_a \leq p-1$ pour $1 \leq a \leq w$. Pour tout entier aléatoire ℓ soit différent de $\ell_1, \ldots, \ell_w$ tel que $0 \leq \ell \leq p-1$, on a*

$$\Pr\left[(x^{\ell_1} + \cdots + x^{\ell_w}) \cdot \mathbf{q}_i(x) \star x^{\ell} \cdot \mathbf{q}_i(x) \neq 0\right] \leq w\frac{m(m-1)}{p-w}$$

PREUVE – Posons $\mathbf{r}(x) = (x^{\ell_1} + \cdots + x^{\ell_w}) \cdot \mathbf{q}_i(x)$. Par la borne de l'union, on a

$$\Pr\left[\mathbf{r}(x) \star x^{\ell} \cdot \mathbf{q}_i(x) \neq 0\right] \leq \sum_{a=1}^{w} \Pr\left[x^{\ell_a} \cdot \mathbf{q}_i(x) \star x^{\ell} \cdot \mathbf{q}_i(x)\right].$$

La probabilité $\Pr\left[x^{\ell_a} \cdot \mathbf{q}_i(x) \star x^{\ell} \cdot \mathbf{q}_i(x)\right]$ est au plus la fraction des entiers ℓ différents de $\ell_1, \ldots, \ell_w$ tels qu'il existe $0 \leq b \leq m$ et $0 \leq c \leq m$ tels que

$$\ell_a + e_b = \ell + e_c \mod p.$$

Ainsi, cette fraction est donnée par le ratio du nombre de paires (e_b, e_c) avec $b \neq c$ sur le nombre des valeurs possibles de ℓ, c'est-à-dire $m(m-1)/(p-w)$. ♢

Proposition 5.4.2 *La probabilité $\Pr\left[x^{\ell} \cdot \mathbf{q}_i(x) \subset \mathbf{g}_{i,j}(x)\right]$ pour $\ell \in \{\ell_1, \ell_m\}$ que le support de $\mathbf{g}_{i,j}(x)$ contienne le support de $x^{\ell} \cdot \mathbf{q}_i(x)$ est bornée par*

$$\Pr\left[x^{\ell} \cdot \mathbf{q}_i(x) \subset \mathbf{g}_{i,j}(x)\right] \geq \left(1 - \frac{m(m-1)}{p-1}\right)^{m-1}.$$

PREUVE – Cette inégalité est obtenue en prenant $w = 1$ dans le lemme 5.4.2 et par l'indépendance du choix des $m-1$ autres monômes de $\mathbf{s}_{i,j}$. ♢

Proposition 5.4.3 *La probabilité q que $\mathbf{g}_{i,j}(x)$ soit exactement de poids m^2 est bornée par*

$$q \geq \prod_{w=1}^{m-1} \left(1 - w\frac{m(m-1)}{p-w}\right).$$

PREUVE – Pour $2 \leq w \leq m$, soit E_w l'événement

$$E_w \ : \ (x^{\ell_1} + \cdots + x^{\ell_{w-1}}) \cdot \mathbf{q}_i(x) \star x^{\ell_w} \cdots \mathbf{q}_i(x) = 0$$

lorsque chaque monôme x^{ℓ_a} est choisi indépendamment et uniformément. On définit E_1 comme étant l'univers entier. On a alors

$$q \geq \Pr\left[E_2 \cup \cdots \cup E_m\right].$$

A l'aide du théorème de Bayes, on a également

$$\Pr\left[E_2 \cup \cdots \cup E_m\right] = \prod_{w=1}^{m} \Pr\left[E_w \mid E_{w-1} \cup \cdots \cup E_1\right].$$

Or, on sait d'après le lemme 5.4.2 que $\Pr\left[E_w \mid E_{w-1} \cup \cdots \cup E_1\right] \geq \left(1 - w\frac{m(m-1)}{p-w}\right)$. $\diamondsuit$

Différentes stratégies

Nous avons donc établi que la clef publique nous permet d'obtenir les $k_0(n_0 - 1)$ équations données par la formule (5.7). Il nous faut alors en extraire les polynômes $\mathbf{q}_i(x)$ ou, de façon équivalente, les polynômes $\mathbf{s}_{i,j}(x)$. Pour cela, nous proposons deux stratégies différentes.

Première stratégie On a vu que, d'après la proposition 5.4.2, le support de $\mathbf{g}_{i,j}(x)$ contient avec très grande probabilité le support d'au moins une version décalée de $\mathbf{q}_i(x)$. En effet, pour les paramètres proposés dans [BC07], nous obtenons une probabilité $\Pr\left[x^\ell \cdot \mathbf{q}_i(x) \subset \mathbf{g}_{i,j}(x)\right] \geq 0,94$.

Une stratégie possible permettant de retrouver le polynôme $\mathbf{q}_i(x)$ pour $1 \leq i \leq n_0 - 1$ (et donc les polynômes $\mathbf{s}_{i,j}(x)$) consiste à énumérer tous les m-tuples $(u_1, \ldots, u_m)$ appartenant au support de $\mathbf{g}_{i,j}(x)$ pour former un polynôme $\mathbf{u}(x) = \sum_a x^{u_a}$ tel que $\mathbf{u}^{-1}(x) \cdot \mathbf{g}_{i,j'}(x)$ soit de poids m pour $1 \leq j' \leq n_0 - 1$. Le coût de cette attaque est $O\left(\binom{m^2}{m} p^2\right)$, ce qui correspond à $2^{50.3}$ opérations binaires pour les paramètres considérés.

Seconde stratégie Une seconde stratégie permet de retrouver la matrice privée S et ainsi les matrices $Q_1, \ldots, Q_{n_0-1}$. Celle-ci requiert de rechercher des mots de très petit poids dans un code linéaire. Comme nous l'avons vu section 3.2.3, le meilleur algorithme permettant d'accomplir cette tâche a été proposé par D. BERNSTEIN, T. LANGE et C. PETERS [BLP08], lequel affine l'approche proposée par F. CHABAUD et A.CANTEAUT dans [CC94], qui elle-même améliore l'algorithme de J. STERN [Ste89]. Cependant, afin d'obtenir une borne simple de la complexité de notre attaque, nous considérons l'algorithme de Stern, comme cela était fait dans [BC07].

L'effort $\Omega_{n,k,w}$ nécessaire à l'algorithme de Stern pour retrouver A_w mots de code de poids w dans un code de longueur n et de dimension k satisfait $\Omega_{n,k,w} \leq \frac{N}{A_w P_w}$ où N est le nombre

d'opérations binaires nécessaires à une itération et P_w représente la probabilité de trouver un mot de poids w au cours d'une itération. Pour deux paramètres g et ℓ à optimiser, on a

$$N = \frac{(n-k)^3}{2} + k(n-k)^2 + 2g\ell\binom{k/2}{g} + 2g(n-k)\frac{\binom{k/2}{g}^2}{2^\ell} \tag{5.8}$$

et

$$P_w = \frac{\binom{w}{g}\binom{n-w}{k/2-g}}{\binom{n}{k/2}} \frac{\binom{w-g}{g}\binom{n-k/2-w+g}{k/2-g}}{\binom{n-k/2}{k/2}} \frac{\binom{n-k-w+2g}{\ell}}{\binom{n-k}{\ell}}. \tag{5.9}$$

Rappelons que $G_{\leq k}^{-1}$ est définie par les polynômes $\mathbf{g}_{i,j}(x)$ et posons $\mathbf{d}_{i,j}(x)$ le polynôme $\mathbf{g}_{i,j}(x)\mathbf{g}_{i,1}^{-1}(x) \mod (x^p - 1)$. Considérons le code $\mathcal{C}_i$ défini par la matrice génératrice

$$E_i = \begin{pmatrix} \mathcal{I}_p & D_{i,2} & \cdots & D_{i,n_0-1} \end{pmatrix}$$

où $D_{i,j}$ est la matrice circulante définie par $\mathbf{d}_{i,j}(x)$. Le code $\mathcal{C}_i$ contient alors au moins p mots de code de petit poids $(n_0-1)m$ (c'est-à-dire de poids 21 pour les paramètres qui nous intéressent) puisque

$$S_{i,1} \times E_i = \begin{pmatrix} S_{i,1} & S_{i,2} & \cdots & S_{i,n_0-1} \end{pmatrix}.$$

Il est donc ainsi possible de retrouver les matrices $S_{i,1}, \ldots, S_{i,n_0-1}$ avec une complexité de 2^{32} opérations binaires en cherchant un mot de poids 21 dans un code de dimension p et de longueur $n = (n_0-1)p = 12096$ à l'aide de l'algorithme de Stern avec les paramètres $(g, \ell) = (3, 43)$.

Extraire le code privé

Chacune des deux stratégies ci-dessus permet de découvrir S, Q_1, …, Q_{n_0-1}. Cependant, sans la connaissance de Q_{n_0} il est impossible de retrouver le code privé $\mathcal{C}'$.

Construisons maintenant la matrice

$$\tilde{G} = S \times G \times \begin{pmatrix} Q_1 & & & 0 \\ & \ddots & & \\ & & Q_{n_0-1} & \\ & & & \mathcal{I}_p \end{pmatrix} = \left(\begin{array}{ccc|c} & & & A_1 \\ & \mathcal{I}_k & & \vdots \\ & & & A_{n_0-1} \end{array}\right)$$

où $A_i = (H_{n_0} H_i)^T \times Q_{n_0}^{-1}$. On peut facilement vérifier que pour $i \neq j$, si H_j est inversible on a $(A_i \times A_j^{-1})^T = H_i \times H_j^{-1}$. On définit donc $B_{i,j} = (A_i \times A_j^{-1})^T$. Alors pour $1 \leq i \leq n_0 - 1$ fixé et deux entiers distincts j et j', on a $H_j \times B_{i,j} = H'_j \times B_{i,j'} = H_i$.

Considérons maintenant le code défini par la matrice génératrice G_1 suivante :

$$G_1 = \begin{pmatrix} \mathcal{I}_p & B_{2,1} & \cdots & B_{n_0-1,1} \end{pmatrix}.$$

On voit clairement que $H_1 \times G_1 = \begin{pmatrix} H_1 & H_2 & \cdots & H_{n_0-1} \end{pmatrix}$, ce qui signifie que G_1 engendre un code dont la distance minimale est inférieure à $(n_0-1)d_v$. Ainsi, en appliquant à nouveau

un algorithme de recherche de mots de petits poids, il est possible de découvrir les matrices $H_1, H_2, \ldots, H_{n_0-1}$. Par exemple, l'algorithme de Stern [Ste89] nécessite environ 2^{37} opérations binaires avec les paramètres $(g, \ell) = (3, 43)$ pour trouver des mots de poids $(n_0-1)d_v = 3\times 13 = 39$ dans un code de longueur $p(n_0 - 1) = 12096$ et de dimension $p = 4032$.

Finalement, il est possible de calculer $(H_i^T)^{-1} \times A_i = (H_{n_0}^{-1})^T \times Q_{n_0}^{-1}$ pour $1 \leq i \leq n_0 - 1$. En inversant cette matrice puis en appliquant à nouveau la seconde stratégie proposée, il est possible de découvrir les matrices H_{n_0} et Q_{n_0}.

5.5 Bilan

L'idée d'introduire les codes quasi-cycliques et quasi-cycliques LDPC est motivée par le besoin pratique de réduire la taille des clefs dans le cryptosystème de McEliece et ses variantes. Cependant, nous avons montré que ces tentatives se font au dépend de la sécurité du schéma.

Nous avons en effet présenté différentes cryptanalyses structurelles de deux variantes du cryptosystème de McEliece utilisant ces idées. La première attaque s'applique à la variante présentée dans [Gab05] et recouvre la permutation secrète supposée masquer la structure secrète du code. Nous avons montré qu'extraire la clef secrète de la clef publique revient à résoudre un système d'équations linéaires sur-contraint. La seconde atttaque réalise un bris total du schéma proposé dans [BC07]. Tout d'abord, nous recherchons des diviseurs de petit poids d'un polynôme public. La seconde étape retrouve la matrice de parité secrète d'un code quasi-cyclique LDPC, également secret. Pour cela, nous recherchons des mots de petit poids dans une version tronquée du code privé, obtenu à partir de la matrice publique. Nos implémentations ont montré que la première étape peut être réalisée en environ 140 secondes et que la seconde peut être obtenue en environ 15 minutes. Ces cryptanalyses ont été considérées lors du design de nouvelles variantes du cryptosystème de McEliece visant à réduire la taille des clefs comme [BCGO09, MB09]. Malheureusement, celles-ci ont par la suite été également cryptanalysées avec succès [FOPT10a]. Les codes de Goppa forment donc aujourd'hui la seule famille de codes permettant d'instancier le schéma de McEliece de façon sûre.

L'existence de ces cryptanalyses montre bien le besoin de présenter une preuve de sécurité réductionniste lors de la proposition d'un nouveau schéma. En effet celle-ci permet, sinon de se prémunir contre toute attaque, tout au moins d'identifier d'éventuelles faiblesses du schéma : les faiblesses de constructions sont alors quantifiées, et les problèmes supposés difficiles clairement exposées.

Troisième partie

Signature fondée sur les codes correcteurs d'erreurs

Chapitre 6

Sécurité réductionniste du schéma de signature CFS

Ce chapitre a fait l'objet de la publication [Dal08].

Sommaire

Dès la réalisation en 1978 du premier schéma de chiffrement à clef publique par R. Rivest, A. Shamir et L. Adelman [RSA78], la signature électronique s'est trouvée au cœur des enjeux de la cryptographie asymétrique. Si RSA permet d'en dériver simplement un schéma de signature, nous verrons que ce n'est pas le cas du schéma de McEliece et de ses variantes.

Cependant, l'article [FS87] de A. Fiat et A. Shamir propose plusieurs solutions aux problématiques liées à la preuve d'identité. Ils y montrent notamment qu'un protocole d'authentification à divulgation nulle de connaissance peut être utilisé (dans le modèle de l'oracle aléatoire) pour construire un schéma de signature électronique, à l'aide d'une heuristique aujourd'hui connue sous le nom d'*heuristique de Fiat-Shamir*. Dans celle-ci, le signataire prend le rôle du prouveur et simule le vérificateur en remplaçant la génération de ses aléas par une fonction de hachage publique. Il communique ensuite ses éléments d'authentification en guise de signature. Le destinataire du message peut alors vérifier la signature en recalculant les « aléas » du vérificateur à l'aide de la fonction de hachage puis en effectuant la vérification finale du protocole d'authentification.

Lors d'Eurocrypt'89, J. Stern a proposé le premier schéma d'identification à divulgation nulle de connaissance fondé sur la théorie des codes correcteurs d'erreurs [Ste90], permettant alors d'en dériver le premier protocole de signature utilisant les codes. Un protocole présentant de meilleures performances — et rendant ainsi possible une utilisation pratique du protocole — a été proposé par le même auteur à CRYPTO'93 [Ste94, Ste96]. Sa sécurité repose essentiellement sur la difficulté du problème CSD présenté section 3.2. Cependant, les signatures obtenues en appliquant l'heuristique de Fiat-Shamir sont de grande taille, bien qu'elles utilisent des clefs de petite taille.

Ce n'est qu'en 1997 que G. Kabatianskii, E. Krouk et B. Smeets proposèrent le premier schéma de signature fondé sur le théorie des codes et ne s'appuyant pas sur un protocole d'identification [KKS97]. Celui-ci fut cryptanalysé en 2006 par P. L. Cayrel, A. Otmani et D. Vergnaud [COV07]. En 2001, N. Courtois, M. Finiasz et N. Sendrier proposèrent un nouveau schéma de signature fondé sur le théorie des codes et permettant d'obtenir des signatures courtes. S'ils donnent un certain nombre d'arguments de sécurité en faveur de leur schéma, ils ne fournissent pas de preuve de sécurité réductionniste telle que nous l'avons décrit dans le chapitre 2. Nous nous proposons d'établir une preuve réductionniste de sa sécurité. Pour cela, il nous faut au préalable introduire une légère modification du schéma.

Ces résultats ont fait l'objet de la publication [Dal08] dans les actes de WEWoRC 2007 (Western European Workshop on Research in Cryptology).

6.1 Schéma CFS

La construction du schéma de signature de N. Courtois, M. Finiasz et N. Sendrier repose sur l'adaptation de l'approche FDH introduite par M. Bellare et P. Rogaway [BR93] permettant de construire un schéma de signature (prouvé sûr dans le modèle de l'oracle aléatoire) à partir d'un schéma de chiffrement à clef publique.

Dans cette approche, le message à signer m est haché pour obtenir un condensé $\mathcal{H}(m)$. Celui-ci est donné en entrée à une fonction de déchiffrement qui interprète le condensé comme le résultat d'un chiffrement et utilise donc la clef secrète du signataire pour l'inverser. Le résultat σ du déchiffrement est donné comme signature du message par le propriétaire de la clef privée utilisée. Un vérificateur applique simplement la fonction de chiffrement à σ avec la clef publique du signataire et vérifie que le résultat est bien le condensé $\mathcal{H}(m)$ du message.

Cependant, cette méthode est applicable uniquement lorsque tous les éléments de l'ensemble d'arrivée de la fonction de hachage (une suite de bits d'une longueur fixée) sont inversibles par la fonction de déchiffrement, ce qui n'est pas le cas dans les chiffrements de McEliece et de Niederreiter. En effet, il n'est possible de décoder un élément aléatoire dans l'espace entier que si cet élément est à une distance suffisamment petite du code. En général, la proportion de tels éléments est très faible. Courtois, Finiasz et Sendrier proposent d'utiliser le chiffrement de Niederreiter avec des codes de Goppa possédant une faible capacité de correction ; dans ce cas, la proportion de syndromes décodables est $\frac{1}{t!}$.

Il est alors possible d'associer à un même message plusieurs condensés différents, par exemple en initialisant un compteur i puis en hachant le message m concaténé à ce compteur, jusqu'à trouver un indice i_0 pour lequel le résultat de $\mathcal{H}(m\|i_0)$ est décodable. La signature est donc la donnée de la valeur du compteur i_0 et du résultat x du décodage de $\mathcal{H}(m\|i_0)$.

Définition 6.1.1 (Schéma CFS) *Le schéma de signature* CFS *est défini par les quatre algorithmes suivants :*

- *CFS.Initialiser*(1^κ)

  1. *Déterminer les paramètres m et t nécessaires pour assurer une sécurité adéquate.*
  2. *Choisir une fonction de hachage $\mathcal{H} : \{0,1\}^* \to \mathbb{F}_2^{mt}$.*
  3. *Renvoyer $\mathcal{P} = (m, t, \mathcal{H})$.*

  Pour faciliter la lecture de la suite, on note $n = 2^m$ et $k = n - mt$.
- *CFS.GenererClefs*$(\mathcal{P})$ = *Niederreiter.GenererClefs*$(\mathcal{P})$

  1. *Tirer aléatoirement un code $\mathcal{C}_0 \xleftarrow{R} \mathcal{G}_{m,t}$.*
  2. *Calculer une matrice de parité H_0 du code $\mathcal{C}_0$.*

3. *Construire un algorithme* Δ_0 *de décodage par syndrome à distance* t *pour le code* $\mathcal{C}_0$ *:*
$\begin{cases} \forall e \in \mathbb{F}_2^n \text{ tel que } wt(e) \leq t,\ \Delta_0(H_0 e^T) = e^T. \\ \forall e \in \mathbb{F}_2^n \text{ tel que } wt(e) > t,\ \Delta_0(H_0 e^T) = \emptyset. \end{cases}$
4. *Tirer aléatoirement deux matrices* U *et* P *telles que*
   (a) U *soit une matrice* inversible *de taille* $(n-k) \times (n-k)k$,
   (b) P *soit une matrice de permutation de taille* $n \times n$.
5. *Calculer la matrice* $H = UH_0P$.
6. *Renvoyer le couple de clef publique/clef privée* (PK, SK) *où* $PK = H$ *et* $SK = (\Delta_0, U, P)$

– *CFS.Signer*$(\mathcal{P}, SK = (\Delta_0, U, P), m \in \{0,1\}^*)$
1. *Initialiser le compteur* $i \leftarrow 0$ *et le vecteur* $\tilde{x}^T \leftarrow \emptyset$.
2. *Tant que* $\tilde{x}^T = \emptyset$ *faire*
   (a) $i \leftarrow i + 1$;
   (b) *Calculer* $\tilde{x} \leftarrow \Delta_0(U^{-1}\mathcal{H}(m, i))$.
3. *Calculer* $x \leftarrow \tilde{x}P$.
4. *Renvoyer* $\sigma = (x, i)$

– *CFS.Vérifier*$(\mathcal{P}, PK, m', \sigma')$

Renvoyer valide *si* $Hx^T = \mathcal{H}(m' \| i')$, invalide *sinon.*

La consistance du schéma est donnée par l'unicité du décodage par syndrome d'une erreur de poids t.

6.2 Jeu de sécurité pour la signature

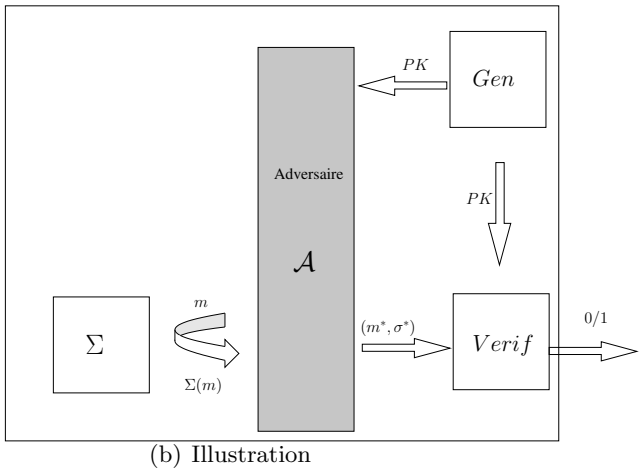
Comme nous l'avons vu section 1.2.2, le modèle de sécurité standard pour la signature est le modèle EF-CMA. Dans ce modèle, l'adversaire considéré reçoit en entrée une clef publique PK, générée par le challenger en utilisant l'algorithme de génération des clefs. L'adversaire a la possibilité d'obtenir la signature de messages de son choix pour la clef privée correspondante. Cette possibilité est modélisée par l'accès à un oracle de signature Σ auquel il peut faire q_Σ requêtes pour obtenir la signature de messages de son choix. À la fin de l'attaque, l'adversaire renvoie une *contrefaçon existentielle*, c'est-à-dire un couple message/signature (m^*, σ^*). Si σ^* est une signature valide pour le message m^* et la clef publique, alors on dit que l'adversaire remporte le jeu. Bien entendu, on exige également que σ' n'ait jamais été fournie comme signature par l'oracle Σ. La description du jeu EF-CMA pour le paramètre de sécurité κ est donnée, d'un point de vue algorithmique, figure 6.1(a) ; elle se place du point de vue d'un challenger auquel l'adversaire est fourni.

On définit le *succès d'un* (τ, q_Σ)*-adversaire* $\mathcal{A}^\Sigma$ (un adversaire s'exécutant en temps τ et effectuant q_Σ requêtes de signature à un oracle Σ) lors d'une attaque EF-CMA de paramètre de sécurité κ contre le schéma de signature S comme la probabilité qu'il remporte le jeu EF-CMA(κ) :

$$\mathsf{Succ}^S_{EF-CMA(\kappa)}(\mathcal{A}) = \Pr[\mathcal{A} \text{ remporte le jeu EF-CMA}(\kappa)].$$

Entrées : Un paramètre de sécurité κ
Données : Un adversaire $\mathcal{A}$
$\mathcal{P} \leftarrow \mathsf{S.Initialiser}(1^\kappa)$;
$(SK, PK) \leftarrow \mathsf{S.GenererClefs}(\mathcal{P})$;
Initialiser l'oracle Σ;
$(m^*, \sigma^*) \leftarrow \mathcal{A}^\Sigma(PK)$;
si *S.Vérifier*$(\mathcal{P}, PK, m^*, \sigma^*)$ = *valide* ***et*** Σ *n'a pas fourni* σ^* **alors**
  $\mathcal{A}$ remporte le jeu
sinon
  $\mathcal{A}$ perd le jeu
fin

(a) Présentation algorithmique, point de vue du challenger



(b) Illustration

FIGURE 6.1 – Jeu de sécurité EF-CMA(κ)

Si la preuve de sécurité se place dans le modèle de l'oracle aléatoire, l'adversaire dispose en plus d'un accès à un oracle de hachage $\mathcal{H}$ auquel il peut faire $q_\mathcal{H}$ requêtes pour obtenir le haché d'un message m de son choix. Dans ce cas, on parlera du jeu ROM-EF-CMA(κ) décrit sur la figure 6.2.

On établit une définition similaire du succès d'un $(\tau, q_\Sigma, q_\mathcal{H})$*-adversaire* $\mathcal{A}^{\Sigma,\mathcal{H}}$ (un adversaire s'exécutant en temps τ, effectuant q_Σ requêtes de signature à un oracle Σ et $q_\mathcal{H}$ requêtes à un oracle de hachage $\mathcal{H}$) lors d'une attaque EF-CMA de paramètre de sécurité κ contre le schéma de signature S dans le modèle de l'oracle aléatoire :

$$\mathsf{Succ}^S_{ROM-EF-CMA(\kappa)}(\mathcal{A}) = \Pr[\mathcal{A} \text{ remporte le jeu ROM-EF-CMA}(\kappa)].$$

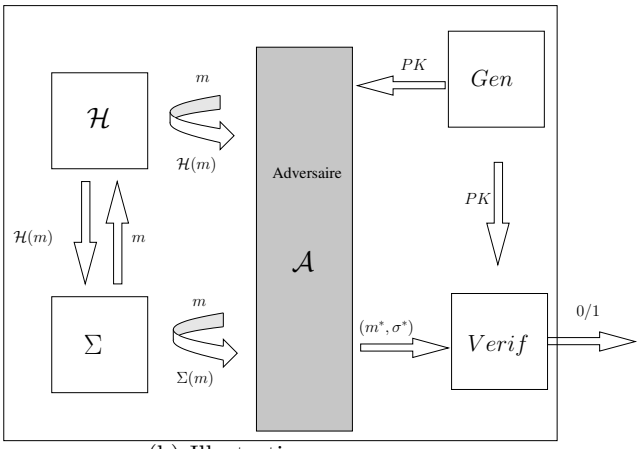
Définition 6.2.1 (schéma de signature (ϵ, τ)-EF-CMA sûr) *Dans le modèle standard (resp. dans le modèle de l'oracle aléatoire), un schéma de signature S est dit $(\epsilon, \tau, q_\Sigma)$-EF-CMA*

Entrées : Un paramètre de sécurité κ

Données : Un adversaire $\mathcal{A}$

1 $\mathcal{P} \leftarrow \mathsf{S.Initialiser}(1^\kappa)$;
2 $(SK, PK) \leftarrow \mathsf{S.GenererClefs}(\mathcal{P})$;
3 Initialiser les oracles Σ et $\mathcal{H}$;
4 $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\Sigma,\mathcal{H}}(PK)$;
5 **si** *S.Vérifier*$(\mathcal{P}, PK, m^*, \sigma^*)$ = *valide et* Σ *n'a pas fourni* σ^* **alors**
6 $\mathcal{A}$ remporte le jeu
7 **sinon**
8 $\mathcal{A}$ perd le jeu
9 **fin**

(a) Présentation algorithmique, point de vue du challenger



(b) Illustration

FIGURE 6.2 – Jeu de sécurité ROM-EF-CMA(κ)

sûr(resp. $(\epsilon, \tau, q_\Sigma, q_\mathcal{H})$*-EF-CMA sûr) si pour tout* (τ, q_Σ)*-adversaire (resp.* $(\tau, q_\Sigma, q_\mathcal{H})$*-adversaire)* $\mathcal{A}$*,*

$$\mathbf{Succ}^{S}_{(resp.\ ROM-)EF-CMA(\kappa)}(\mathcal{A}) \leq \epsilon(\tau).$$

6.3 Modification du schéma

Le schéma de COURTOIS, FINIASZ et SENDRIER telle qu'il est présenté section 6.1 ne permet pas d'être prouvé. En effet, le schéma original itère le hachage du message en incrémentant un compteur jusqu'à ce que le résultat soit décodable à distance t. Dès lors si la signature d'un message m est $\sigma = (x, i)$, on sait que pour tout indice $j \leq i$ le haché $\mathcal{H}(m\|j)$ n'est pas décodable. Pour prouver que la sécurité du schéma est reliée au problème CGPSD, il faudrait donc être en mesure de simuler le comportement de l'oracle de hachage une fois la clef publique remplacée par un code aléatoire. En particulier, il serait nécessaire de fournir pour ces indices j un syndrome *non décodable*. Or, comme nous l'avons vu sections 3.2 et 4.1.2, savoir si un syndrome aléatoire est décodable ou non dans un code aléatoire est un problème difficile (et même NP-complet) ; il est donc impossible d'être certain de fournir un syndrome non décodable simplement en le tirant au hasard. Pour pouvoir exhiber une preuve de sécurité du schéma il est donc nécessaire soit de casser cette régularité, soit d'être en mesure de calculer des syndromes non-décodables.

Une solution pour casser la régularité des réponses de l'oracle aléatoire consiste à ne plus effectuer le hachage du message avec des valeurs de compteur consécutives, mais de plutôt choisir des indices aléatoires. Ainsi, si la signature d'un message m est $\sigma = (x, i)$, on sait bien évidemment que $\mathcal{H}(m\|i)$ est décodable, mais ceci ne nous apprend rien sur les autres hachés $\mathcal{H}(m\|j)$ possibles. La simulation de l'oracle de hachage est alors possible puisqu'il pourra se contenter de renvoyer un syndrome aléatoire. Nous appelons cette variante mCFS et nous définissons l'algorithme de signature, les autres algorithmes restant inchangés :

mCFS.Signer $(\mathcal{P}, SK = (\Delta_0, U, P), m \in \{0,1\}^*)$

1. Initialiser le compteur $i \leftarrow 0$ et le vecteur $\tilde{x}^T \leftarrow \emptyset$.
2. Tant que $\tilde{x}^T = \emptyset$ faire
   (a) $i \xleftarrow{R} \{1, \ldots, 2^{mt}\}$;
   (b) Calculer $\tilde{x} \leftarrow \Delta_0(U^{-1}\mathcal{H}(m, i))$.
3. Calculer $x \leftarrow \tilde{x}P$.
4. Renvoyer $\sigma = (x, i)$

Une méthode permettant de construire un syndrome non-décodable est de limiter le poids des signatures à $t' = t - \epsilon$. Ainsi, le syndrome s d'un mot $e \in \mathbb{F}_2^n$ de poids compris entre t et $t + \epsilon$ n'est pas décodable à distance t'. En effet, dans le cas contraire il existerait par définition un mot $x \in \mathbb{F}_2^n$ de poids inférieur à t' de syndrome $s = Hx^T$. Le mot $x + e$ serait alors de syndrome nul, et appartiendrait donc au code. Or, ce mot est de poids $t \leq wt(x + e) \leq t' + t + \epsilon = 2t$, c'est-à-dire inférieur à la distance minimale du code. Ce mot ne peut donc pas exister et s n'est pas décodable à distance t'.

Cependant, cette méthode pose plusieurs problèmes. Tout d'abord, en procédant ainsi la probabilité qu'un syndrome aléatoire soit décodable diminue de $\frac{1}{t!}$ à $\frac{1}{(t-\epsilon)!}$, augmentant ainsi le coût de génération d'une signature. Par exemple, avec $t = 9$ et $\epsilon = 2$, il faudra réaliser 81 fois plus de décodages à chaque signature. De plus, lors de la réduction, chaque requête de signature implique de modifier $t'!$ valeurs de l'oracle de hachage. Or chacune de ces modifications est susceptible de faire diminuer la probabilité de succès de l'algorithme final. Pour ces raisons, nous ne retenons pas cette solution et préférons tirer des indices aléatoires.

6.4 Preuve de sécurité

Dans cette section, nous prouvons la sécurité du schéma mCFS en utilisant une approche par jeu, selon la méthodologie proposée par Shoup et présentée section 2.1.3. La séquence de jeu proposée relie le jeu ROM-EF-CMA au jeu de difficulté de CGPSD, en utilisant la difficulté du problème GD. Le challenger contrôle l'environnement d'un éventuel adversaire contre le schéma

mCFS et peut donc produire des simulations de ces oracles utilisant l'adversaire pour remporter le jeu CGPSD. Nous énonçons le théorème suivant :

Théorème 6.4.1 (Sécurité de mCFS) *Si les problèmes GD et CGPSD sont respectivement* (ϵ_{GD}, τ') *et* $(\epsilon_{CGPSD}, \tau')$*-difficiles, alors le schéma* CFS modifié mCFS *est* $(\epsilon, \tau, q_\Sigma, q_\mathcal{H})$*-sûr* dans le modèle de l'oracle aléatoire *où :*

$$\epsilon(\tau) = (q_\Sigma + q_\mathcal{H} + 1)\epsilon_{CGPSD}(\tau') + \epsilon_{GD}(\tau') + 2 - \left(1 - \frac{1}{2^{mt}}\right)^{q_\Sigma + q_\mathcal{H} + 1} - \left(1 - \frac{q_\Sigma}{2^{mt}}\right)^{q_\mathcal{H}}$$

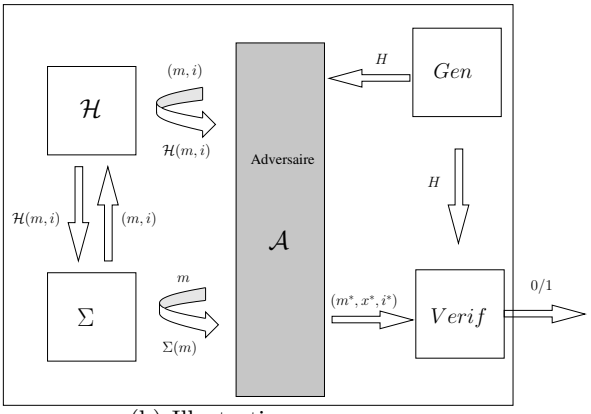
et

$$\tau' = \tau + (q_\Sigma + q_\mathcal{H} + 1) \cdot n^2$$

Preuve – Soient $\mathcal{A}$ un $(\epsilon, \tau, q_\Sigma, q_\mathcal{H})$-adversaire contre le schéma CFS modifié mCFS et Jeu 0 (figure 6.3) le jeu ROM-EF-CMA, adapté à notre schéma : afin de simplifier la preuve, on considère que les requêtes faites à l'oracle de hachage $\mathcal{H}$ portent sur des paires (m, i) de messages et d'indices.

Entrées : Un paramètre de sécurité κ
Données : Un adversaire $\mathcal{A}$
$\mathcal{P} = (m, t) \leftarrow$ CFS.Initialiser(1^κ);
$(SK, PK) \leftarrow$ CFS.GenererClefs$(\mathcal{P})$;
Initialiser les oracles Σ et $\mathcal{H}$;
$(m^*, \sigma^* = (x^*, i^*)) \leftarrow \mathcal{A}^{\Sigma,\mathcal{H}}(PK = H)$;
si *CFS.Vérifier*$(\mathcal{P}, PK, m^*, \sigma^*) =$ *valide* ***et*** Σ *n'a pas fourni* σ^* **alors**
  $\mathcal{A}$ remporte le jeu
sinon
  $\mathcal{A}$ perd le jeu
fin

(a) Présentation algorithmique, point de vue du challenger



(b) Illustration

Figure 6.3 – Jeu 0 : Jeu de sécurité ROM-EF-CMA(κ) pour le schéma CFS modifié

Les oracles $\mathcal{H}$ et Σ maintiennent des listes $\Lambda_\mathcal{H}$ et Λ_Σ respectivement des requêtes qu'ils ont reçues ainsi que les réponses fournies. Rappelons que l'environnement de l'adversaire, et donc en particulier les oracles, est contrôlé par le challenger. Ainsi, tous les éléments autres que l'adversaire peuvent accéder à chacune des listes. La liste $\Lambda_\mathcal{H}$ est étendue pour stocker une valeur de décodage associée à la valeur renvoyée : pour tout message m et tout indice $i, \Lambda_\mathcal{H}(m, i) = (s, x)$ où s est le syndrome renvoyé comme hachage de m et de i et x son décodage. Ainsi, $\Lambda_\Sigma(m) = (x, i)$.

L'objectif de la preuve est d'amener $\mathcal{A}$ à résoudre le problème CGPSD en cas de succès de son attaque. Pour cela, nous modifions petit à petit son environnement (en particulier ses oracles) pour, au final, le plonger dans le jeu de difficulté CGPSD (figure 6.4). La première chose

Entrées : Un décodeur $\mathcal{D}ec$
Données : $n = 2^m$, $k = 2^m - mt$
$H \xleftarrow{R} \mathcal{B}_{mt,2^m}$;
$s \xleftarrow{R} \mathbb{F}_2^{mt}$;
$x \leftarrow \mathcal{D}ec(H, s)$;
si $Hx^T = s$ ***et*** $wt(x) \leq \frac{n-k}{\log_2 n}$ **alors**
  $\mathcal{D}ec$ remporte le jeu
sinon
  $\mathcal{D}ec$ perd le jeu
fin

(a) Jeu CGPSD(m, t)

Entrées : Un paramètre de sécurité κ
Données : Un adversaire $\mathcal{A}$
$c \xleftarrow{R} \{1, \ldots, q_{\mathcal{H}} + q_\Sigma + 1\}$;
$\mathcal{P} = (m, t) \leftarrow \mathsf{CFS.Initialiser}(1^\kappa)$;
$H \xleftarrow{R} \mathcal{B}_{2^m,2^m-mt}$;
Initialiser les oracles $\mathcal{H}'$ and Σ';
$(m^*, \sigma^*, i^*) \leftarrow \mathcal{A}^{\Sigma',\mathcal{H}'}(H)$;
si $\begin{cases} \mathcal{H}'(m^*, i^*) = H^*\sigma^{*T} \\ wt(\sigma^*) \leq t \end{cases}$ ***et*** $\begin{cases} \Sigma' \textit{ n'a pas fourni } \sigma^* \\ \textit{la } c^{eme} \textit{ requête à } \mathcal{H}' \textit{ était } (m^*, i^*) \end{cases}$ **alors**
  $\mathcal{A}$ remporte le jeu
sinon
  $\mathcal{A}$ perd le jeu
fin

(b) Utilisation de l'adversaire pour résoudre CGPSD

FIGURE 6.4 – Utilisation d'un adversaire contre mCFS pour résoudre le problème CGPSD

à faire est donc de remplacer la clef publique donnée à l'adversaire par une matrice aléatoire et d'ajouter le tirage d'un syndrome aléatoire au début de la simulation ; celui-ci sera transmis à l'adversaire au cours de la requête qu'il fera à l'oracle de hachage pour réaliser sa falsification. S'il y parvient, il aura résolu une instance du problème CGPSD.

Mais, si l'on remplace la clef publique, il est évident que l'oracle de signature Σ ne pourra plus utiliser la clef secrète pour réaliser les signatures qu'on lui demande ; il est donc nécessaire d'avoir au préalable remplacé Σ par une simulation lui permettant de feindre le décodage d'un haché. Pour cela, l'oracle de hachage remplace, lors de requêtes destinées à la signature, le tirage d'un syndrome aléatoire par le calcul, à l'aide de la matrice publique, du syndrome d'un mot aléatoire de poids t ; celui-ci est stocké dans une liste. Lors de la signature d'un message, il suffit à Σ de lire la valeur du décodage dans la liste. La suite de la preuve détaille les changements décrits brièvement ci-dessus, remis dans l'ordre et à raison d'une modification par jeu.

Jeu 1 – Dans ce jeu, le challenger remplace l'oracle de hachage $\mathcal{H}$ par une simulation que l'on notera $\mathcal{H}'$, afin de préparer la simulation de l'oracle de signature. $\mathcal{H}'$ utilise une nouvelle liste Λ pour fixer message par message quel indice produira un haché décodable, et donc sur lequel une signature pourra être réalisée (figure 6.5, lignes 1 à 3). Ainsi, deux comportements de l'oracle sont possibles suite à une requête (m, i) : soit $i \neq \Lambda(m)$, soit $i = \Lambda(m)$.

Lorsque $i \neq \Lambda(m)$, alors $\mathcal{H}'$ se comporte exactement comme le ferait un oracle aléatoire maintenant la liste $\Lambda_{\mathcal{H}}$ (figure 6.5, lignes 7 et 8). Lorsque $i = \Lambda(m)$, $\mathcal{H}'$ construit un syndrome décodable à distance t et enregistre la valeur de décodage dans la liste $\mathcal{H}$: tout d'abord il tire aléatoirement un mot de poids t de $\mathbb{F}_2^n$ puis calcule son syndrome, qui sera alors la réponse de l'oracle, et enregistre les valeurs de x et s dans la liste $\Lambda_{\mathcal{H}}$ (figure 6.5, lignes 10 à 12). Bien entendu, avant toute nouvelle sélection, $\mathcal{H}'$ vérifie si une valeur de hachage n'est pas déjà présente dans $\Lambda_{\mathcal{H}}$ pour la requête (figure 6.5, ligne 5).

À la fin de la simulation, l'oracle $\mathcal{H}'$ a produit $q_{\mathcal{H}} + q_\Sigma + 1$ syndromes. Certains d'entre eux sont produits par la partie de l'oracle qui a été modifiée (lorsque $i = \Lambda(m)$). Soit M la variable aléatoire qui représente le nombre de syndromes produits par la partie modifiée. On a :

$$\begin{aligned} \Pr[S_1] &= \Pr\left[(S_1 \cap (M = 0)) \cup (S_1 \cap (M > 0))\right] \\ &\leq \Pr\left[S_1 \cap (M = 0)\right] + \Pr\left[S_1 \cap (M > 0)\right]. \end{aligned}$$

$\Pr[S_1 \cap (M = 0)]$ correspond au cas où l'adversaire remporte Jeu 1 avec des syndromes produits par un oracle aléatoire, ce qui est exactement Jeu 0. D'où, $\Pr[S_1 \cap (M = 0)] = \Pr[S_0]$. $\Pr\left[S_1 \cap (M > 0)\right]$ peut être majorée par $\Pr\left[(M > 0)\right]$ en considérant que l'adversaire remporte le jeu chaque fois qu'il reçoit en réponse à l'une de ses requêtes à l'oracle de hachage un syndrome produit par la partie modifiée de l'oracle. Or, M respecte une loi binomiale de paramètres $q_\Sigma + q_{\mathcal{H}} + 1$ et $1/2^{mt}$ et donc $\Pr\left[(M > 0)\right] = 1 - \left(1 - \frac{1}{2^{mt}}\right)^{q_\Sigma + q_{\mathcal{H}} + 1}$. On a alors

$$\Pr[S_1] \leq \Pr[S_0] + 1 - \left(1 - \frac{1}{2^{mt}}\right)^{q_\Sigma + q_{\mathcal{H}} + 1}$$

Jeu 2 – Dans ce jeu, le challenger remplace l'oracle de signature Σ par une simulation Σ' (figure 6.6) réalisant la signature d'un message sans utiliser la clef publique. Σ' effectue une

Entrées : Une paire (m, i)
Sorties : Un syndrome s

si $\Lambda(m) = \emptyset$ **alors**
  $\Lambda(m) \xleftarrow{R} \{1, \ldots, 2^{mt}\}$;
fin
$(s, x) \leftarrow \Lambda_{\mathcal{H}}(m, i)$;
si $s = \emptyset$ **alors**
  si $i \neq \Lambda(m)$ **alors**
    $s \xleftarrow{R} \mathbb{F}_2^{mt}$;
    $\Lambda_{\mathcal{H}}(m, i) \leftarrow (s, \emptyset)$;
  sinon
    $x \xleftarrow{R} \{w \in \mathbb{F}_2^n | \mathsf{wt}(w) \leq t\}$;
    $s \leftarrow Hx^T$;
    $\Lambda_{\mathcal{H}}(m, i) \leftarrow (s, x)$;
  fin
fin
retourner $\mathcal{H}(m, i) = s$;

FIGURE 6.5 – $\mathcal{H}'$: simulation de l'oracle de hachage $\mathcal{H}$ (Jeu 1)

requête de hachage auprès de $\mathcal{H}'$ pour le couple $(m, \Lambda(m))$. Il s'ensuit que $\mathcal{H}'$ enregistre dans $\Lambda_{\mathcal{H}}$ la valeur de décodage de sa sortie et Σ' peut produire une signature sans posséder la clef secrète. Une fois la signature produite, Σ' efface l'entrée $\Lambda(m)$ de façon à ce que deux requêtes de signature sur le même message ne produisent pas nécessairement le même résultat, reproduisant ainsi le comportement du schéma modifié.

Entrées : Un message m
Sorties : Une signature (x, i)

si $\Lambda(m) = \emptyset$ **alors**
  $\Lambda(m) \xleftarrow{R} \{1, \ldots, 2^{mt}\}$;
fin
$\mathcal{H}'(m, \Lambda(m))$;
$(s, x) \leftarrow \Lambda_{\mathcal{H}}(m, \Lambda(m))$;
$\Lambda(m) \leftarrow \emptyset$;
retourner $\Sigma(m) = (x, i)$;

FIGURE 6.6 – Σ' : simulation de l'oracle de signature Σ (Jeu 2)

Cependant, lors de l'exécution de son attaque, $\mathcal{A}$ peut interroger $\mathcal{H}'$ sur un message dont il a déjà obtenu une signature auprès de Σ'. Dans ce cas, avec probabilité au plus $\frac{q_\Sigma}{2^{mt}}$, $\mathcal{H}'$ renvoie un syndrome produit par la partie modifiée. Soit S la variable aléatoire représentant le nombre de ces syndromes. On a donc

$$\begin{aligned}
\Pr[S_2] &= \Pr\left[(S_2 \cap (S=0)) \cup (S_2 \cap (S>0))\right] \\
&\leq \Pr[S_2 \cap (S=0)] + \Pr[S_2 \cap (S>0)] \\
&\leq \Pr[S_1] + \Pr[S>0] \\
&\leq \Pr[S_1] + 1 - \left(1 - \tfrac{q_\Sigma}{2^{mt}}\right)^{q_\mathcal{H}}
\end{aligned}$$

Jeu 3 – Puisque maintenant l'oracle de signature n'utilise plus la clef secrète pour calculer sa réponse mais uniquement la clef publique, il est possible de remplacer la génération de cette dernière par le tirage d'une matrice aléatoire.

Dans ce jeu le challenger remplace donc la sélection aléatoire d'un code de Goppa par la sélection aléatoire d'un code binaire aléatoire. Il est alors possible de construire le distingueur $\mathcal{D}$ représenté sur la figure 6.7.

Entrées : Une matrice de parité H de taille $(n-k) \times n$
Données : Un adversaire $\mathcal{A}$ contre mCFS
Sorties : Un bit b

$t \leftarrow \frac{n-k}{\log_2 n}$;
Initialiser les oracles $\mathcal{H}'$ et Σ';
$(m^*, x^*, i^*) \leftarrow \mathcal{A}^{\Sigma', \mathcal{H}'}(H)$;
si $\mathcal{H}'(m^* \| i^*) = Hx^{*T}$ ***et*** $wt(x^*) \leq t$ ***et*** Σ' *n'a pas fourni* (x^*, i^*) **alors**
  retourner 1
sinon
  retourner 0
fin

FIGURE 6.7 – Distingueur $\mathcal{D}$ (Jeu 3)

Si H est une matrice de parité d'un code de Goppa, $\mathcal{D}$ se comporte comme le Jeu 2 et donc $\Pr\left[\mathsf{Exp}^0_{\mathsf{GD},\mathcal{D}}(n,k)=1\right] = \Pr[S_2]$. En revanche, si H est une matrice de parité d'un code binaire aléatoire, $\mathcal{D}$ se comporte comme le Jeu 3 et alors $\Pr\left[\mathsf{Exp}^1_{\mathsf{GD},\mathcal{D}}(n,k)=1\right] = \Pr[S_3]$. Il s'ensuit que $\mathtt{Adv}_{\mathsf{GD}(n,k)}(\mathcal{D}) = |\Pr[S_3] - \Pr[S_2]|$. Or, par hypothèse, CGPSD est $(\tau_{\mathsf{GD}}, \epsilon_{\mathsf{GD}})$-difficile et donc

$$|\Pr[S_3] - \Pr[S_2]| \leq \epsilon_{\mathsf{GD}}(\tau')$$

où τ' est le temps total d'exécution de la simulation.

Jeu 4 – Dans ce jeu (figure 6.8), le challenger modifie la condition de victoire. Ce jeu est conditionné par le fait que l'adversaire effectue sa contrefaçon à partir du résultat d'une requête de hachage précise. Ceci permettra par la suite de substituer aux valeurs sur lesquelles l'adversaire réalise sa contrefaçon le challenge du problème CGPSD.

Pour y parvenir, le challenger commence par tirer une requête aléatoire c dans $\{1, \ldots, q_{\mathcal{H}} + q_\Sigma + 1\}$; $\mathcal{A}$ remporte ce jeu si, en plus des conditions existantes dans les jeux précédents, la c^{eme} requête à l'oracle de hachage portait sur (m^*, i^*).

Entrées : Un paramètre de sécurité κ
Données : Un adversaire $\mathcal{A}$
$c \xleftarrow{R} \{1, \ldots, q_{\mathcal{H}} + q_\Sigma + 1\}$;
$\mathcal{P} = (m, t) \leftarrow \mathsf{CFS.Initialiser}(1^\kappa)$;
$H \xleftarrow{R} \mathcal{B}_{2^m, 2^m - mt}$;
Initialiser les oracles $\mathcal{H}'$ and Σ';
$(m^*, \sigma^*, i^*) \leftarrow \mathcal{A}^{\Sigma', \mathcal{H}'}(H)$;
si $\begin{cases} \mathcal{H}'(m^*, i^*) = H\sigma^{*T} \\ \mathsf{wt}(\sigma^*) \leq t \end{cases}$ ***et*** $\begin{cases} \Sigma' \textit{ n'a pas fourni } \sigma^* \\ \textit{la } c^{eme} \textit{ requête à } \mathcal{H}' \textit{ était } (m^*, i^*) \end{cases}$ **alors**
  $\mathcal{A}$ remporte le jeu
sinon
  $\mathcal{A}$ perd le jeu
fin

FIGURE 6.8 – Jeu 4 : Ajout d'une condition de victoire supplémentaire

Cet événement, indépendant du choix fait par l'adversaire, a une probabilité de $\frac{1}{q_{\mathcal{H}} + q_\Sigma + 1}$. On a donc

$$\Pr[4] = \frac{\Pr[S_3]}{q_{\mathcal{H}} + q_\Sigma + 1}$$

Jeu 5 – Dans ce jeu, le challenger modifie, encore une fois, l'oracle de hachage de façon à ce qu'à la c^{eme} requête qu'il reçoit, celui-ci renvoie un syndrome aléatoire s^* de $\mathbb{F}_2^{mt}$. L'espace de probabilité n'est pas modifié et donc $\Pr[S_5] = \Pr[S_4]$.

La restriction faite dans le jeu précédent nous assure que la contrefaçon sera réalisée à partir du résultat s^* de la c^{eme} requête (m^*, i^*) faite à l'oracle de hachage. Il s'ensuit que si $\mathcal{A}$ remporte le jeu, on a

$$\begin{cases} H\sigma^{*T} = \mathcal{H}'(m^*, i^*) = s^* \\ \mathsf{wt}(\sigma^*) \leq t \end{cases}$$

Cela signifie que l'adversaire plongé dans son environnement simulé remporte le jeu CGPSDet donc $\Pr[S_5] = \mathtt{Succ}_{\mathsf{CGPSD}(n,k)}(\mathcal{A}^{\Sigma',\mathcal{H}'})$. Or, par hypothèse, CGPSDest $(\epsilon_{\mathsf{CGPSD}}, \tau_{\mathsf{CGPSD}})$-difficile et donc

$$\Pr[S_5] \leq \epsilon_{\mathsf{CGPSD}}(\tau')$$

où τ' est le temps total d'exécution de la simulation.

Conclusion – On obtient de la séquence de jeux précédente les égalités et inégalités suivantes :

1. $\Pr[S_0] = Succ^{EF-CMA}_{mCFS}(\mathcal{A})$
2. $|\Pr[S_0] - \Pr[S_1]| \leq 1 - \left(1 - \frac{1}{2^{mt}}\right)^{q_{\mathcal{H}}+q_\Sigma+1}$
3. $|\Pr[S_1] - \Pr[S_2]| \leq 1 - \left(1 - \frac{q_\Sigma}{2^{mt}}\right)^{q_{\mathcal{H}}}$
4. $\Pr[S_2] = \Pr[S_3]$
5. $|\Pr[S_3] - \Pr[S_4]| \leq \epsilon_{\mathsf{GD}}(\tau')$
6. $\frac{\Pr[S_4]}{q_{\mathcal{H}}+q_\Sigma+1} = \Pr[S_5] = \Pr[S_6] \leq \epsilon_{\mathsf{CGPSD}}(\tau')$

Des lignes 2, 3, 4 et 5, on obtient par inégalité triangulaire

$$|\Pr[S_0] - \Pr[S_4]| \leq \epsilon_{\mathsf{GD}} + 2 - \left(1 - \frac{1}{2^{mt}}\right)^{q_{\mathcal{H}}+q_\Sigma+1} - \left(1 - \frac{q_\Sigma}{2^{mt}}\right)^{q_{\mathcal{H}}}$$

Or, d'après la ligne 6, $\Pr[S_4] \leq (q_{\mathcal{H}} + q_\Sigma + 1)\epsilon_{\mathsf{CGPSD}}(\tau')$; on a donc

$$Succ^{EF-CMA}_{mCFS}(\mathcal{A}) \leq (q_{\mathcal{H}} + q_\Sigma + 1)\epsilon_{\mathsf{CGPSD}}(\tau') + \epsilon_{\mathsf{GD}}(\tau') + 2 - \left(1 - \frac{1}{2^{mt}}\right)^{q_{\mathcal{H}}+q_\Sigma+1} - \left(1 - \frac{q_\Sigma}{2^{mt}}\right)^{q_{\mathcal{H}}}$$

Le temps de calcul τ' de la simulation est essentiellement le temps τ de calcul de l'adversaire $\mathcal{A}$ auquel il faut ajouter le calcul des $q_{\mathcal{H}} + q_\Sigma + 1$ simulation de hachage. Ceux-ci nécessitent au plus $q_{\mathcal{H}} + q_\Sigma + 1$ calculs de syndromes nécessitant chacun $n(n-k)$ opérations binaires, ce qui nous fournit la formule pour τ' et conclut la preuve. $\diamondsuit$

6.5 Interprétation

Si l'on est en mesure de majorer les probabilités $\epsilon_{\mathsf{CGPSD}}(\tau')$ et $\epsilon_{\mathsf{GD}}(\tau')$, la preuve de sécurité précédente nous permettra donc de majorer la probabilité de succès d'un adversaire contre le schéma mCFS.

Remarquons tout d'abord que la quantité

$$E = 2 - \left(1 - \frac{1}{2^{mt}}\right)^{q_{\mathcal{H}}+q_\Sigma+1} - \left(1 - \frac{q_\Sigma}{2^{mt}}\right)^{q_{\mathcal{H}}}$$

peut être rendue négligeable. En effet, en utilisant le développement limité de $(1+x)^a$ au voisinage de 0, on obtient :

$$\begin{aligned} E &= 2 - 1 + \tfrac{q_{\mathcal{H}}+q_\Sigma+1}{2^{mt}} - 1 + \tfrac{q_{\mathcal{H}}q_\Sigma}{2^{mt}} + O(\tfrac{1}{2^{2mt}}) \\ &= \tfrac{q_{\mathcal{H}}+q_\Sigma+1}{2^{mt}} + \tfrac{q_{\mathcal{H}}q_\Sigma}{2^{mt}} + O(\tfrac{1}{2^{2mt}}). \end{aligned}$$

Il suffit donc de s'assurer que $\frac{q_{\mathcal{H}} q_\Sigma}{2^{mt}}$ est négligeable devant $2^{-\kappa}$, où κ est le paramètre de sécurité (traditionnellement $\kappa = 80$ ou $\kappa = 100$). Le nombre de requêtes à l'oracle de chiffrement qu'un éventuel adversaire peut effectuer peut être arbitrairement borné. Pour cela, il suffit de restreindre le nombre de signatures autorisées pour une même clef publique. Une valeur classique est $q_\Sigma = 2^{30}$. En revanche, le nombre de requêtes autorisée à l'oracle de hachage n'est limité que par la puissance de calcul de l'adversaire. Un oracle de hachage étant considéré en temps constant, on considère $q_{\mathcal{H}} = 2^\kappa$.

Par exemple, en considérant les paramètres $m = 22$ et $t = 9$ fournis par la borne de M. Finiasz et N. Sendrier (présentée section 3.2.3), pour $\kappa = 80$ et les valeurs de $q_{\mathcal{H}}$ et de q_Σ précédentes, on a $\frac{q_{\mathcal{H}} q_\Sigma}{2^{mt}} = 2^{-88}$ et $\frac{q_{\mathcal{H}} + q_\Sigma + 1}{2^{mt}} \simeq 2^{-118}$.

Les paramètres $m = 22$ et $t = 9$ ci-dessus sont obtenus en considérant que la réduction de sécurité est *fine*, c'est-à-dire que pour tout adversaire $\mathcal{A}$, $Succ^{EF-CMA}_{mCFS}(\mathcal{A}) \simeq \epsilon_{\mathsf{CGPSD}}(\tau') + \epsilon_{\mathsf{GD}}(\tau')$ et en négligeant $\epsilon_{\mathsf{GD}}(\tau')$, comme cela était fait dans [CFS01]. Or, notre preuve nous permet uniquement de s'assurer que la probabilité $Succ^{EF-CMA}_{mCFS}(\mathcal{A})$ sera $q_{\mathcal{H}} + q_\Sigma + 1$ fois plus grande que la probabilité de résoudre CGPSD. Il faudrait donc choisir des paramètres de façon à ce que $\epsilon_{\mathsf{CGPSD}}(\tau') \leq \frac{2^{-\kappa}}{q_{\mathcal{H}} + q_\Sigma + 1}$, ce qui rendrait le schéma impraticable. Par exemple, pour $t = 11$ (ce qui représente un peu moins de 40 millions de décodages par signature), le code serait de longueur 2^{39}.

Cependant, il n'existe pas à l'heure actuelle de preuve d'optimalité de notre réduction. L'existence d'une preuve de sécurité fine du schéma mCFS n'est donc pas exclue. La preuve de sécurité ci-dessus doit donc être interprétée plus comme un argument supplémentaire en faveur de la soldité du schéma que comme une prescription de sécurité concrète.

Chapitre 7

Nouveau schéma de signature de cercle à seuil prouvé sûr

Ce chapitre a fait l'objet de la publication [DV09], en collaboration avec D. Vergnaud.

Sommaire

7.1 Signatures de cercle à seuil

Nous avons vu à la section 1.2.2 que les schémas de signature standards visent à reproduire les qualités de la signature manuscrite. Cependant, les propriétés de sécurité offertes par la signature électronique peuvent, lors de certaines applications, s'avérer trop fortes ou à l'inverse insuffisantes. Ainsi, de nouveaux schémas de signature sont apparus, permettant soit d'alléger une propriété traditionnelle, soit d'en ajouter de nouvelles. C'est le cas de la signature de cercle qui permet de réaliser une authentification anonyme. A première vue, ces deux qualités semblent incompatibles, pourtant il existe de nombreuses applications nécessitant un certain anonymat dans la signature, lors du vote électronique par exemple.

Ce concept est apparu pour la première fois en 1991 lorsque D. Chaum et E. van Heyst ont proposé l'idée de la signature de groupe [Cv92]. Celle-ci permet à un membre d'un groupe de signer anonymement un message au nom du groupe. Au cœur d'un schéma de signature de groupe, se trouve un *manager* qui a la charge de l'ajout de nouveaux membres et qui dispose de la possibilité de lever l'anonymat du signataire en cas de conflit. Les *signatures de cercle*, formalisées par R. Rivest, A. Shamir et Y. Tauman dans [RST01], sont similaires aux signatures de groupe mais diffèrent en deux points essentiels : tout d'abord l'anonymat du signataire ne peut être levé ; ensuite, tout ensemble d'utilisateurs — pourvu qu'ils disposent d'une clef publique — peut être utilisé pour former le groupe, sans nécessiter d'opération supplémentaire. Les membres du *cercle* ainsi formé n'ont pas besoin d'intervenir lors de la génération de la signature, et peuvent ainsi ne même pas être conscients de faire parti du cercle : le cercle est défini entièrement « à la volée » par le signataire.

Depuis 1991, les signatures de cercle constituent un domaine de recherche actif et de nombreuses applications ont été proposées : [RST01, Nao02, DKNS04] par exemple. L'application originale proposée par R. Rivest, A. Shamir et Y. Taumam était la dénonciation, illustrée par l'exemple d'un membre du gouvernement souhaitant révéler à la presse certaines irrégularités sans pour autant mettre en jeu sa position. Les signatures de cercle lui permettent d'atteindre cet objectif : il lui suffit de produire une signature de cercle à l'aide des clefs publiques de chacun des membres du gouvernement. Ainsi, les journalistes pourront s'assurer que le message provient bien de l'un des ministres, sans pour autant savoir lequel de ceux-ci a réellement révélé l'information.

Une application populaire — et peut-être moins sujette à controverse — des signatures de cercle est la signature à vérificateur désigné [JSI96]. Un expéditeur souhaitant que seul le destinataire puisse vérifier sa signature peut produire une signature pour un cercle de deux personnes : lui-même et le destinataire. Le destinataire du message, sachant qu'il n'a pas signé le message, peut s'assurer que l'expéditeur du message est bien celui attendu ; il ne peut cependant pas transférer cette conviction puisqu'il fait parti du cercle et pourrait donc avoir signé le message

lui-même.

En 2002, E. Bresson, J. stern et M. Szydlo ont étendu le concept de ces signatures aux *signatures de cercle à seuil* [BSS02] où un ensemble de ℓ signataires coopère pour produire la signature, sans pour autant révéler leurs identités au sein du cercle. Fournir ℓ signatures de cercle ne permet clairement pas de s'assurer que le message ait été signé par différents signataires. Un schéma de signature de cercle à seuil doit donc prouver qu'effectivement ℓ signataires ont participé à la création de la signature.

Le premier schéma de signature de cercle fondé sur la théorie des codes a été proposé en 2007 par D. Zheng and X. Li et K. Chen [ZLC07]. Celui-ci permet d'obtenir des signatures de petite taille ($144 + 126N$ bits pour une sécurité d'aujourd'hui environ 2^{63}, où N est la taille du cercle) mais utilise des clefs publiques de grande taille (environ $1,2$ Mo). De plus, le design de ce protocole ne permet pas l'ajout d'un seuil. Plus récemment, C. Aguillar, P-L. Cayrel et P. Gaborit ont introduit le premier schéma de signature de cercle à seuil fondé sur la théorie des codes [ACG08]. Ce dernier utilise des clefs de petite taille (347 bits pour une sécurité en 2^{80}) mais produit des signatures de grande taille (environ 20 Ko par membre du cercle).

Nous présentons ici un nouveau schéma de signature de cercle à seuil fondé sur les codes correcteurs d'erreurs. Ce schéma permet d'obtenir des signatures de petite taille ($414N - 144\ell$ bits et $579N - 198\ell$ bits, où N est la taille du cercle et ℓ est le nombre de signataires, pour des sécurités respectives en 2^{63} et 2^{80}) mais utilise en revanche des clefs de grande taille (environ $1,2$ Mo et 99 Mo respectivement). Ce schéma, réalisé en collaboration avec D. Vergnaud, a fait l'objet de la publication [DV09] à IMACC 2009 (Twelfth IMA International Conference on Cryptography and Coding).

7.1.1 Définition formelle

Dans la définition d'un schéma de signature de cercle à seuil , nous désignerons par *cercle* tout ensemble d'utilisateurs $\mathcal{N} = \{1, \ldots, N\}$ tel que tout membre u possède une paire de clefs publique et privée (PK_u, SK_u). Pour tout sous-ensemble $\mathcal{L} \subset \mathcal{N}$, nous désignons par $\mathcal{SK}_{\mathcal{L}}$ l'ensemble $\{SK_i \mid i \in \mathcal{L}\}$ des clefs privées des utilisateurs inclus dans $\mathcal{L}$.

Définition 7.1.1 (Schéma de signature de cercle à seuil) *Un schéma* TRS *de signature de cercle à seuil est défini par quatre algorithmes formant un ensemble consistant :* TRS.Initialiser, TRS.GénérerClefs, TRS.Signer *et* TRS.Vérifier *qui, respectivement, initialise les paramètres du schéma, génère les clefs d'un utilisateur, signe un message et vérifie une signature. Plus formellement :*

- TRS.Initialiser(1^κ) *reçoit en entrée un paramètre de sécurité* κ *et renvoie un ensemble* $\mathcal{P}$ *de paramètres ;*

- *TRS.GénérerClefs*$(\mathcal{P})$ *reçoit en entrée un jeu de paramètres* $\mathcal{P}$ *et renvoie une paire* (SK_u, PK_u) *de clefs secrète et publique ;*
- *TRS.Signer*$(\mathcal{P}, M, \mathcal{N}, \mathcal{SK}_{\mathcal{L}})$ *reçoit en entrée un jeu de paramètres* $\mathcal{P}$*, un message* M *un ensemble d'utilisateurs et un ensemble* $\mathcal{SK}_{\mathcal{L}}$ *formé des clefs privées des membres d'un ensemble* $\mathcal{L} \subset \mathcal{N}$ *(*$|\mathcal{L}| = \ell < |\mathcal{N}| = N$*). Cet algorithme, qui peut être interactif renvoie une* ℓ*-signature* σ *pour le cercle* $\mathcal{N}$ *;*
- *TRS.Vérifier*$(\mathcal{P}, M, \sigma, \mathcal{N})$ *reçoit en entrée un jeu de paramètres* $\mathcal{P}$*, un message* M*, une signature potentielle* σ *et un cercle* $\mathcal{N}$*. Il renvoie* valide *si* σ *est une* ℓ*-signature valide du message* M *relativement aux clefs publiques des membres de* $\mathcal{N}$ *et renvoie* invalide *si ce n'est pas le cas.*

Ces algorithmes sont consistants si, pour tout paramètre de sécurité $\kappa > 0$*, pour tout jeu de paramètres* $\mathcal{P} \leftarrow$ *Setup*(1^{κ})*, pour tout cercle* $\mathcal{N}$ *et tout sous-ensemble* $\mathcal{L}$ *tels que* $\mathcal{L} \subset \mathcal{N}$*,* $|\mathcal{L}| = \ell$*,* $|\mathcal{N}| = N$ *on a :*

$$\forall M \in \{0,1\}^*, \forall \sigma \leftarrow \textit{TRS.Signer}(\mathcal{P}, M, \mathcal{N}, \mathcal{SK}_{\mathcal{L}}), \textit{TRS.Vérifier}(\mathcal{P}, M, \sigma, \mathcal{N}) = \textit{valide}.$$

7.1.2 Modèle de sécurité

Un schéma de signature de cercle à seuil doit satisfaire deux exigences de sécurité : il doit préserver l'anonymat des signataires tout en garantissant qu'il est impossible de produire une ℓ-signature en utilisant moins de ℓ clefs privées.

Anonymat Informellement, la condition d'anonymat des signatures de cercle signifie qu'il doit être impossible à un adversaire de découvrir les membres du cercle ayant réellement participé à la création d'une ℓ-signature donnée. Le plus souvent, cet anonymat est montré de façon inconditionnelle, c'est-à-dire en n'utilisant aucune autre hypothèse que les outils de la théorie de l'information.

Plus formellement, on considère une PPTM, $\mathsf{Simu}(\mathcal{P}, M, \mathcal{N}, \mathcal{L}, \mathcal{SK}_{\mathcal{L}\setminus\{i\}})$ désignée comme une simulation, qui prend en entrée un jeu de paramètres $\mathcal{P}$, un message M, un cercle $\mathcal{N}$, un sous ensemble $\mathcal{L} \subset \mathcal{N}$ de taille ℓ et l'ensemble $\mathcal{SK}_{\mathcal{L}\setminus\{i\}}$ des clefs secrètes des membres de $\mathcal{L}$ à l'exception d'un utilisateur $i \in \mathcal{L}$. Elle tente de simuler une ℓ-signature de cercle pour les clefs publiques des membres de $\mathcal{N}$. Un schéma de signature de cercle ℓ-parmi-$\mathcal{N}$ est dit *anonyme* si pour tout paramètre de sécurité $\kappa > 0$, pour tout jeu de paramètres $\mathcal{P} \leftarrow \mathsf{TRS.Initialiser}(\kappa)$, tout cercle $\mathcal{N}$ tel que tout utilisateur $i \in \mathcal{N}, (Sk_i, PK_i) \leftarrow \mathsf{TRS.GénérerClefs}(\mathcal{P})$, tout sous-ensemble $\mathcal{L} \subset \mathbb{N}$, tout utilisateur $u \in \mathcal{L}$ et tout message M,

$$\Pr[\mathsf{Simu}(\mathcal{P}, M, \mathcal{N}, \mathcal{L}, \mathcal{SK}_{\mathcal{L}\setminus\{i\}}) = \sigma] = \Pr[\mathsf{TRS.Signer}(\mathcal{P}, M, \mathcal{N}, \mathcal{SK}_{\mathcal{L}}) = \sigma].$$

Résistance aux contrefaçons Comme pour les signatures standards, un schéma de signature de cercle à seuil doit résister à une contrefaçon existentielle lors d'une attaque à messages choisis. Le modèle d'attaque considéré est le suivant : un ℓ-adversaire $\mathcal{A}$ (qui est bien entendu en possession des clefs publiques des membres d'un cercle $\mathcal{N}$ de taille N) tente de produire une ℓ-signature de cercle, nommée ici σ^*, valide pour un message M^* et le cercle $\mathcal{N}$ de son choix avec les paramètres $\mathcal{P} \leftarrow \mathsf{TRS.Initialiser}(\kappa)$. Pour y parvenir il peut obtenir adaptivement :

- les clefs privées de $\ell-1$ utilisateurs par le biais de $\ell-1$ requêtes à un oracle de corruption Υ ;
- les signatures de q_Σ messages de son choix par un sous-ensemble de ℓ utilisateurs de son choix relativement au cercle $\mathcal{N}$ par le biais d'un oracle Σ ;

ainsi que, lorsque la preuve se place respectivement dans les modèles de l'oracle aléatoire ou du chiffrement idéal (présentés à la section 2.1.1) :

- les hachés de $q_\mathcal{H}$ messages de son choix auprès d'un oracle aléatoire $\mathcal{H}$,
- les chiffrés ou déchiffrés par une permutation aléatoire de $q_\mathcal{E}$ valeurs de son choix grâce à un oracle de chiffrement $\mathcal{E}$.

Bien entendu, $\mathcal{A}$ n'est pas autorisé à produire comme contrefaçon une signature qu'il a obtenu de l'oracle de signature. On appelle succès d'un adversaire $\mathcal{A}$ contre le schéma TRS de paramètre de sécurité κ, noté $\mathtt{Succ}^{\mathsf{TRS}(\kappa)}_{EF-CMA}(\mathcal{A})$, la probabilité qu'il produise une contrefaçon valide.

7.2 Construction générique de Bresson, Stern et Szydlo

En 1979, A. Shamir a proposé d'utiliser l'interpolation polynomiale de Lagrange, rappelée ci-dessous (Théorème 7.2.1), pour permettre le partage d'un secret entre plusieurs utilisateurs, de façon à ce qu'un nombre minimum de participants aient à collaborer pour pouvoir reconstruire ce secret [Sha79]. Tant qu'un nombre minimal de participants ne collabore pas à la reconstruction du secret, celui-ci est inconditionnellement préservé.

Théorème 7.2.1 (Interpolation polynomiale de Lagrange) *Soit $\mathbb{F}$ un ensemble fini et N paires $(x_1, y_1), \ldots, (x_N, y_N)$ de $\mathbb{F}^2$ tel que tous les x_i soient distincts. Il existe un unique polynôme p de degré $N-1$ dans $\mathbb{F}[X]$ tel que pour tout indice i, $p(x_i) = y_i$. Ce polynôme est :*

$$p(x) = \sum_{i=1}^{N} y_i \prod_{j=1, j\neq i}^{N} \frac{x - x_j}{x_i - x_j}.$$

Lors de l'introduction des schémas de signature de cercle à seuil, E. Bresson, J. Stern et M. Szydlo ont proposé une construction générique pour obtenir, à partir d'une fonction à sens unique à trappe, un schéma de signature de cercle, dans le modèle du chiffrement idéal [BSS02]. Celle-ci utilise également l'interpolation polynomiale de Lagrange pour protéger l'anonymat des signataires.

Dans le schéma qu'ils ont proposé, chaque membre du cercle est associé à une fonction à sens unique pour laquelle il possède une trappe (sa clef privée). Les ℓ signataires commencent par produire $N-\ell$ valeurs à l'aide des fonctions à sens unique de chacun des membres du cercle non signataires. Ils en déduisent un polynôme d'interpolation. Celui-ci permet alors de calculer ℓ valeurs associées à chacun des signataires. Ceux-ci utilisent alors leur clef privée pour inverser leur fonction à sens unique sur ces valeurs et ainsi produire la signature. Cette dernière est composée du polynôme et des N valeurs obtenues à l'aide des fonctions à sens unique. La vérification consiste simplement à s'assurer de l'égalité entre les valeurs du polynôme et l'application des fonctions à sens unique. Une étape de chiffrement symétrique (dans le modèle du chiffrement idéal) est ajoutée pour rendre aléatoires les valeurs interpolées. Intuitivement, l'anonymat des signataires est garanti par l'impossibilité de distinguer les valeurs obtenues par évaluation du polynôme ; tandis que la résistance aux contrefaçons est assurée par la solidité de la fonction à sens unique.

Soient $\mathcal{N}$ l'ensemble des N utilisateurs formant le cercle, p et q deux entiers, $\mathcal{F}$ un ensemble de fonctions à sens unique et leurs trappes $(f_i, \mathcal{T}_i)$ telles que $f_i : \{0,1\}^p \to \{0,1\}^q$, $\mathcal{H} : \{0,1\}^* \to \{0,1\}^q$ une fonction de hachage et $(E_{k,i})$ une famille doublement indexée de permutations aléatoires chiffrant des messages de q bits à l'aide de clefs de q_0 bits. Pour chaque utilisateur $i \in \mathbb{N}$, l'algorithme de génération des clefs a fourni une fonction à sens unique f_i (pour le paramètre de sécurité κ) rendue publique et une trappe associée T_i gardée secrète. Un ensemble $\mathcal{L} \subset \mathbb{N}$ de taille ℓ de signataires signe alors un message M de la façon suivante :

1. Fixer une clef $k \leftarrow \mathcal{H}(M)$ et une valeur initiale $y_0 \leftarrow \mathcal{H}(M \| \mathcal{N})$.
2. Pour chaque utilisateur $i \in \mathcal{N} \backslash \mathcal{L}$:
   (a) Tirer aléatoirement une valeur $x_i \xleftarrow{R} \{0,1\}^p$;
   (b) Calculer $y_i \leftarrow f_i(x_i)$.
3. Calculer un polynôme d'interpolation

$$P \in \mathbb{F}_{2^q}[X] \text{ tel que } \begin{cases} d(P) = N - \ell \\ P(0) = y_0 \\ P(i) = E_{k,i}(y_i) \text{ pour tout } i \in \mathbb{N} \backslash \mathcal{L} \end{cases}$$

4. Pour tout utilisateur $i \in \mathcal{L}$:
   (a) Calculer $y_i = E_{k,i}^{-1}(P(i))$;
   (b) Calculer $x_i = f_i^{-1}(y_i)$ à l'aide de la trappe $\mathcal{T}_i$.
5. Renvoyer $\sigma = (x_1, \ldots, x_N, P)$ comme ℓ-signature du message M pour le cercle $\mathcal{N}$.

Une signature $\sigma = (x_1, \ldots, x_N, P)$ peut être vérifiée relativement au cercle $\mathcal{N}$ en s'assurant que $P(0) = \mathcal{H}(M \| \mathcal{N})$ et que pour tout utilisateur $i \in \mathcal{N}$, $P(i) = E_{\mathcal{H}(M),i}(f_i(x_i))$.

7.3 Notre schéma

Comme nous l'avons déjà vu, les fonctions à sens unique utilisées dans les chiffrements de McEliece et de Niederreiter ne sont pas surjectives. Or, la construction d'E. Bresson, J Stern et M. Szydlo précédente suppose que la fonction à sens unique utilisée est bijective. Il est donc nécessaire d'adapter le schéma à ce cas particulier.

Afin d'obtenir des signatures les plus courtes possibles, nous souhaitons utiliser la fonction de chiffrement de Niederreiter comme fonction à sens unique. Un polynôme P serait alors interpolé à partir de syndromes (qui sont alors vus comme des entiers de $\mathbb{F}_2^{n-k}$) de mots de poids t aléatoires dans le code de longueur n et de dimension k associé à chacun des $N-\ell$ membres non-signataires du cercle . Chacun des ℓ membres signataires devrait ensuite décoder des syndromes obtenus par évaluation du polynôme P. Or, seule une petite proportion P_{dec} de ces syndromes est décodable. Trouver un polynôme qui fournirait ℓ syndromes décodables nécessiterait donc en moyenne $\left(\frac{1}{P_{dec}}\right)^{\ell}$ essais et donc au moins autant de tentatives de décodage ; ce qui n'est évidemment pas raisonnable. Il est donc nécessaire que chacun des signataires puisse dériver un syndrome décodable de l'évaluation du polynôme.

Pour ce faire, nous nous inspirons de la technique employée par N. Courtois, M. Finiasz et N. Sendrier dans [CFS01] : un nouveau syndrome, obtenu par hachage du message avec une valeur aléatoire, est ajouté aux valeurs du polynôme $P(i)$ à décoder pour former un nouveau syndrome s, jusqu'à ce que ce dernier soit décodable. La construction des valeurs d'interpolation est bien entendu adaptée en conséquence. Procéder ainsi permet d'effectuer en moyenne $\ell(\frac{1}{P_{dec}})$ décodages pour l'ensemble des signataires. Afin de maintenir un nombre raisonnable de décodages, nous utilisons des codes de Goppa de faible capacité de correction $t \leq 10$ dont la probabilité de décodage est alors environ $\frac{1}{t!}$.

Ainsi notre schéma, que nous désignons dans la suite par DV, se place à la fois dans les deux modèles équivalents que sont le modèle de l'oracle aléatoire et le modèle du chiffrement idéal (voir section 2.1.1).

7.3.1 Description

Le schéma DV est décrit par les quatre algorithmes suivants, formant un ensemble consistant :

– DV.Initialiser(1^{κ})

1. Déterminer les paramètres m et t nécessaires pour assurer une sécurité adéquate.
2. Choisir une fonction de hachage $\mathcal{H} : \{0,1\}^* \to \mathbb{F}_2^{mt}$.
3. Choisir une fonction de hachage $\mathcal{H}_{\kappa} : \{0,1\}^* \to \mathbb{F}_2^{\kappa}$.
4. Choisir une famille doublement indexée $(E_{k,i})$. de permutations aléatoires chiffrant des messages de mt bits avec des clefs k de longueur κ.

5. Renvoyer $\mathcal{P} = (m, t, \mathcal{H}, \mathcal{H}_\kappa, (E_{k,i}))$.

Pour faciliter la lecture de la suite, notons $n = 2^m$ et $k = n - mt$.

– DV.GénérerClefs($\mathcal{P}$) procède comme Niederreiter.GénérerClefs($\mathcal{P}$) :

1. Tirer aléatoirement un code $\mathcal{C} \xleftarrow{R} \mathcal{G}_{m,t}$.
2. Calculer une matrice de parité H du code $\mathcal{C}$.
3. Construire un algorithme Δ de décodage par syndrome à distance t pour le code $\mathcal{C}$:
$$\begin{cases} \forall e \in \mathbb{F}_2^n \text{ tel que } wt(e) \leq t,\ \Delta(He^T) = e^T. \\ \forall e \in \mathbb{F}_2^n \text{ tel que } wt(e) > t,\ \Delta(He^T) = \emptyset. \end{cases}$$
4. Renvoyer le couple de clef publique/clef privée (PK, SK) où $PK = H$ et $SK = \Delta$.

Dans la suite, nous considérerons que tout membre i d'un cercle $\mathcal{N}$ est associé à une paire de clefs (PK_i, SK_i) où $SK_i = \Delta^{(i)}$ et $SK_i = H^{(i)}$.

– DV.Signer($\mathcal{P}, M, \mathcal{N}, \mathcal{SK}_\mathcal{L}$)

1. Initialisation :
   (a) Calculer une clef $k \leftarrow \mathcal{H}_\kappa(M)$.
   (b) Calculer une valeur initiale $y_0 \leftarrow \mathcal{H}(M \| \mathcal{N})$.
2. Pour tout membre non-signataire $i \in \mathcal{N} \backslash \mathcal{L}$:
   (a) tirer aléatoirement $x_i \xleftarrow{R} \{x \in \mathbb{F}_2^{2^m} | wt(x) \leq t\}$;
   (b) tirer aléatoirement $r_i \xleftarrow{R} \{1, \ldots, 2^{mt}\}$;
   (c) calculer $y_i \leftarrow H^{(i)} x_i^T + \mathcal{H}(M \| r_i)$.
3. Calculer un polynôme d'interpolation
$$P \in \mathbb{F}_{2^{mt}}[X] \text{ tel que } \begin{cases} d(P) = |\mathcal{N}| - |\mathcal{L}| \\ P(0) = y_0 \\ P(i) = E_{k,i}(y_i) \text{ pour tout } i \in N \backslash L \end{cases}$$
4. Pour tout signataire $i \in \mathcal{L}$:
   (a) $x_i \leftarrow \emptyset$;
   (b) Tant que $x_i = \emptyset$ faire :
      i. $r_i \xleftarrow{R} \{1, \ldots, 2^{mt}\}$;
      ii. $x_i \leftarrow \Delta^{(i)} \left(E_{k,i}^{-1}(P(i)) + \mathcal{H}(M \| r_i) \right)$;
5. Renvoyer $\sigma = (\mathcal{N}, x_1, \ldots, x_n, r_1, \ldots, r_n, P)$ comme signature de cercle.

– TRS.Vérifier($\mathcal{P}, M, \sigma, \mathcal{N}$) où $\sigma = (\mathcal{N}, x_1, \ldots, x_n, r_1, \ldots, r_n, P)$

– Si $P(0) = \mathcal{H}(M \| \mathcal{N})$ et si pour tout membre $i \in \mathcal{N}$:
$$\begin{cases} wt(x_i) \leq t \\ P(i) = E_{\mathcal{H}_\kappa(M),i}(H^{(i)} x_i^T + \mathcal{H}(M \| r_i)), \end{cases}$$
alors renvoyer valide ;

– Sinon, renvoyer invalide.

7.3.2 Sécurité

Examinons maintenant la sécurité du schéma ainsi formé. Comme nous l'avons vu à la section 7.1.2, un schéma de signature de cercle à seuil doit satisfaire deux exigences de sécurité : l'anonymat du groupe des signataires et la résistance aux contrefaçons.

Anonymat

L'anonymat du groupe des signataires peut être aisément établi dans le schéma proposé. Une simulation possible consiste simplement à échanger le rôle de l'utilisateur exclu des signataires avec l'un des membres non-signataires. Que ce soit lors d'une simulation ou lors d'une génération à l'aide de notre algorithme de signature, les valeurs interpolées proviennent directement des fonctions de chiffrement idéales et sont donc uniformément réparties. Or, tout polynôme de degré $N-\ell$ peut être obtenu par interpolation de $N-\ell+1$ valeurs. Il s'ensuit donc qu'une simulation aura la même probabilité de produire une signature donnée σ que notre algorithme de signature.

Résistance aux contrefaçons

Afin de prouver la résistance aux contrefaçons de notre schéma, nous relions la difficulté de produire une contrefaçon existentielle à celle de résoudre les problèmes CGPSD et GD présentés à la section 3.2. Nous établissons le théorème suivant :

Théorème 7.3.1 (Résistance aux contrefaçons) *Soit $\mathcal{A}$ un adversaire contre DV qui renvoie en temps τ une contrefaçon avec probabilité $\epsilon(\tau)$. $\mathcal{A}$ peut effectuer $q_{\mathcal{H}}$ requêtes à un oracle de hachage $\mathcal{H}$, $q_{\mathcal{E}}$ requêtes à un oracle de chiffrement $\mathcal{E}$ et q_Σ requêtes à un oracle de signature Σ. $\mathcal{A}$ peut également obtenir les clefs privées de $\ell-1$ utilisateurs. $\mathcal{A}$ peut alors être utilisé pour résoudre les problèmes CGPSD ou GD et*

$$\epsilon(\tau) \leq q_{\mathcal{E}} q_{\mathcal{H}} (N-t+1) \binom{N}{t} Succ_{n,k}^{CGPBD}(\tau') + Adv_{n,k}^{GD}(\tau')$$

$$+ \frac{q_{\mathcal{E}}}{2^{mt}} \binom{N}{\ell} \left(\frac{q_{\mathcal{E}}}{N-\ell}\right)^{N-\ell}$$

où $n = 2^m$, $k = 2^m - mt$ et $\tau' = Nmt^2 q_\Sigma$.

Preuve – Soit $\epsilon(\tau)$ la probabilité de succès en temps τ de $\mathcal{A}$. Nous montrons qu'il est possible de l'utiliser pour construire un algorithme $\mathcal{D}$ inversant le problème CGPSD. $\mathcal{D}$ reçoit en entrée une matrice binaire H^* de taille $mt \times 2^m$ aléatoire et un vecteur aléatoire $s^* \in \mathbb{F}_2^{mt}$. Son objectif est de trouver x^* tel que $wt(x^*) \leq t$ et $H^*(x^*)^T = s^*$.

$\mathcal{D}$ commence par fixer le cercle $\mathbb{N}$ et le nombre ℓ, il choisit ensuite aléatoirement un utilisateur i_0 parmi les membres du cercle $\mathcal{N}$ et un sous-ensemble $I_0 \subset \mathcal{N}$ de taille $\ell-1$ tel que $i_0 \notin I_0$. $\mathcal{D}$ fait le pari que tous les utilisateurs corrompus par l'adversaire appartiennent à I_0. Il donne

H^* à i_0 en guise de clef publique PK_{i_0} puis il exécute l'algorithme de génération de clefs pour attribuer une paire de clefs publique et privée (PK_i, SK_i) à chaque utilisateur $i \in I_0$.

Pour chacun des autres membres de $\mathcal{N}$, $\mathcal{D}$ remplace leur clef publique par une matrice de parité aléatoire d'un code de Goppa : leurs clefs privées ne seront jamais utilisées. De plus, $\mathcal{D}$ tire aléatoirement deux indices $q_{\mathcal{E},0}$ et $q_{\mathcal{H},0}$ respectivement dans $[1,\ldots,q_{\mathcal{E}}]$ et $[1,\ldots,q_{\mathcal{H}}]$ où $q_{\mathcal{H}}$ est le nombre total de requêtes à l'oracle de hachage $\mathcal{H}$ et $q_{\mathcal{E}}$ le nombre total de requêtes (en chiffrement – notées requêtes E – ou en déchiffrement – notées E^{-1}) à l'oracle de chiffrement $\mathcal{E}$. Enfin, il tire un vecteur $\tilde{s}$ aléatoirement dans $\mathbb{F}_2^{mt}$. $\mathcal{A}$ est alors initialisé avec l'ensemble $\{PK_i\}_{i\in N}$ de clefs publiques.

$\mathcal{D}$ simule l'oracle $\mathcal{H}$ de façon classique en renvoyant une valeur aléatoire dans $\mathbb{F}_2^{mt}$ à chaque nouvelle requête qui lui est soumise ; il maintient également une liste des requêtes afin de simuler le déterminisme des fonctions de hachage. Il simule également l'oracle de chiffrement $\mathcal{E}$ en renvoyant une nouvelle valeur aléatoire pour chaque nouvelle requête tout en conservant la bijectivité des permutations. Cependant, à la ${q_{\mathcal{H},0}}^{eme}$ requête à l'oracle de hachage, il renvoie $\tilde{s}$ et lors de la ${q_{\mathcal{E},0}}^{eme}$ requête à l'oracle de chiffrement (de type E^{-1}), il renvoie $s^* + \tilde{s}$. Remarquons que $\mathcal{D}$ doit maintenir une liste lui permettant de retrouver si x est la réponse à une requête $E^{-1}(y)$ ou si y est la réponse à une requête $E(x)$. Durant la simulation, $\mathcal{A}$ peut corrompre $\ell - 1$ utilisateurs en effectuant une requête sur leur clef privée. $\mathcal{D}$ répond trivialement à celles-ci mais effectue une sortie en échec si $\mathcal{A}$ demande la clef d'un membre de I_0.

$\mathcal{D}$ simule l'oracle de signature Σ pour un sous-ensemble arbitraire de signataires en tirant aléatoirement ses éléments et en adaptant les réponses de l'oracle $\mathcal{E}$. Plus précisément, pour une requête $(M, \mathcal{L})$, $\mathcal{D}$ simule les requêtes $\mathcal{H}_\kappa(M)$ et $\mathcal{H}(M\|\mathcal{N})$ permettant de fixer k et y_0, il tire un polynôme P aléatoire de degré $N - \ell$ tel que $P(0) = y_0$ et N vecteurs aléatoires $x_1, \ldots, x_N$ de $\mathbb{F}_2^n$ dont le poids est au plus t ainsi que N valeurs aléatoires $r_1, \ldots, r_N$. $\mathcal{D}$ simule alors $\mathcal{H}$, fixe les valeurs pour $E_{k,i}(H^{(i)}x_i^T + \mathcal{H}(M\|r_i))$ à $P(i)$ et renvoie $(m, x_1, \ldots, x_n, r_1, \ldots, r_n, P)$ en guise de signature. A la fin de la simulation, $\mathcal{A}$ renvoie avec probabilité $\epsilon(\tau)$ une contrefaçon $\sigma^* = (M^*, x_1, \ldots, x_n, r_1, \ldots, r_n, P^*)$. Par définition, $P^*(0) = \mathcal{H}(M^*\|\mathcal{N})$ et pour tout $i \in \mathcal{N}$, $P^*(i) = E_{\mathcal{H}(M),i}(H^{(i)}x_i^T)$. Si $H^*x_{i_0}^T = s^*$ et $wt(x_{i_0}) \leq t$, alors $\mathcal{D}$ renvoie x_{i_0}.

Bresson *et al.* ont montré dans [BSS02] que $\mathcal{A}$ produit une contrefaçon telle qu'au plus $N - \ell$ requêtes de chiffrement $\mathcal{E}$ mettent en cause une valeur $P^*(i)$ avec probabilité au moins $\epsilon(\tau) + \frac{q_{\mathcal{E}}}{2^{mt}}\binom{N}{\ell}\left(\frac{q_{\mathcal{E}}}{N-\ell}\right)^{N-\ell}$. Ainsi, il y a au moins ℓ indices i pour lesquels $\mathcal{A}$ a effectué une requête de déchiffrement pour $P^*(i)$; notons I^* l'ensemble de ces indices. Puisque $\mathcal{A}$ peut corrompre au plus $\ell - 1$ utilisateurs, il existe au moins un indice i^* pour lequel $\mathcal{A}$ n'a pas corrompu la clef privée de l'utilisateur correspondant. Soit q^* l'indice de la requête faite à $\mathcal{E}$ sur $P^*(i^*)$. Avec probabilité au moins $1/q_{\mathcal{E}}$, $q^* = q_{\mathcal{E},0}$, avec probabilité au moins $1/(n - \ell + 1)$, $i^* = i_0$ et avec probabilité au moins $1/q_{\mathcal{H}}$ la ${q_{\mathcal{H},0}}^{eme}$ requête à l'oracle de hachage est $(M\|r_{i_0})$. Dans ce cas, x_{i_0} est de poids inférieur à t et vérifie $Hx_{i_0}^T = E^{-1}_{\mathcal{H}(M),i}(P(i_0)) + \mathcal{H}(M\|r_{i_0}) = s^* + \tilde{s} + \tilde{s} = s^*$. Ainsi,

x_{i_0} est un décodage à distance t de s^*.

Le temps d'exécution τ' de $\mathcal{D}$ est essentiellement le temps d'exécution τ de $\mathcal{A}$ auquel il faut ajouter le coût du calcul de Nq_Σ syndromes, chacun d'eux nécessitant mt^2 opérations binaires. Remplacer la clef publique de l'utilisateur i_0 par une matrice aléatoire n'affecte pas la probabilité de succès de la simulation au delà de l'avantage $Adv^{GD}_{2^m,2^m-mt}(\tau')$ du meilleur algorithme permettant de résoudre le problème CGPSD : dans le cas contraire, $\mathcal{D}$ fournirait un meilleur distingueur. Notons $\epsilon_{GPBD}(\tau') = Succ^{GPBD}_{2^m,2^m-mt}(\tau')$ et $\epsilon_{GD}(\tau') = Adv^{GD}_{2^m,2^m-mt}(\tau')$; on obtient alors :

$$\left(\epsilon(\tau) - \frac{q_\mathcal{E}}{2^{mt}}\binom{N}{\ell}\left(\frac{q_\mathcal{E}}{N-\ell}\right)^{N-\ell} + \epsilon_{GD}(\tau')\right)\frac{1}{q_\mathcal{E}q_\mathcal{H}(N-t+1)\binom{N}{t}} \leq \epsilon_{GPBD}(\tau'),$$

ce qui conclue la preuve. ♢

La réduction proposée n'est clairement pas fine puisque la probabilité de résoudre le problème CGPSD est multipliée par un facteur $q_\mathcal{E}q_\mathcal{H}(N-t+1)\binom{N}{t}$. De plus, minimiser la probabilité $\epsilon(\tau)$ implique de rendre négligeable la quantité $\frac{q_\mathcal{E}}{2^{mt}}\binom{N}{\ell}\left(\frac{q_\mathcal{E}}{N-\ell}\right)^{N-\ell}$. Ceci nécessite soit d'utiliser des codes de grande taille, soit une grande taille de cercle. La sécurité offerte par notre preuve est donc uniquement asymptotique.

7.3.3 Performances et comparaisons

La preuve de sécurité précédente nous indique donc que les signatures formées à l'aide de notre schéma seront résistantes aux contrefaçons si les problèmes CGPSD et GD sont tous deux difficiles. Comme cela est fait dans [CFS01], nous faisons l'hypothèse que la difficulté de GD est toujours plus grande que celle de CGPSD. De plus, il est impératif de maintenir raisonnable le nombre de décodages à effectuer par chacun des signataires, c'est-à-dire la capacité de correction t du code utilisé.

Les bornes fournies par M. Finiasz et N. Sendrier dans [FS09] présentées section 3.2.3 nous permettent de proposer les paramètres $m = 16$ et $t = 9$ pour une sécurité de 2^{63} et $m = 22$ et $t = 9$ pour obtenir une sécurité de 2^{81}.

Performances

Pour un jeu de paramètres (m, t),

- chacune des clefs publiques est une matrice de parité d'un code de Goppa de longueur 2^m et de dimension $2^m - mt$ nécessitant le stockage de $mt2^m$ bits ;
- la génération d'une signature nécessite le calcul de $N - \ell$ syndromes (exigeant chacun mt^2 opérations binaires), N évaluations de polynôme et $\ell(t!)$ décodages (coûtant chacun t^2m^3 opérations binaires) ;

– une signature est composée de N vecteurs de poids t de $\mathbb{F}_2^{2^m}$, de N aléas de $\{0, \ldots, 2^m - 1\}$ et d'un polynôme de $\mathbb{F}_{2^{mt}}$ de degré $N - \ell$: elle nécessite donc

$$\left(\left\lfloor \log_2 \sum_{j=1}^{t} \binom{2^m}{j} \right\rfloor + 2mt + 1 \right) \times N - mt\ell \text{ bits} ;$$

– sa vérification nécessite N calculs de syndromes et $N + 1$ évaluations de polynôme.

Comparaisons

Les signatures produites par notre schéma utilisé avec les paramètre $m = 16$ et $t = 9$ (fournissant ainsi le même niveau de sécurité que le schéma de D. Zheng, X. Li et K. Chen [ZLC07]) produit des signatures de $414N - 144\ell$ bits contre $144 + 126N$ bits pour le schéma de D. Zheng *et al.*, pour un coût de génération équivalent. L'ajout d'un seuil et les structures différentes des deux schémas expliquent en partie cette différence ; mais elle est également due à une différence dans la sélection des indices associés à un mot de petit poids. Alors que notre schéma tire des indices de valeurs aléatoires, le schéma de D. Zheng *et al.* utilise des indices successifs, permettant ainsi de les maintenir petits ($\log_2(t!)$ bits en moyenne). Or, nous avons vu à la section 6.3 qu'un tel procédé ne permet pas de prouver la sécurité du schéma.

Avec les paramètres $m = 22$ et $t = 9$ (fournissant un niveau de sécurité équivalent à celui du schéma de C. Aguillar, P.-L. Cayrel et P. Gaborit [ACG08]) la matrice publique nécessite environ 99 Mo pour être stockée mais produit des signatures de $579N - 198\ell$ bits, d'autant plus courtes que le nombre de signataires augmente. Pour un même niveau de sécurité, les signatures produites par le schéma de C. Aguillar *et al.* ont une taille constante d'environ $20N$ Ko et utilisent des clefs de 347 bits. Ce schéma et le notre souffrent donc des qualités et des défauts du schéma de signature standard sur lesquels ils sont basés et n'entrent donc pas réellement en concurrence. Ils permettent de disposer de deux alternatives selon les contraintes imposées par l'environnement dans lequel ils pourraient être déployées.

Conclusion

Le sujet de ce manuscrit est l'étude de protocoles cryptographiques fondés sur la théorie des codes correcteurs d'erreurs pour le chiffrement et la signature. Deux points de vue y sont adoptés : celui du concepteur de protocole et celui du cryptanalyste.

Au cours des trente-cinq dernières années, les concepteurs de protocoles cryptographiques se sont dotés d'un outil puissant : la sécurité réductionniste. Celle-ci permet, sinon de se prémunir de toute attaque, tout au moins d'identifier et de quantifier les éventuelles faiblesses d'un schéma, relativement à la difficulté d'un problème mathématique. Nous avons donc présenté les principaux schémas de chiffrement fondés sur les codes correcteurs d'erreurs en présentant une analyse rigoureuse de leur sécurité située dans ce cadre. Celle-ci nous a permis de mettre en évidence les problèmes auxquels sont reliés le cryptosystème de McEliece [McE78], celui de Niederreiter [Nie86] et le schéma hybride de Biswas & Sendrier [BS08] ; à savoir le problème du décodage borné restreint, que nous introduisons dans ce mémoire, le problème équivalent du décodage par syndrome restreint et le problème de la distinguabilité des codes de Goppa (problème GD).

Nous avons ensuite proposé les cryptanalyses structurelles de deux variantes du schéma de McEliece visant à réduire la taille des clefs. La première attaque s'applique à la variante présentée dans [Gab05], qui utilise des codes quasi-cycliques construits à partir d'une matrice publique. La seconde attaque s'applique à une variante utilisant des codes quasi-cycliques LDPC proposée dans [BC07]. Ces deux attaques parviennent au bris total des cryptosystèmes en déduisant la structure de la clef privée à partir de la clef publique. Elles ne remettent donc aucunement en cause la difficulté du décodage dans des codes aléatoires de même caractéristique que les codes utilisés, mais bien la capacité des transformations utilisées à masquer un code très structuré.

Une étude de la sécurité du schéma de signature CFS proposé en 2001 par N. Courtois, M. Finiasz et N. Sendrier [CFS01] dans un cadre réductionniste nous a permis de mettre en évidence la nécessité de modifier légèrement le schéma pour en exhiber une preuve. Une fois cette modification réalisée, nous avons montré que la sécurité du schéma est elle aussi reliée à la difficulté du problème du décodage par syndrome et à celle du problème de la distinguabilité des codes de Goppa. Ce travail nous a par la suite permis de proposer un nouveau schéma de signature de cercle à seuil, dont la sécurité est également reliée à ces problèmes. Cette construc-

tion offre une alternative au seul autre schéma de signature de cercle à seuil fondé sur la théorie des codes correcteurs d'erreurs [ACG08] en permettant d'obtenir des signatures de petite taille mais nécessitant des clefs plus conséquentes.

Dans ce manuscrit, nous nous sommes volontairement peu attardés sur les algorithmes de décodages pour nous concentrer sur une approche plus protocolaire. Ce choix, certes contestable, permet de mettre en avant le caractère générique des solutions apportées, en particulier pour pallier la non surjectivité des fonctions à sens unique utilisées.

Il ressort des travaux précédents deux constats : l'extrême nécessité de masquer la structure du code public utilisé et la difficulté de dériver de nouveaux schémas d'une fonction à sens unique fondée sur les codes correcteurs d'erreurs.

La nécessité de masquer la structure du code est avant tout illustrée par les cryptanalyses que nous avons réalisées. Elles démontrent en effet clairement que l'utilisation d'un code public trop structuré peut permettre à un attaquant d'en déduire la clef privée. Dans le cas où les codes de Goppa sont utilisés, cette nécessité est démontrée par l'omniprésence du problème de la distinguabilité des codes de Goppa dans les réductions de sécurité. A ce titre, les preuves de sécurité que nous avons exhibées doivent être mises en perspectives avec les récents résultats de J.C. Faugère, A. Otmani, L. Perret et J.P. Tillich [FOPT10b] dans lesquels ils résolvent le problème de la distinguabilité des codes de Goppa pour des instances de grand rendement, ce qui est le cas des codes utilisés dans les schémas de signature. Si l'existence d'un distingueur efficace pour les codes de Goppa ne remet pas en cause la validité des preuves précédentes, elle en diminue cependant considérablement la portée pratique : la preuve nous dit alors que casser le schéma est plus difficile qu'un problème que l'on peut désormais considérer comme facile. De plus, ce distingueur ne permet pas, à l'heure actuelle, d'en dériver une attaque pratique sur les schémas utilisant l'hypothèse de la difficulté du problème de la distinguabilité des codes de Goppa. Deux approches sont donc possibles pour obtenir un schéma de signature fondé sur les codes correcteurs d'erreurs pour lequel on dispose d'une preuve pertinente de sa sécurité : trouver une preuve du schéma CFS ne faisant jamais intervenir l'hypothèse de la difficulté de GD ou décrire un nouveau schéma sans utiliser cette hypothèse.

La difficulté de dériver un schéma de signature d'une fonction à sens unique fondée sur les codes a été soulignée par N. Courtois, M. Finiasz et N. Sendrier lorsqu'ils ont proposé le schéma de signature CFS [CFS01]. En effet, aucun schéma de chiffrement fondé sur les codes correcteurs d'erreurs n'est injectif, alors que de nombreuses constructions nécessitent une fonction à sens unique qui doit l'être. La construction proposée par N. Courtois, M. Finiasz et N. Sendrier, également au cœur de notre schéma de signature de cercle à seuil, nécessite l'inversion de nombreuses valeurs aléatoires. Ceci conduit à utiliser des codes de faible capacité de décodage (et donc de grande taille) pour garder un coût de signature raisonnable, ce qui rend difficile le choix des paramètres et peut apporter d'éventuelles faiblesses, comme nous l'avons vu

au paragraphe précédent.

L'utilisation des fonctions à sens unique fondées sur les codes correcteurs d'erreurs se heurte également à un autre obstacle : l'impossibilité de combiner deux instances des problèmes de décodage borné. En effet, la somme de deux mots de code en erreur est également un mot de code en erreur, mais le poids de celle-ci augmente pour atteindre au plus la somme des poids des deux erreurs. Ceci, ne permettant pas de s'assurer que le décodage de la somme sera la somme de leurs décodages. Le caractère discret des erreurs rend le contrôle de l'expansion de leur poids difficile à maîtriser. Si cette propriété peut être une force en rendant les schémas cryptographiques non malléables, elle présente également l'inconvénient de rendre inapplicable un certain nombre de techniques utilisées pour dériver de nouveaux schémas des cryptosystèmes fondés sur l'arithmétique modulaire. En particulier la technique utilisée par J.S. Coron dans [Cor00] pour obtenir une preuve plus fine de la sécurité de RSA-FDH [BR93] ne peut être appliquée au schéma CFS.

Une solution à cette dernière difficulté peut être apportée par les réseaux euclidiens, des objets mathématiques proches des codes. Un réseau euclidien est l'ensemble des combinaisons linéaires entières d'un ensemble de vecteurs de $\mathbb{R}^n$, lequel définit une base du réseau. Il existe un certain nombre de problèmes considérés comme difficiles et proches de ceux des codes : le problème du plus court vecteur (SVP), que l'on peut rapprocher de la recherche de la distance minimale d'un code, et le problème du vecteur le plus proche (CVP), que l'on peut rapprocher des problèmes de décodage. Ce dernier est particulièrement intéressant de notre point de vue : si ce problème est difficile, il est alors délicat de retrouver un message chiffré par un vecteur du réseau auquel on aurait appliqué une légère perturbation (un vecteur de $\mathbb{R}^n$ de petite norme), construction qui est l'équivalent du chiffrement de McEliece dans les réseaux. Le caractère continu des perturbations permet alors un meilleur contrôle de l'expansion que ce qui est possible avec des codes correcteurs d'erreurs. Cette idée a été utilisée avec succès dans [MCG08] et [MG08], par exemple. De plus, les problèmes sur les réseaux euclidiens considérés comme difficiles ont fait l'objet d'une recherche intensive depuis l'apparition de l'algorithme LLL en 1982 [LLL82], ce qui permet d'avoir aujourd'hui une relativement bonne connaissance de ces problèmes. Une étude comparée de ces deux familles de cryptosystème pourrait donc permettre de mieux appréhender la sécurité des schémas fondés sur la théorie des codes, ainsi que l'émergence de nouvelles constructions cryptographiques.

Bibliographie

[ACG08] C. Aguilar Melchor, P. L. Cayrel, and P. Gaborit. A new efficient threshold ring signature scheme based on coding theory. In *Post-Quantum Cryptography*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46. Springer-Verlag, Berlin, 2008.

[BC07] M. Baldi and G. F. Chiaraluce. Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In *IEEE International Symposium on Information Theory*, pages 2591–2595, 2007.

[BCGO09] T. Berger, P-L. Cayrel, P. Gaborit, and A. Otmani. Reducing key length of the McEliece cryptosystem. In *Progress in Cryptology – Second International Conference on Cryptology in Africa (AFRICACRYPT 2009)*, volume 5580 of *Lecture Notes in Computer Science*, pages 77–97, 2009.

[BCP97] W. Bosma, J. J. Cannon, and C. Playoust. The MAGMA algebra system I : The user language. *Journal of Symbolic Computation*, 24(3/4) :235–265, 1997. Special Issue on Computational Algebra and Number Theory : Proceedings of the First MAGMA Conference.

[BDPR98] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryptions schemes. In *Advances in Cryptology — CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45, 1998.

[BLP08] J Bernstein, T. Lange, and C. Peters. Attacking and defending the McEliece cryptosystem. In *Post-Quantum Cryptography*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46. Springer Berlin, 2008.

[BMv78] E. Berlekamp, R. J. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3) :384–386, 1978.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical : A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR95] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Proc. of Eurocrypt'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1995.

[BRC60] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error-correcting binary group codes. *Information and Control*, 3(1) :68–79, 1960.

[BS08] B. Biswas and N. Sendrier. McEliece cryptosystem in real life : theory and practice. In J. Buchmann and J. Ding, editors, *PQCrypto*, number 5299 in Lecture Notes in Computer Science, pages 47–62. Springer-Verlag, 2008.

[BSS02] E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *Proc. of Crypto'02*, volume 2442 of *Lecture Notes in Computer Science*, pages 465–480. Springer-Verlag, Berlin, 2002.

[CC94] A. Canteaut and H. Chabanne. A further improvement of the work factor in an attempt at breaking McEliece's cryptosystem. In P. Charpin, editor, *EUROCODE 94*, 1994. http://www.inria.fr/rrrt/rr-2227.html.

[CC98] A. Canteaut and F. Chabaud. 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*, 44(1) :367–378, 1998.

[CDMP05] J. S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgå rd revisited : How to construct a hash function. In *Advances in Cryptology — CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448, 2005.

[CFS01] N. Courtois, M. Finiasz, and N. Sendrier. How to achieve a McEliece-based digital signature scheme. In *Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174. Springer, 2001.

[CGH04] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4) :557–594, 2004.

[Cor00] J.S. Coron. On the exact security of full domain hash. In *Advances in Cryptology — CRYPTO'00*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–236. Springer-Verlag, 2000.

[Cov73] T. Cover. Enumerative source encoding. *IEEE Transactions on Information Theory*, 19(1) :73–77, 1973.

[COV07] P-L. Cayrel, A. Otmani, and D. Vergnaud. On Kabatianskii-Krouk-Smeets signatures. In C. Carlet and B. Sunar, editors, *International Workshop on the Arithmetic of Finite Fields (WAIFI 2007)*, volume 4547 of *Lecture Notes in Computer Science*, pages 237–251. Springer, 2007.

[CPS08] J. S. Coron, J. Patarin, and Y. Seurin. The random oracle model and the ideal cipher model are equivalent. In *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 1–20, 2008.

[CS98] A. Canteaut and N. Sendrier. Cryptanalysis of the original McEliece cryptosystem. In *Advances in Cryptology – ASIACRYPT'98*, volume 1514 of *Lecture Notes in Computer Science*, pages 187–199. Springer Berlin, 1998.

[Cv92] D. Chaum and E. van Heyst. Group signatures. In *Proc. of Eurocrypt'01*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer-Verlag, Berlin, 1992.

[Dal08] L. Dallot. Towards a concrete security proof of Courtois, Finiasz and Sendrier signature scheme. In S. Lucks, A-R. Sadeghi, and C. Wolf, editors, *Research in Cryptology – Second Western European Workshop, WEWoRC 2007, Revised Selected Papers*, volume 4945 of *Lecture Notes in Computer Science*, pages 65–77, 2008.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22 :644–654, 1976.

[DKNS04] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad-hoc groups. In *Advances in Cryptology – Eurocrypt'04*, volume 3027 of *Lecture Notes in Computer Science*, pages 609–626. Springer, 2004.

[DMQN09] R. Dowsley, J. Müller-Quade, and A. Nascimento. A CCA2 secure public key encryption scheme based on the McEliece assumptions in the standard model. In *Topics in Cryptology – CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 240–251. Springer Berlin / Heidelberg, 2009.

[DR02] J. Daemen and V Rijmen. *The Design of Rijndael : AES – The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[DV09] L. Dallot and D. Vergnaud. Provably secure code-based threshold ring signatures. In Matthew G. Parker, editor, *12th IMA international conference, Cryptography and Coding 2009*, volume 5921 of *Lecture Notes in Computer Science*, pages 222–235, Cirencester, UK, 2009.

[Fin04] M. Finiasz. *Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie à clef publique*. PhD thesis, INRIA – Ecole Polytechnique, October 2004. http://www-rocq.inria.fr/codes/Matthieu.Finiasz/research/2004/finiasz-these.pdf.

[FO99a] E. Fujisaki and T. Okatomo. How to enhance the security of public-key encryption at minimum cost. In *Proc. of PKC'99*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68. Springer-Verlag, 1999.

[FO99b] E. Fujisaki and T. Okatomo. Secure integration of asymmetric and symmetric encryption schemes. In *Proc. of Crypto'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 535–554. Springer-Verlag, 1999.

[FOPT10a] J-C. Faugère, A. Otmani, L. Perret, and J-P. Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. In *Proceedings of the 29th International Conference on Cryptology — EUROCRYPT 2010*, 2010.

[FOPT10b] J.C. Faugère, A. Otmani, L. Perret, and J.P. Tillich. A distinguisher for high rate McEliece cryptosystem. Communication privée, Avril 2010.

[FS87] A. Fiat and A. Shamir. How to prove yourself : Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO'86*, volume 236 of *Lecture Notes in Computer Science*, pages 186–194. Springer Berlin, 1987.

[FS09] M. Finiasz and N. Sendrier. Security bounds for the design of code-based cryptosystems. In *Advances in cryptology – Asiacrypt 2009*, number 5912 in Lecture Notes in Computer Science, pages 88–105. Springer, 2009.

[Gab05] P. Gaborit. Shorter keys for code based cryptography. In *Proceedings of the 2005 Workshop on Coding and Cryptography (WCC 2005)*, pages 81–90, Bergen, Norway, 2005.

[Gal63] R. G. Gallager. *Low Density Parity Check Codes*. MIT Press, Cambrige, MA, 1963.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and Systems Sciences*, 28(2) :270–299, 1984.

[GMR85] S. Goldwasser, S. Micali, and R. L. Rivest. A paradoxical solution to the signature problem (abstract). In G. R. Blakley and D. C. Chaum, editors, *Advances in Cryptology – CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, page 467, 1985.

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2) :281–308, 1988.

[Gol66] S. Golomb. Run-length encoding. *IEEE Transactions on Information Theory*, 12(3) :399–401, 1966.

[Gop70] V. D. Goppa. A new class of linear correcting codes. *Probl. Peredachi Inf.*, 6(3) :24–30, 1970.

[GZ61] D. Gorenstein and N. Zierler. A class of error correcting codes in p^m symbols. *Journal of the Society for Industrial and Applied Mathematics*, 9(2) :207–214, 1961.

[Hoc59] A. Hocquenghem. Codes correcteurs d'erreurs. *Chiffres, Revue assoc. franc. calcul*, 2 :147–156, 1959.

[JH04] J. Justesen and T. Høholdt. *A course in error-correcting codes*. Textbooks in Mathematics. European Mathematical Society, Jan. 2004.

[JSI96] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Advances in Cryptology — Eurocrypt'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer-Verlag, 1996.

[KI01] K. Kobara and H. Imai. Semantically secure McEliece public-key cryptosystems — conversions for McEliece PKC. In *PKC'01 : Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, pages 19–35. Springer-Verlag, 2001.

[KKS97] G. Kabatianskii, E. Krouk, and B. J. M. Smeets. A digital signature scheme based on random error-correcting codes. In *IMA International Conference*, volume 1355 of *Lecture Notes in Computer Science*, pages 161–167. Springer, 1997.

[LB88] P. J. Lee and E. F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In *Advances in Cryptology – EUROCRYPT'88*, volume 330 of *Lecture Notes in Computer Science*, pages 275–280. Springer Berlin, 1988.

[LDW94] Y. X. Li, Robert H. Deng, and X. M. Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, 40(1) :271–273, 1994.

[Leo88] J. S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5) :1354–1359, 1988.

[LLL82] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematishe Annalen*, 261 :515–534, 1982.

[Mas69] J. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15(1) :122–127, 1969.

[MB09] R. Misoczki and P. S. L. Barreto. Compact McEliece keys from Goppa codes. In *Selected Areas in Cryptography (SAC 2009)*, 2009.

[McE78] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. Technical report, DSN Progress report # 42–44, Jet Propulsion Laboratory, Pasadena, Californila, 1978.

[MCG08] C.A. Melchor, G. Castagnos, and P. Gaborit. Lattice-based homomorphic encryption of vector spaces. In *The 2008 IEEE International Symposium on Information Theory (ISIT'08)*, pages 1858–1862, 2008.

[MG08] C.A. Melchor and P. Gaborit. A fast private information retrieval protocol. In

The 2008 IEEE International Symposium on Information Theory (ISIT'08), pages 1848–1852, 2008.

[Mic10] D. Micciancio. Cryptographic functions from worst-case complexity assumptions. In P. Q. Nguyen and B. Vallée, editors, *The LLL algorithm*, Information Security and Cryptography, pages 427–452. Springer, 2010.

[MN95] D. J. C. MacKay and R. M. Neal. Good codes based on very sparse matrices. In *5th IMA Conference on Cryptography and Coding*, volume 1025 of *Lecture Notes in Computer Science*, pages 100–111. Springer-Verlag, 1995.

[Moo05] T. K. Moon. *Error Correction Coding – Mathematical Methods and Algorithms*. Wiler-Interscience, 2005.

[MRS00] C. Monico, J. Rosenthal, and A. Shokrollahi. Using low density parity check codes in the McEliece cryptosystem. In *IEEE International Symposium on Information Theory (ISIT 2000)*, page 215, Sorrento, Italy, 2000.

[MS77] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[MS07] L. Minder and A. Shokrollahi. Cryptanalysis of the Sidelnikov cryptosystem. In *Advances in Cryptology — Eurocrypt'2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 347–360. Springer-Verlag, 2007.

[Nao02] M. Naor. Deniable ring authentification. In *Advances in Cryptology – Crypo 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 481–498. Springer, 2002.

[Nie86] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15 :159—-166, 1986.

[NIKM08] R. Nojima, H. Imai, K. Kobara, and K. Morozov. Semantic security for the mceliece cryptosystem without random oracles. *Design, Codes and Cryptography*, 49(1–3) :289–305, 2008.

[OTD08a] A. Otmani, J-P. Tillich, and L. Dallot. Cryptanalysis of a McEliece cryptosystem based on quasi-cyclic LDPC codes. In *Symbolic Computation and Cryptography (SCC 2008)*, Beijing, China, April 2008.

[OTD08b] A. Otmani, J-P. Tillich, and L. Dallot. Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. *Mathematics in Computer Science*, Special Issue on Symbolic Computation and Cryptography, 2008. A paraître.

[Pat75] N. Patterson. The algebraic decoding of goppa codes. *IEEE Transactions on Information Theory*, 21(2) :203–207, 1975.

[Pet60] W. Peterson. Encoding and error-correction decoding procedures for the Bose-Chaudhuri codes. *IEEE Transactions on Information Theory*, 6 :459–470, 1960.

[Poi00] D. Pointcheval. Chosen-Cyphertext security for any one-way cryptosystem. In *PKC 2000*, volume 1751 of *Lecture Notes in Computer Science*, pages 129–146. Springer-Verlag, 2000.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of ACM*, 21 :120–126, 1978.

[RST01] R. L. Rivest, A. Shamir, and Y Tauman. How to leak a secret. In *Proceedings of Asiacrypt'01*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer-Verlag, Berlin, 2001.

[Sen98] N Sendrier. On the concatenated structure of a linear code. *Applicable Algebra in Engineering, Communication and Computing*, 9(3) :221–242, 1998.

[Sen02] N. Sendrier. *Cryptosystèmes à clé publique basés sur les codes correcteurs d'erreurs.* Habilitation à diriger les recherches, Université Pierre et Marie Curie, Paris 6, Paris, France, Mars 2002.

[Sha79] A. Shamir. How to share a secret. *Commun. of the ACM*, 22(11) :612–613, 1979.

[Sha84] A. Shamir. A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem. *IEEE Transactions on Information Theory*, 30(5) :699–704, 1984.

[Sho04] V. Shoup. Sequences of games : a tool for taming complexity in security proofs. manuscript, November 2004. Revised, May 2005 ; Jan. 2006.

[Sid94] V. M. Sidelnikov. A public-key cryptosystem based on binary reed-muller codes. *Discrete mathematics and applications*, 4(3), 1994.

[Sin01] S. Singh. *Histoire des codes secrets.* Le Livre de Poche, 2001. Traduction de C. Coqueret.

[SS92] V. M. Sidelnikov and S. O. Shestakov. On insecurity of cryptosystems based on generalized Reed-Solomon codes. *Discrete Mathematics and Applications*, 2 :439–444, 1992.

[Ste89] J. Stern. A method for finding codewords of small weight. In *Coding Theory and Applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer Berlin, 1989.

[Ste90] J. Stern. An alternative to the Fiat-Shamir protocol. In *Advances in Cryoptology – Eurocrypt'89*, volume 434 of *Lecture Notes in Computer Science*, pages 173–180. Springer-Verlag, 1990.

[Ste94] J. Stern. A new identification scheme based on syndrome decoding. In *Advances in Cryoptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 13–21. Springer Berlin, 1994.

[Ste96] J. Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory*, 42(6) :1757–1768, 1996.

[Tur37] A. Turing. On computable numbers, with an application to the Entscheidung's problem. In *Proceedings of the London Mathematical Society*, volume 42, pages 230–365, 1937.

[Vau98] S. Vaudenay. Cryptanalysis of the Chor-Rivest cryptosystem. In Springer, editor, *Advances in Cryptology – CRYPTO 98*, volume 1462 of *Lecture Notes in Computer Science*, pages 243–256, 1998.

[Wag02] D. Wagner. A generalized birthday problem. In *CRYPTO '02 : Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, Lecture Notes in Computer Science, pages 288–303. Springer-Verlag, 2002.

[Wel88] D. Welsh. *Codes and Cryptography*. Oxford University Press, 1988.

[ZLC07] D. Zheng, X. Li, and K Chen. Code-based ring signature scheme. *International Journal of Network Security*, 5(2) :154–157, 2007.