UNIVERSITÉ DE LIMOGES
ECOLE DOCTORALE Science - Technologie - Santé
Faculté des Sciences et Techniques

Département de Mathématiques et Informatique – Laboratoire XLIM

Thèse N˚ 39 - 2008

Thèse
pour obtenir le grade de
DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline : Mathématiques et ses applications

soutenue et présentée
par

Pierre-Louis CAYREL
le 2 octobre 2008, à 14h00

Construction et optimisation de cryptosystèmes basés sur les codes correcteurs d'erreurs

Jury

Président		
	Thierry BERGER	Professeur à l'Université de Limoges
Directeur		
	Philippe GABORIT	Professeur à l'Université de Limoges
Rapporteurs		
	Marc GIRAULT	Expert émérite, France Telecom Caen HDR
	Nicolas SENDRIER	Directeur de Recherches INRIA Rocquencourt
Examinateurs		
	Jean-Claude BAJARD	Professeur à l'Université de Montpellier
	Yassine LAKHNECH	Professeur à l'Université de Grenoble
	Jean-Louis LANET	Professeur à l'Université de Limoges

Remerciements

Mes remerciements vont en premier lieu à Philippe Gaborit pour m'avoir confié ce sujet de recherche et m'avoir soutenu par ses conseils et ses idées durant ces trois années.

Ensuite, je tiens à exprimer mes remerciements aux membres du jury à savoir, Jean-Claude Bajard, Thierry Berger, Yassine Lakhnech et Jean-Louis Lanet qui ont accepté d'évaluer mon travail.

Remerciements, également avec grand plaisir, à Nicolas Sendrier et Marc Girault qui ont accepté de rapporter sur cette thèse.

Certaines personnes ne peuvent être oubliées à savoir ceux qui sont passés au moins un jour durant ces trois années de thèse par le *bureau vert*.

Initialement squatté par Guilhem Castagnos (en espérant qu'il se remette des défaites toulousaines), puis par ordre chronologique Elsa Bousquet, Ahmed Ait Mokhtar, Maboubeh Rezai, Mohsen Asghari Larimi, Ali Mouhib, Sinaly Traoré et plus récemment Benjamin Pousse (monsieur le président) et Ainhoa Aparicio Monforte qui partagent actuellement ce bureau avec moi depuis maintenant une année.

Au delà de ces gens là, je tiens à adresser mes remerciements aux thésards qui sont passés par Limoges durant cette thèse à savoir Samuel Maffre, Nicolas Le Roux (alias Mister the russet), Laurent Dubreuil, Marius Durea, Hassan Saoud, Christophe Chabot (alias Totof), Julien Angeli (et sa bouteille de manzana), Aurore Bernard, Sandrine Jean, Romain Validire, Adrien Poteaux et Daouda Diatta.

Mais, je n'oublie pas les secrétaires du bâtiment de mathématiques et de l'IUT : Yolande Viecieli (pour sa gentillesse et sa disponibilité de tous les instants), Aurélie Doucet, Patricia Vareille, Sylvie Laval et Marie-Paule Fleury.

Ensuite, je n'oublie pas Sabine Rouanet, Henri Massias, les PICSIs : Olivier Rossi, Constantin Yamkoudougou, ainsi que Mikaël Lescop, Matthias Meulien et Sergi Simon.

S'agissant des permanents du bâtiment de mathématiques (et des bars limougeauds pour certains), je remercie Stéphane Vinatier, Thomas Cluzeau, François Arnault, Pierre Dusart, Abdelkader Necer, Jacques-Arthur Weil, Marc Rybowicz, et tant d'autres.

Pensée particulière à mes co-auteurs (pas encore cités) sans lesquels cette thèse n'aurait pas vu le jour : Frederik Armknecht, Olivier Ruatta, Carlos Aguilar Melchor, David Galindo, Emmanuel Prouff, Ayoub Otmani, Damien Vergnaud, Léonard Dallot, Nadia El Mrabet, Morgan Barbier.

A Montpellier, j'adresse des remerciements particuliers à Arnaud Tisserand (alias Nono), Laurent Imbert (alias Lolo), Valérie Berthé, Pascal Giorgi et Nicolas Méloni (alias Sam Smith).

Remerciements aux membres du projet SECRET de l'INRIA de m'avoir accepté durant leur séjour niçois : Maria Naya Plasencia, Andrea Roeck, Yann Laigle-Chapuy, Frederic Didier, Thomas Camara, Cédric Lauradoux (Scarab).

Et aussi Anne Canteaut, Daniel Augot, Matthieu Finiasz, Rafael Overbeck et Mathieu Cluzeau.

N'oubliant pas Frederic Edoukou et François Rodier grâce auxquels j'ai eu la chance d'être accueilli deux fois à Marseille pour faire un exposé.

Toulon et ces occupants auxquels j'adresse mes remerciements : Pascal Véron pour son soutien et les discussions sur les codes en cryptographie ainsi qu'à Yves Aubry pour son accueil.

Survolons la France vers le nord avec les caennais qui m'ont accueilli dans leur équipe à savoir : Fabien Laguillaumie, Cécile Delerablée, Iwen Coisel.

Pour les parisiens que je rejoins en cette rentrée 2008, un grand merci : Sihem Mesnager pour son accueil et son soutien, Claude Carlet pour sa gentillesse, son soutien et sa disponibilité, n'oubliant pas Philippe Guillot, Rafael Fourquet et Farid Mokhrane.

Orientons nous un peu plus à l'ouest de la France pour saluer l'équipe de Rennes avec en particulier Johann Barbier pour son accueil.

Une halte dans la ville rose pour saluer Alain Couvreur pour son accueil (il doit être en train de rédiger un article ...).

Restons pas très loin de là pour remercier Jean Creignou pour son accueil à Bordeaux et son concert de guitare.

Le sud-ouest est aussi à remercier en la personne de Christophe Negre pour son soutien lors de ma candidature d'ATER à Perpignan et le kite-surfeur Philippe Langlois.

Evoluons maintenant au delà de nos frontières et surtout changeons de climat, pour remercier chaleureusement (ils en ont besoin) le Claude Shannon Institute de Dublin en particulier Gary McGuire pour son accueil, Eymar Birne, Richard Moloney, Cathy McFadden, John Sheekey, Vitaly Skachek, Laura Hitt, Nadya Markin, Carl Bracken et Konstantinos Drakakis.

Une fois dans le grand froid irlandais, j'ai eu la chance de faire la rencontre de Brian Baldwin et Fiona que je tiens à remercier pour leur gentillesse et leur accueil à Cork.

Reste à remercier les personnes que j'ai rencontré durant mon stage de master recherche où j'ai eu la chance d'être accueilli au ministère de la défense où, au delà de la recherche j'ai pu rencontrer : Mon maître (alias Julien Playe), Thomas Roigt et sa culture des pratiques asiatiques hors pair, Patrice Kerdudo et son goût pour la bière, Damien Delangle ainsi que Antoine Coutant (et son chien Vlad) et Yannou.

Sans mes relecteurs *indépendants*, cette thèse n'aurait certainement jamais été finalisée, un grand merci à Théophane Lumineau, Samuel Besse ainsi que Julie.

Organizarea de dracu faite par Nicolae Constantinescu en Roumanie pour le colloque franco-roumain, multumesc !

Une pensée à : Prokop Hybler, Jessica Franco, Bertrand Zanzotto pour l'implémentation en C du schéma de Stern (lors de son stage de master pro), Sophie Couteaud et Magalie Maffre pour leurs travaux effectués sur les codes quasi-cycliques sur des anneaux de matrices, Madeleine Boxberger et les canailles (Rodrigue et Gabriel), Nicolas Nardeau, monsieur Langlet (mon professeur de terminale qui m'a confirmé mon goût pour les mathématiques), Marc Arcostanzo, Dinh The Luc ainsi que Bernard Deflesselles pour soutien permanent.

Toujours dans la même lignée, un grand merci aux ADDMULiens pour m'avoir élu deux fois d'affilée (c'est de l'inconscience) !

Inévitablement mes amis m'ont soutenu : Tomtom, Martinou et Royto, entre autres, un grand merci à eux.

Enfin je terminerai ces remerciements (à rallonge) par mes parents, ceux de Marine (pour leur périple jusqu'à la soutenance), mon frère (pour certaines images), ma grand mère qui m'a hébergé, mon oncle, ma tante, mon cousin (ainsi que ses copains) et ma cousine qui m'ont permis de retrouver de la famille à Limoges.

Ne pas la remercier serait un oubli de taille, un grand merci à Marine pour son soutien de tous les instants ...

Table des matières

Table des figures

Introduction

La notion de cryptographie est née à partir du moment où l'on a voulu communiquer à l'abri d'oreilles indiscrètes. De Jules César et son armée, à Roméo et Juliette, ou encore les joueurs de la partie de cartes de Pagnol, en passant par toutes les cartes au trésor, il a fallu chiffrer certains renseignements. La période contemporaine n'a rien amélioré dans ce domaine. Au contraire, la société de consommation a fait apparaître de nouveaux besoins cryptographiques. Bien entendu sont restées présentes les applications militaires des communications secrètes, ainsi que les problèmes bancaires. Mais en outre il faut assurer le secret des communications sur des réseaux télé-informatiques, ainsi qu'empêcher les pirates des temps modernes de s'infiltrer dans ces réseaux d'ordinateurs. La cryptographie sert de plus en plus dans des domaines variés.

Jusqu'à récemment, la sécurité de ces systèmes reposait sur une information secrète que se partageaient les utilisateurs, et qui permettait de communiquer de manière confidentielle. Pour cette raison, l'ensemble de ces systèmes est appelé *cryptographie à clé secrète.* Comme systèmes à clé secrète, on se sert d'algorithmes utilisant la même clé en chiffrement et en déchiffrement et qui, pour cela, sont appelés algorithmes de chiffrement symétriques. Le DES, l'AES, IDEA en sont les exemples les plus célèbres. Bien que ces algorithmes soient encore très utilisés pour le chiffrement de messages en raison du très haut débit, ceux-ci ne satisfont plus les besoins nouveaux. La *cryptographie à clé publique* a été formalisée et a permis de répondre à ces besoins. Il s'agit de l'ensemble des moyens qui permettent de fournir une solution aux nouveaux problèmes qui sont l'identification, l'authentification et la confidentialité de messages.

La cryptographie à clé publique intervient dans de nombreuses applications quotidiennes, de l'utilisation de cartes à puce en passant par la téléphonie mobile, jusqu'à la connexion d'un utilisateur à un ordinateur. Cependant, la sécurité de ces applications repose *essentiellement* sur deux problèmes réputés difficiles de théorie des nombres : le problème de factorisation et le problème du logarithme discret. Bien que ces deux problèmes résistent encore aux cryptanalystes, ils ne sont pas à l'abri d'une percée théorique ni même de l'ordinateur quantique qui mettrait en danger la difficulté de les résoudre.

Alors que la cryptographie tente de protéger les secrets des attaques d'éventuels fraudeurs, le codage a pour but de protéger les messages transmis des imperfections des systèmes de transmission tels que lignes téléphoniques, réseau hertzien, communication avec un satellite, disques compacts, bouche à oreille, etc . . . Les imperfections peuvent être entre autres : parasites, défauts sur la ligne, problèmes de compréhension.

La théorie des codes étudie donc les moyens de protéger l'information que l'on désire transmettre des altérations qu'elle pourrait subir, altérations provoquées par les imperfections du moyen physique de transmission. La méthode utilisée consiste à envoyer sur le canal plus de données que la quantité d'information à transmettre. Une redondance est ainsi introduite. Si cette redondance est structurée de manière exploitable, il est alors possible de corriger d'éventuelles erreurs introduites

sur le canal. On peut alors, malgré le bruit, retrouver l'intégralité des informations transmises au départ.

Une grande famille de codes correcteurs d'erreurs est constituée des codes par blocs. Pour ces codes l'information est d'abord coupée en blocs de taille constante et chaque bloc est transmis indépendamment des autres, avec une redondance qui lui est propre. La plus grande sous-famille de ces codes rassemble ce que l'on appelle les codes linéaires.

Dans un code linéaire, les messages que l'on veut coder sont lus sous la forme d'un k-uplet d'éléments d'un corps fini $\mathbb{F}$. Cet élément de $\mathbb{F}^k$ est ensuite transformé en élément de $\mathbb{F}^n$ par une application linéaire. La longueur n est choisie plus grande que la dimension k du code et c'est ainsi que la redondance est ajoutée.

En 1978, R.J. McEliece présenta un système de chiffrement à clé publique dont la sécurité reposait sur le problème du décodage borné d'un code correcteur d'erreurs. La clé privée est constituée d'un code correcteur structuré pour lequel on dispose d'un algorithme polynomial de décodage, et la clé publique est constituée d'une matrice génératrice du code, préalablement destructuré. Tandis que d'autres systèmes furent cryptanalysés avec succès, celui-ci est l'un des rares à encore résister à toute cryptanalyse.

Dans cette thèse, on s'intéresse à l'étude des systèmes de chiffrement ainsi que des schémas de signature dont la sécurité repose sur des problèmes difficiles de théorie des codes correcteurs d'erreurs. Ces activités de recherche ont été motivées, d'une part d'un point de vue théorique par la création de nouveaux schémas de signature avec des propriétés spéciales ainsi que d'une manière de réduire la taille de clés du schéma de McEliece, et d'autre part, d'un point de vue pratique visant à utiliser des propriétés structurelles afin d'obtenir des implémentations effectives d'un schéma de signature fondé sur les codes correcteurs d'erreurs.

Les résultats présentés ici sont le fruit de trois années de recherche, de novembre 2005 à septembre 2008, au sein de l'équipe PI2C du laboratoire XLIM de l'Université de Limoges. Ces travaux ont fait l'objet de plusieurs publications dans des conférences internationales [8, 28, 30, 31], ainsi que d'un chapitre d'ouvrage [29] et de plusieurs publications dans des conférences nationales [7, 10, 25, 26].

Les travaux les plus récents [2, 14, 24, 27], présentés dans ce document, suivent encore actuellement le processus d'acceptation et ne sont donc pas officiellement publiés.

Ce document est structuré en trois parties. Dans la première partie, composée de deux chapitres, on introduit les notions de base de la théorie des codes correcteurs d'erreurs et de la cryptographie, essentielles à la compréhension du reste du manuscrit. On détaille les motivations qui font de la théorie des codes correcteurs d'erreurs un outil intéressant pour la réalisation de primitives cryptographiques. On décrit les exemples les plus célèbres à savoir les cryptosystèmes de McEliece [86] et de Niederreiter [96], ainsi que les schémas de signature de Stern [130] et de Courtois, Finiasz et Sendrier [39].

La seconde partie, constituée de 4 chapitres, est dévolue à la description de schémas de signature basés sur les codes correcteurs d'erreurs. On présente une étude détaillée réalisée avec Philippe Gaborit et Emmanuel Prouff, de l'implémentation du schéma d'identification et de signature de Stern, à partir de la construction proposée dans [50], ainsi que de sa sécurisation contre les attaques SPA et DPA. Nous obtenons une authentification en 5,5 secondes et une signature en 22 secondes sur une carte à puce *sans crypto-processeur*. On détaille les bonnes propriétés de ce protocole dans un environnement à faibles ressources.

On décrit dans le chapitre suivant, l'étude que nous avons faite avec Ayoub Otmani et Damien Vergnaud, du schéma de signature de Kabatianski, Krouk et Smeets. Nous décrivons le fait qu'à partir de peu de signatures nous sommes capables de retrouver les paramètres secrets et donnons un moyen pour réduire la taille de la clé publique.

Ensuite, on présente un protocole de signature basé sur l'identité, réalisé en collaboration avec Philippe Gaborit et Marc Girault, en utilisant le schéma de Courtois, Finiasz et Sendrier ainsi que celui de Stern. Ce schéma de signature est le premier schéma non générique basé sur l'identité prouvé sûr (travail réalisé avec David Galindo) n'utilisant pas de problème basé sur la théorie des nombres.

Enfin, on décrit un travail effectué avec Carlos Aguilar Melchor et Philippe Gaborit. Il s'agit d'une nouvelle construction de signature de cercle à seuil dont la sécurité repose sur celle du schéma de Stern. Ce protocole est complètement anonyme et possède la plus petite taille de clé publique, de signature et de complexité pour un tel schéma.

La troisième et dernière partie, formée d'un seul chapitre, est dédiée à l'étude d'une construction permettant de réduire considérablement la taille de la clé publique des cryptosystèmes de McEliece ou Niederreiter. Ce travail a été effectué en collaboration avec Thierry Berger, Ayoub Otmani et Philippe Gaborit. Cette nouvelle construction nous donne des tailles de clé publique de l'ordre de $6\ 000$ bits. Notre étude apporte, du point de vue de la sécurité, la preuve que le décodage de codes quasi-cycliques est un problème NP-complet. Ce problème est lié au type de codes que nous utilisons dans notre système à savoir des codes alternants quasi-cycliques.

En annexe, on présente un travail effectué avec Frederik Armknecht, Philippe Gaborit et Olivier Ruatta, traitant des attaques algébriques de registre à décalage avec mémoire. Tout d'abord, on décrit de nombreuses améliorations pour le calcul d'annulateurs de certains ensembles. Ensuite, on présente comment utiliser ces résultats pour exclure l'existence d'équations de degré 3 pour E_0 avec un nombre de tops d'horloge compris entre 5 et 9. Ces résultats constituent les meilleurs résultats en ce qui concerne la non-existence d'équations de bas degré pour E_0.

On présente aussi brièvement un travail effectué avec Christophe Chabot et Abdelkader Necer traitant des codes cycliques sur des anneaux de matrices.

Notations

p un nombre premier, s un entier positif, $\mathbb{F}_q$ le corps fini à $q = p^s$ éléments.

Notations du Chapitre 1 :

Concernant les codes, nous noterons :

$\mathscr{C}$	un code linéaire
$\mathsf{wt}(u)$	poids de u
$supp(u)$	le support de u
$d(u, v)$	distance de Hamming de u et v
d	distance minimale du code
n	longueur du code
k	dimension du code
$[n, k, d]$	code de paramètres n, k, d
$\mathcal{G}$	matrice génératrice d'un code
$\mathcal{H}$	matrice de parité d'un code
g	polynôme générateur d'un code
r	ordre d'un code de Reed-Muller
m	nombre de variables d'un code de Reed-Muller
$\mathcal{RM}(r, m)$,	code de Reed-Muller d'ordre r en m
I_n	la matrice identité d'ordre n
A^T	la transposée de n'importe quelle matrice A

Concernant la cryptographie, nous noterons :

A	Alice
B	Bob
x	le texte clair
y	le texte chiffré
e	le vecteur erreur

Notations du Chapitre 2 :

En plus des notations précédentes, nous noterons :

C_A	fonction de chiffrement
D_A	fonction de déchiffrement
P	prouveur
V	vérifieur
s_P	clé secrète du prouveur
$a\|b$	concaténation de a et b

Notations du Chapitre 3 :

$I_n = \{0,1\}^n$	ensemble des chaînes binaires de longueur n
F_n	ensemble des fonctions de I_n vers I_n
P_n	ensemble des permutations de I_n vers I_n
Ψ	un tour du schéma Feistel
$f_1, ..., f_k$	k fonctions indépendantes choisies aléatoirement dans F_n
Ψ^k	$\Psi(f_1, ..., f_k)$
$\psi(f)$	tour de Feistel de f

Notations du Chapitre 4 :

$\mathbb{F}_q^n$	ensemble des vecteurs de longueur n sur $\mathbb{F}_q$
M	ensemble des messages de syndromes corrigibles
M_β	matrice représentant $x \mapsto x\beta$

Notations du Chapitre 5 :

$\frac{a}{b}$	symbole de Legendre
1^k	une suite de k uns
$\mathcal{O}_1, \mathcal{O}_2$	oracles
$\mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2,\cdots}(x, y, \ldots)$	algorithme avec en entrée $x, y, \ldots$ et $\mathcal{O}_1, \mathcal{O}_2, \ldots$

Notations du Chapitre 6 :

N	la taille du cercle
t	le nombre de signataires

Notations du Chapitre 7 :

β	un élément de $\mathbb{F}_q$
$\boldsymbol{\beta}^n$	le vecteur $(1, \beta, \ldots, \beta^{n-1})$
J_n	$\{0, \ldots, n-1\}$
$\mathcal{S}_n$	l'ensemble des permutations de J_n
$\mathbf{1}_n$	le vecteur tout-à-un $(1, \ldots, 1)$ de longueur n
$\boldsymbol{u} \otimes \boldsymbol{v}$	produit de Kronecker de $\boldsymbol{u}$ et $\boldsymbol{v}$
$A \otimes B$	la matrice $[a_{i,j}B]$ pour toutes matrices A et B
$\mathsf{diag}(\boldsymbol{v})$	la matrice carrée dont la diagonale est $\boldsymbol{v}$
$V_k(\boldsymbol{v})$	la matrice de Vandermonde
$\mathcal{S}_\ell \wr \mathcal{S}_{n_0}$	le produit en couronne de $\mathcal{S}_\ell$ par $\mathcal{S}_{n_0}$
$\mathscr{R}_k(\boldsymbol{\alpha}^n)$	le code de Reed-Solomon de longueur n sur $\mathbb{F}_q$
$\mathscr{G}_k(\boldsymbol{\lambda}, \boldsymbol{v})$	le code de Reed-Solomon Généralisé de dimension k sur $\mathbb{F}_q$
$\mathfrak{G}_{\mathbb{F}_q}(\ell, N_0, t)$	la classe de tous les codes de Reed Solomon quasi-cycliques d'ordre N_0

Première partie

Préliminaires

Chapitre 1

Cryptographie et théorie des codes

Les années 1990 ont été marquées par l'explosion des systèmes de communication, qui ont permis le développement des échanges électroniques, tant dans le domaine industriel et bancaire que dans celui du commerce en ligne et récemment celui des relations entre les citoyens et les administrations. Si, jusqu'à présent, l'ouverture et l' *interopérabilité* des réseaux et systèmes, ainsi que leurs performances, ont été privilégiées aux dépens de la sécurité, on assiste maintenant à une prise de conscience des problèmes par les acteurs de ces nouveaux réseaux, qui ont engagé des réflexions sur la sécurité et sur la cryptographie qui en constitue une brique fondamentale.

Longtemps réservée au domaine diplomatique et militaire, s'appuyant alors sur des principes mathématiques élémentaires, la cryptographie a commencé à évoluer vers le milieu du siècle avec le début des télécommunications en intégrant essentiellement des techniques de codage de l'information ; mais il a fallu attendre les années 1970 pour qu'elle passe du secret des laboratoires militaires au domaine public, et s'institue comme une véritable science dans le domaine universitaire. Beaucoup d'articles ont alors été publiés et des conférences publiques ont été instituées.

1.1 Cryptographie

1.1.1 Généralités

Dès que l'on désire communiquer de manière secrète, deux problèmes se posent. Il faut s'assurer :

- que l'information ne va pas être reçue par une personne autre que son destinataire légitime (menace contre la *confidentialité* de l'information). L'attaque est alors passive : l'*espion* écoute.
- que l'information ne provient pas d'une personne autre que son véritable auteur, ou n'a pas été modifiée durant son transfert (menace contre *l'intégrité* de l'information). L'attaque est donc active : l'*espion* dénature frauduleusement l'information.

La confidentialité et l'intégrité peuvent être assurées soit grâce à une protection physique du système de transmission, soit grâce à une protection logique des informations transférées.

La confidentialité est assurée en chiffrant le message envoyé, c'est-à-dire en le transformant de manière à le rendre illisible pour toute personne ne connaissant pas la clé de déchiffrement, tandis que l'intégrité est garantie par une signature électronique du message.

1.1.2 Le chiffrement

Il existe deux types d'algorithmes de chiffrement :

Les algorithmes *symétriques* (ou schémas à clé secrète)

Ils sont pour la plupart connus et utilisés depuis longtemps (dès l'Antiquité pour certains) : l'envoyeur *A* (appelé Alice par convention) et le receveur *B* (appelé Bob) possèdent la même clé.

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

Les algorithmes *asymétriques* (ou schémas à clé publique)

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

Un algorithme asymétrique est généralement basé sur un problème complexe, c'est-à-dire difficile à résoudre, de sorte qu'il n'existe pas d'algorithme efficace permettant de trouver la solution. L'efficacité d'un algorithme est susceptible d'évoluer en fonction des progrès des ordinateurs (gain en puissance et en rapidité), mais aussi en fonction des découvertes et des améliorations d'algorithmes permettant de résoudre les problèmes difficiles sous-jacents plus efficacement.

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

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

Ceci permet, contrairement aux algorithmes symétriques, de publier les clés (publiques, bien entendu), ce qui a pour avantage de permettre à n'importe qui d'envoyer un message à la personne qui a publié sa clé, ou de vérifier qu'un message a bien été rédigé par une personne dont la clé est dans l'annuaire. En contrepartie, cet annuaire étant accessible en lecture à tous, *A* voulant se faire passer pour *B* n'a qu'à remplacer la clé publique de *B* par la sienne pour parvenir à ses fins. Pour empêcher une telle fraude, il faut généralement une autorité qui certifie les clés publiques. Nous verrons dans le chapitre 5 page 87 d'autres moyens de gérer les clés publiques.

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

1.1.3 Classification de Shamir des protocoles de reconnaissance

En 1986, Fiat et Shamir ont classifié les différentes catégories de protocoles de reconnaissance. Ils distinguèrent alors trois niveaux de protection. Supposons qu'Alice veuille prouver son identité à Bob, les différents niveaux sont alors :

1. ***protocole d'authentification*** : Alice peut prouver à Bob qu'elle est véritablement Alice, mais personne d'autre ne peut prouver à Bob qu'elle est Alice.
2. ***protocole d'identification*** : Alice peut prouver à Bob qu'elle est véritablement Alice, et Bob ne peut extraire aucune information de cette preuve pour tenter de faire croire à quelqu'un d'autre qu'il est Alice.
3. ***protocole de signature*** (ou ***non-répudiation***) : ce protocole a les mêmes propriétés que le précédent avec, en plus : Alice ne peut pas dire qu'elle n'a pas prouvé son identité à Bob, ou que le message qu'elle a signé était différent de celui que Bob prétend avoir reçu (elle ne peut pas répudier sa signature). En d'autres termes, Bob ne peut pas faire croire qu'Alice lui a prouvé son identité alors qu'elle ne l'a pas fait.

En fait, le protocole d'authentification au sens de Fiat et Shamir ne sera utilisé que lorsque Alice et Bob ont leurs intérêts en communs, pour prévenir une attaque venue de l'extérieur. Par contre, lorsque Alice et Bob ont des intérêts divergents, l'un des deux autres types de schéma doit être employé. Avec un protocole d'authentification, Bob pourrait a posteriori se faire passer pour Alice auprès d'un troisième intervenant. Ceci n'est plus possible lorsque l'on a affaire à un schéma d'identification. En revanche, dans ce cas, comme dans le cas de l'authentification, Bob peut faire croire qu'il a reçu un message venant d'Alice (en quelque sorte en *imitant* sa signature), tandis qu'un protocole de signature l'en interdit.

Nous n'utiliserons cependant pas le vocabulaire de Shamir, en ne faisant pas de distinction entre authentification et identification. En effet l'authentification au sens de Shamir présuppose qu'Alice et Bob ne sont pas rivaux, mais aussi que personne ne peut *écouter* une preuve d'authentification entre Alice et Bob, car sinon l'indélicat personnage serait dans les mêmes conditions que Bob pour restituer la clé secrète d'Alice, et pourrait donc se faire passer pour Alice lors des conversations ultérieures. Or, la confidentialité physique est généralement loin d'être parfaitement assurée.

1.1.4 La signature

La signature numérique est un mécanisme permettant d'authentifier l'auteur d'un document électronique et de garantir son intégrité, par analogie avec la signature manuscrite d'un document papier. Un mécanisme de signature numérique doit présenter les propriétés suivantes :

- Il doit permettre au lecteur d'un document d'identifier la personne ou l'organisme qui a apposé sa signature.
- Il doit garantir que le document n'a pas été altéré entre l'instant où l'auteur l'a signé et le moment où le lecteur le consulte.

Pour cela, les quatre conditions suivantes doivent être réunies :

1. **Authentique :** L'identité du signataire doit pouvoir être retrouvée de manière certaine.
2. **Infalsifiable :** La signature ne peut pas être falsifiée. Personne ne peut se faire passer pour un autre.

3. **Non réutilisable :** La signature n'est pas réutilisable. Elle fait partie du document signé et ne peut être déplacée sur un autre document.
4. **Inaltérable :** Un document signé est inaltérable. Une fois qu'il est signé, on ne peut plus le modifier.
5. **Irrévocable :** La personne qui a signé ne peut le nier.

La signature électronique n'est devenue possible qu'avec la cryptographie asymétrique.

1.1.5 Fonctionnement de la signature numérique

Supposons qu'Alice dispose d'un algorithme de chiffrement à clé publique. Notons C_A, la fonction de chiffrement et D_A celle de déchiffrement. Rappelons que la fonction C_A est connue de tous, par la clé publique associée à l'algorithme, tandis que D_A n'est connue que par Alice la seule à détenir la clé privée.

Lorsque Alice souhaite signer un message M, elle calcule $S = C_A(M)$. Toute personne disposant du message M et de la signature S peut alors vérifier qu'Alice est à l'origine de la signature en calculant $D_A(S)$. Si cette quantité est bien égale à M, alors on peut être certain qu'Alice est l'auteur de la signature, car seule elle peut produire $D_A(M)$, puisqu'elle est la seule à connaître D_A. On peut être également sûr que le message n'a pas été altéré. En effet, pour altérer le message, il faudrait également altérer la signature de manière cohérente, ce qui n'est possible que si l'on dispose de D_A.

En définitive, ce n'est jamais un message M qu'Alice signe, mais l'empreinte de M par une fonction de hachage. La sûreté de la signature dépend alors du soin apporté au choix de la fonction de hachage. Il faut, qu'étant donné un message et son empreinte, il soit très difficile de fabriquer un message ayant une empreinte (et donc une signature) égale. L'intérêt de la fonction de hachage est de permettre de signer une quantité de données beaucoup plus petite que le message entier (et de longueur fixe).

La signature numérique nécessite l'utilisation de certificats électroniques. Ceux-ci sont générées par des autorités de certification, qui permettent d'identifier de façon unique la personne (ou l'entité) qui détient les clés publique et privée : les certificats peuvent être vus comme la carte d'identité numérique de cette personne ou de cette entité.

1.1.6 Protocoles à divulgation nulle de connaissance

Supposons qu'Alice veuille montrer à Bob qu'elle connaît un secret ou qu'elle sait faire quelque chose, sans toutefois révéler ce secret ou ce savoir-faire, ni même donner la moindre indication sur ce qui doit rester caché. Pour ce faire elle utilise un protocole à divulgation nulle de connaissance *zero-knowledge*.

On ne peut arriver à faire une preuve à divulgation nulle de connaissance qu'avec une probabilité négligeable d'erreur, c'est-à-dire avec une probabilité très faible de croire les affirmations d'Alice alors qu'elle ne sait rien. C'est une idée qui convient parfaitement au problème d'authentification. En effet, Alice veut montrer qu'elle est bien elle-même, et par conséquent qu'elle possède vraiment la clé secrète relative à la clé publique à son nom, sans cependant révéler des renseignements susceptibles de permettre à Bob ou à un espion de reconstituer, même de façon partielle, sa clé secrète.

La plupart des protocoles d'authentification (au sens large) sont des algorithmes interactifs sans révélation.

Principe intuitif de protocole à divulgation nulle de connaissance

Alice joue avec Bob à une variante du jeu : *Où est Charlie ?* Le principe de ce jeu est de retrouver un personnage (puis un autre) *caché* dans une image. Oscar regarde Alice et Bob jouer à ce jeu là. Alice veut être certaine que Bob a trouvé un personnage (par exemple Charlie) sans qu'Oscar n'en déduise la position de ce personnage dans l'image.

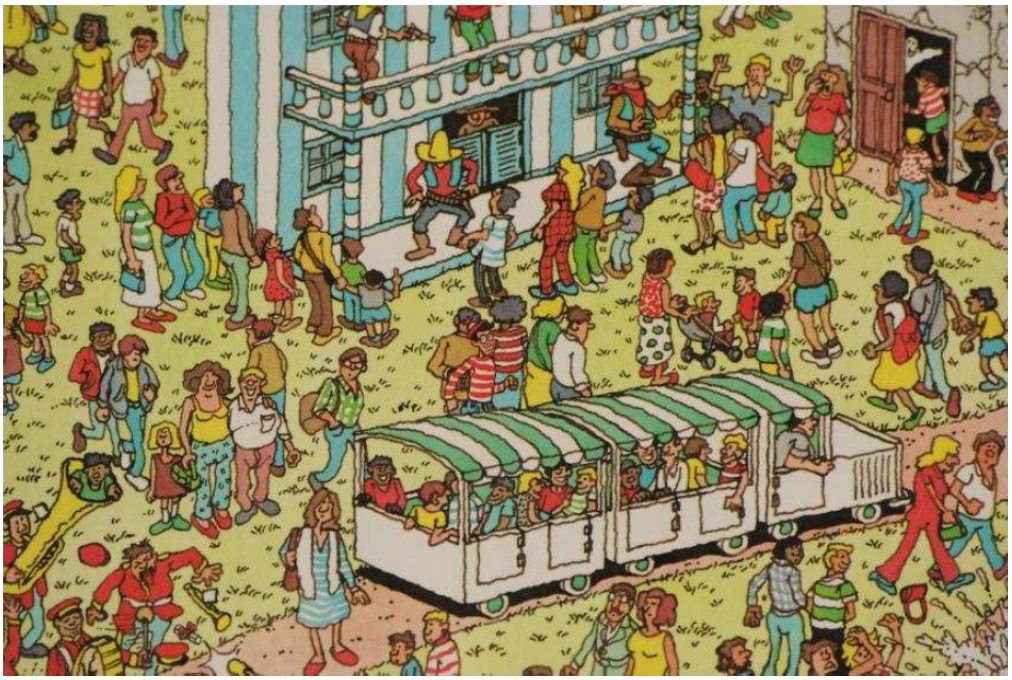


FIGURE 1.1 – Où est Charlie ?

Comment Bob peut-il prouver à Alice qu'elle a trouvé Charlie sans révéler où il est ?

Une solution à ce problème est la suivante. Bob photocopie l'image et découpe (à l'abri des regards indiscrets) le portrait de Charlie. Bob transmet ensuite le portrait de Charlie à Alice qui est ainsi convaincue que Bob a trouvé Charlie. Oscar qui a observé le jeu n'est pas capable de retrouver Charlie à l'intérieur de l'image. Les deux protagonistes jouent ensuite à ce jeu avec un nouveau personnage, jusqu'à ce qu'Alice soit convaincue que ce n'est pas le hasard qui a permis à Bob de trouver le personnage.

Un exemple

Considérons un exemple de processus à divulgation nulle de connaissance en particulier :

Supposons qu'Alice prétende savoir résoudre l'équation $x^2 = m \mod n$ (en supposant que m est un carré modulo n). Elle veut en convaincre Bob. Pour cela, elle choisit au hasard $x' \in]1, n-1[$ et calcule $m' = x'^2 \mod n$.

Soit les problèmes suivants :

(P1) *trouver* x *tel que* $x^2 = m' \bmod n$;
(P2) *trouver* x *tel que* $x^2 = mm' \bmod n$.

Alice donnera la solution d'un de ces deux problèmes, suivant le choix de Bob. Il y a deux cas de figure :

1. Si Bob choisit (P1) Alice répondra x';
2. Si Bob choisit (P2), elle répondra x_0x' où x_0 est la solution du problème initial.

Bob peut vérifier qu'Alice ne lui ment pas. La seule stratégie gagnante pour Alice, si elle ne sait pas résoudre la première équation, est de deviner le choix que va effectuer Bob. En effet, Alice peut donner une réponse vraie au problème (P1) pour peu qu'elle fabrique m' comme indiqué ci-dessus, mais dans ce cas, elle ne connaîtra pas la solution de (P2) si elle ne connaît pas x_0 ; cependant elle peut fabriquer m ainsi : soit x' aléatoire, soit $m'' = x'^2$ et $m' = m''m^{-1} \bmod n$. Dans ce cas, si Bob choisit (P2), alors Alice répondra x'. Bob la croira. Par contre elle ne saurait pas répondre à (P1). Un tricheur a, par conséquent, une chance sur deux de réussir. Il suffit de répéter un certain nombre de fois le schéma précédent jusqu'à ce que Bob soit convaincu (i.e. jusqu'à ce que la probabilité qu'Alice triche soit négligeable). En outre, Bob ne peut rien déduire sur x_0 de ces échanges successifs.

Un protocole d'authentification est dit à divulgation nulle de connaissance lorsque le prouveur est le seul à connaître son secret et que les informations qu'il divulgue, ne permettent pas de retrouver ce secret, ni d'usurper son identité. La plupart des protocoles sont basés sur des problèmes arithmétiques difficiles à résoudre, comme la factorisation, le logarithme discret ou le problème RSA.

Ce type d'authentification trouve une application dans le domaine des cartes à puce, permettant d'identifier leurs utilisateurs.

1.1.7 Heuristique de Fiat-Shamir

Dans [46], A. Fiat et A. Shamir définissent la notion de procédé d'identification sans divulgation d'information, ainsi qu'une heuristique, dite heuristique de Fiat-Shamir, qui permet de transformer un procédé d'identification sans divulgation d'information en schéma de signature.

Un procédé d'identification sans divulgation d'information comprend un procédé de génération de clé publique et privée ainsi qu'un protocole d'identification. Ce protocole d'identificatoin se décrit à l'aide de trois étapes, entre un prouveur P et un vérifieur V.

Les trois étapes se définissent à partir de 3 fonctions publiquement connues A, B et C. Le protocole se déroule de la manière suivante :

– étape 1 : P génère un nombre aléatoire r, calcule $x = A(r)$ et envoie x à V;
– étape 2 : V génère un nombre aléatoire g et l'envoie à P;
– étape 3 : P calcule, à l'aide de son secret s, le nombre $y = B(s, g, r)$, et l'envoie à V. V peut alors vérifier si $x = C(y, g)$, auquel cas l'identification est acceptée.

Le protocole d'identification peut également comporter la vérification que les éléments sont dans des ensembles connus. Un moyen simple de vérifier l'appartenance de ces éléments à ces ensembles est de ne définir les fonctions A, B, C que sur ces ensembles.

L'heuristique Fiat-Shamir consiste à supprimer l'interactivité du procédé d'identification sans divulgation d'information : pour cela, on utilise une fonction de hachage, appelée h. Le signataire est appelé S, tandis que le vérifieur est appelé V.

Le schéma de signature s'obtient grâce au procédé de génération de clé du procédé d'identification sans divulgation d'information, et par le procédé de génération de signature suivant :

- génération d'un nombre aléatoire r et calcul de $x = A(r)$;
- calcul de $g = h(m, x)$, où m est le message à signer ;
- calcul de $y = B(s, g, r)$; la signature alors obtenue est (x, y).

Enfin, le procédé de vérification de la signature (x, y) du message m consiste à calculer $g = h(m, x)$, puis à vérifier que $x = C(y, g)$.

1.2 Théorie des codes

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

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

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

1.2.1 Définitions

Définition 1.2.1

Soit un alphabet A, soit A^* l'ensemble des vecteurs formés à partir des éléments de A. Un code $\mathscr{C}$ est un sous-ensemble de vecteurs de A^*. $\mathscr{C}$ sera dit binaire si $A = \{0, 1\}$, un élément de $\mathscr{C}$ est appelé mot de code.

Définition 1.2.2

Soient u et v deux vecteurs binaires de taille n ($u = (u_i)_{i=1,...n}$ avec $u_i = 0$ ou 1, idem pour v). On définit :

- *le poids de u (noté $\mathsf{wt}(u)$) comme le nombre de 1 dans u.*
  On définit ainsi une norme sur $\mathbb{F}_2^n$,
- *le support de u noté $supp(u)$ comme l'ensemble des indices i tels que $u_i = 1$,*
- *la distance de Hamming de u et v (notée $d(u, v)$) comme le cardinal de l'ensemble des indices i tels que u_i soit différent de v_i. La fonction $d(.,.)$ ainsi définie vérifie les propriétés*

d'une distance sur $\mathbb{F}_2^n$. On a $d(u,v) = \mathsf{wt}(u+v)$.

La distance minimale (notée d) d'un code est le minimum des distances de Hamming entre les mots du code pris distincts deux à deux.

C'est aussi le poids du mot (non nul) de plus petit poids du code.

Définition 1.2.3

On notera $[n,k,d]$ un code de longueur n de dimension k et de distance minimale d.

Définition 1.2.4

Soit $\mathcal{C}$ un code linéaire $[n,k,d]$ sur $\mathbb{F}_q$, une matrice génératrice $\mathcal{G}$ est une matrice $(k \times n)$ dont les lignes forment une base de $\mathcal{C}$. Si c appartient à $\mathscr{C}$ alors c est une combinaison linéaire de lignes de $\mathcal{G}$. Si $\mathcal{G} = (I_k | A_{k\times(n-k)})$ avec I_k la matrice identité de dimension k et $A_{k\times(n-k)}$ une matrice $(k, n-k)$, on dit que $\mathcal{G}$ est sous forme systématique.

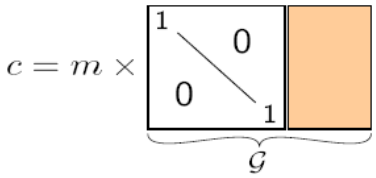


FIGURE 1.2 – $\mathcal{G}$: Matrice génératrice sous forme systématique

Définition 1.2.5

Soit $\mathcal{C}$ un code (non nécessairement linéaire) dans $\mathbb{F}^n$. Le code dual de $\mathcal{C}$, noté $\mathcal{C}^\perp$, est le code :

$$\mathcal{C}^\perp = \{x \in \mathbb{F}^n | x * c = 0, \text{ pour tout } c \in \mathcal{C}\},$$

où $x * c$ dénote le produit usuel.

Une matrice génératrice $\mathcal{H}$ du code dual $\mathcal{C}^\perp$ d'un code linéaire $\mathcal{C}$ est appelée matrice de parité de $\mathcal{C}$.

On a

$$\mathscr{C} = \ker \mathcal{H}.$$

Si la matrice génératrice $\mathcal{G}$ est sous forme systématique alors :

$$\mathcal{H} = \left(-A^T_{(n-k)\times k} | I_{n-k}\right),$$

où A^T est la transposée de A.

Définition 1.2.6

Deux codes sont dits équivalents si leurs deux matrices génératrices se déduisent l'une de l'autre par permutation de colonnes.

Pour encoder un message m on le multiplie par une matrice génératrice $\mathcal{G}$. On obtient un mot de code c. Ensuite après passage par un canal bruité, une erreur e s'ajoute au mot de code c on obtient $c' = c + e$.

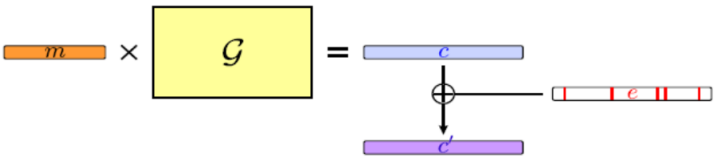


FIGURE 1.3 – Décodage

En calculant le produit $\mathcal{H}c'$ on obtient un vecteur s qu'on appelle le syndrome de l'erreur e.

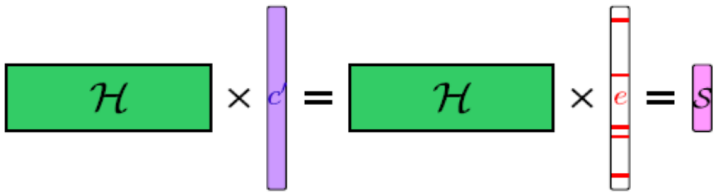


FIGURE 1.4 – Syndrome

Décoder dans un code $\mathcal{C}$ désigne l'action d'associer un mot du code (un élément du sous-espace vectoriel) à un mot de l'espace vectoriel. On cherche le plus souvent à décoder en associant à un mot le mot de code duquel il est le plus proche. Cependant il faut d'abord décider du sens que l'on veut donner à l'expression le plus proche.

Dans le cas du décodage d'un message transmis le long d'un canal bruité, on s'intéresse essentiellement au *décodage au maximum de vraisemblance*. Cela consiste à toujours associer le mot de code qui a la plus grande probabilité d'avoir donné ce mot en étant transmis sur le canal.

Définition 1.2.7

La majoration suivante est vérifiée pour tous les codes linéaires. Elle se nomme borne du singleton :

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

Si la borne de Singleton est atteinte, le code est dit MDS.

Codes détecteurs d'erreurs et code de parité

Les codes détecteurs d'erreurs consistent à détecter des erreurs lors d'une transmission sur un canal bruité. On peut protéger autant de symboles que l'on veut avec un symbole. Ces codes sont très simples et économes donc très utilisés (calcul de CRC dans les fichiers informatiques, numéro RIB, numéro de sécurité sociale, transmission modems). Le code de parité est un code $[k+1, k, 2]$, dont le principe est d'ajouter un bit au mot de sorte que le nombre de bits à 1 du mot soit pair. Il suffit alors de vérifier que le poids du mot reçu est pair pour savoir si une erreur de transmission a été commise. On est alors en mesure de dire s'il y a eu une erreur mais on ne peut pas la corriger. De plus on ne peut détecter qu'une seule erreur. Ce codage est très rapide mais ne permet aucune correction d'erreur. Il est particulièrement utilisé dans les modems, puisque chaque octet en contient un ce qui permet au modem de redemander un octet erroné, avec une perte de temps minimale.

Usuellement, on considère que le mot de code émis est celui se trouvant le plus près du mot reçu, ce qui revient à supposer que le minimum de lettres a été modifié. Ce procédé conduit à une erreur de décodage chaque fois que l'erreur est supérieure à la capacité corrective du code. La question naturelle est celle de la valeur de t correspondant au nombre maximum d'erreurs corrigibles.

Une interprétation géométrique donne un élément de réponse. les boules fermées de rayon t centrées sur les mots de code doivent être disjointes. La capacité de correction d'un code correspond au plus grand entier t vérifiant cette propriété, c'est aussi le plus grand entier strictement plus petit que $\frac{d}{2}$ où d représente la distance minimale du code.

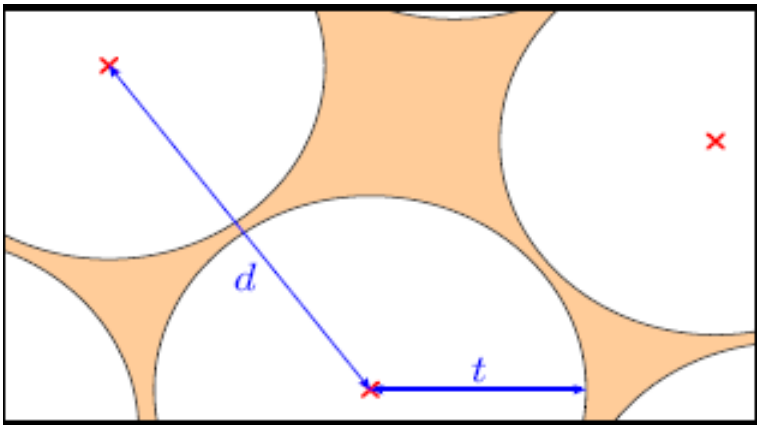


FIGURE 1.5 – Sphères de rayon t centrées sur les mots du code

Il existe une configuration idéale, correspondant au cas où les boules fermées de rayon t ayant comme centre les mots du code forment une partition de l'espace ambiant. Si la transmission ne produit jamais plus d'une altération, alors l'erreur est corrigible. Il n'existe aucune redondance inutile, le code est le plus compact possible pour garantir la correction certaine d'une erreur. De tels codes sont dit *parfaits*.

Code à répétitions

Plaçons nous dans le cas binaire. Supposons les messages constitués d'une seule lettre, par exemple 0, le code à triple répétition donne 000. Comme l'alphabet ne contient que deux lettres, deux au moins sur trois des lettres d'un élément de $\mathbb{F}$ sont semblables, en conclusion tout mot de $\mathbb{F}$ est à distance de 1 d'un mot du code. De plus, un mot de $\mathbb{F}$ n'est à une distance d'au plus 1 que d'un unique mot du code, ce qui démontre que ce code est parfait.

Cette propriété n'existe plus si le code contient plus de deux lettres, en effet il existe des éléments de $\mathbb{F}$ constitués de trois lettres différentes et donc à une distance de 2 de trois mots différents du code et à distance de 1 d'aucun mot du code.

Code de Hamming

Un code de Hamming est un code correcteur linéaire. Il permet la détection et la correction automatique d'une erreur si elle ne porte que sur une lettre du message.

Un tel code est parfait, ce qui signifie que pour une longueur de code donnée, il n'existe pas d'autre code plus compact ayant la même capacité de correction. En ce sens, son rendement est maximal.

Il existe une famille de codes de Hamming. Le plus célèbre, après le code de répétition binaire de dimension trois et de longueur un, est sans doute le code binaire de paramètres $[7, 4, 3]$. Pour chaque alphabet ayant pour nombre de lettres une puissance d'un nombre premier et pour chaque longueur l de code, il existe un code de Hamming utilisant cet alphabet et de longueur au moins égale à l.

Codes cycliques

On définit la fonction décalage sur $\mathbb{F}_q$, qui est une permutation circulaire des coordonnées :

$$s : (c_0, c_1, \ldots, c_{n-1}) \rightarrow (c_{n-1}, c_0, c_1, \ldots, c_{n-2})$$

Définition 1.2.8

Soit $\mathcal{C}$ un code linéaire sur $\mathbb{F}_q$. On dit que $\mathcal{C}$ est cyclique si $s(\mathcal{C}) = \mathcal{C}$.

On vérifie facilement que, si $\mathcal{C}$ est cyclique, $\mathcal{C}^{\perp}$ est aussi cyclique. Un exemple très simple de code cyclique est le code de contrôle de parité.

Soit $\mathcal{C}$ un code linéaire de longueur n.À chaque mot m de $\mathcal{C}$ on associe un polynôme $m(x)$

$$m = (a_0, a_1, \cdots, a_{n-1}) \rightarrow m(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$$

$$(a_{n-1}, a_0, \cdots, a_{n-2}) \rightarrow m(x) = a_{n-1} + a_0 x + \cdots + a_{n-2} x^{n-1}$$

ou

$$a_{n-1} + a_0 x + \cdots + a_{n-2} x^{n-1} = x.m(x) (\mathrm{mod} x^n - 1)$$

Définition 1.2.9

La représentation polynomiale d'un code $\mathcal{C}$, notée $\mathcal{C}(x)$, est l'ensemble des représentations polynomiales $m(x)$ des mots m de $\mathcal{C}$.

Proposition 1.2.1

$\mathcal{C}$ est cyclique si et seulement si $\forall m \in \mathcal{C}, x.m(x)(\mathrm{mod} x^n - 1)$ est la représentation polynômiale de $\mathcal{C}$

Propriété 1.2.1

- *si $x.m(x) \in \mathcal{C}(x)$ alors $\forall i \in \mathbb{N}, x^i.m(x) \in \mathcal{C}(x)$*
- *$\mathcal{C}$ est cyclique si et seulement si tout multiple modulo $x^n - 1$ d'un polynôme de $\mathcal{C}(x)$ est aussi dans $\mathcal{C}(x)$.*

Proposition 1.2.2

- *$\mathbb{F}_q[x]/(x^n - 1)$ est un anneau commutatif.*
- *l'application $\Phi : (a_0, a_1, \cdots, a_{n-1}) \rightarrow m(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$ est un morphisme d'espace vectoriel.*
- *$\mathcal{C}(x) = \Phi(C)$ est un sous groupe additif de $\mathbb{F}_q[x]/(x^n - 1)$.*
- *$\mathcal{C}$ est cyclique si et seulement si $\mathcal{C}(x)$ est un idéal.*

Théorème 1.2.1

$\mathcal{C}$ est cyclique si et seulement si $\mathcal{C}(x)$ est un idéal de $\mathbb{F}_q[x]/(x^n - 1)$.

Codes de Reed-Solomon

La matrice de Vandermonde $V_k(\boldsymbol{v})$ où $\boldsymbol{v} \in \mathbb{F}_q^n$ (k étant un entier positif) est définie par :

$$V_k(\boldsymbol{v}) = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ v_0 & v_1 & \cdots & v_{n-1} \\ \vdots & \vdots & & \vdots \\ v_0^{k-1} & v_1^{k-1} & \cdots & v_{n-1}^{k-1} \end{bmatrix}.$$

Définition 1.2.1

Soit α un élément primitif de $\mathbb{F}_q$ et supposons que $n = q - 1$. Le code de Reed-Solomon $\mathscr{R}_k(\boldsymbol{\alpha}^n)$ est le code de longueur n sur $\mathbb{F}_q$ définie par la matrice génératrice $V_k(\boldsymbol{\alpha}^n)$.

Nous verrons plus en détail cette famille de codes dans le chapitre 7 page 121.

Autres familles de codes

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

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

Chaque élément de cette matrice est ensuite décomposé en m éléments de $\mathbb{F}_p$, placés en colonnes, en utilisant une projection de $\mathbb{F}_{p^m}$ dans $\mathbb{F}_p^m$. On passe ainsi d'une matrice de taille $t \times n$ sur $\mathbb{F}_{p^m}$ à une nouvelle matrice de parité H de taille $mt \times n$ sur $\mathbb{F}_p$. Les éléments du code $\Gamma(L, g)$ seront donc tous les éléments c de $\mathbb{F}_p^n$ tels que

$$H \times c^T = 0.$$

C'est donc un code de longueur n et dimension $k \geq n - mt$. De plus, un tel code a une distance minimale au moins égale à $t + 1$. En effet, toute sous-matrice carrée $t \times t$ de H est inversible car H s'écrit comme le produit d'une matrice de Vandermonde et d'une matrice diagonale inversible :

$$H = \begin{pmatrix} 1 & \ldots & 1 \\ \vdots & \vdots & \vdots \\ \alpha_0^{t-1} & \ldots & \alpha_{n-1}^{t-1} \end{pmatrix} \times \begin{pmatrix} \frac{1}{g(\alpha_0)} & & 0 \\ & \vdots & \\ 0 & & \frac{1}{g(\alpha_{n-1})} \end{pmatrix}$$

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

Nous verrons ques les codes de Goppa sont utilisés en cryptographie notamment dans le schéma de signature de Courtois, Finiasz et Sendrier [39].

Codes BCH, Reed Muller, Golay Il existe beaucoup d'autres classes de codes par blocs, toutes présentant leurs intérêts, soit du point de vue de la capacité de correction, soit de la simplicité du codage ou du décodage.

On ne fait que mentionner leur existence car leur connaissance n'est pas nécessaire à la compréhension de cette thèse. Ils n'est en revanche pas exclu que des applications cryptographiques de la théorie algébrique des codes puissent passer par l'utilisation de ces codes : c'est d'ailleurs déjà le cas pour certains problèmes (recherche de fonctions booléennes, traitor tracing. . .) que l'on n'exposera pas dans ce document.

1.2.2 Problèmes difficiles en théorie des codes

Voici une liste de quelques problèmes difficiles liés à la théorie des codes correcteurs d'erreurs ainsi qu'une borne permettant de déterminer les paramètres d'un code. Il existe de nombreux autres problèmes dérivés de ces problèmes très généraux utilisés en pratique. Nous définissons ici un dérivé du problème de décodage par syndrome dans le cas des codes quasi-cycliques.

On peut remarquer que ces problèmes, en plus d'être calculatoirement difficiles sur une machine de Turing classique, restent calculatoirement difficiles quand on essaye de les résoudre à l'aide d'une machine quantique. Contrairement aux problèmes de factorisation et de logarithmes discrets, aucun algorithme actuellement connu ne permet d'attaquer ces problèmes liés aux codes correcteurs d'erreurs.

La borne de Gilbert-Varshamov permet de déterminer des paramètres pour lesquels on est sûr qu'un tel code linéaire existe, mais elle n'est pas constructive. Elle s'énonce ainsi :

Proposition 1.2.3 (*Borne de Gilbert Varshamov*)

Il existe un code linéaire de dimension k, de longueur n sur $\mathbb{F}_q$ et de distance $\geq d$ dès lors que :

$$q^{n-k} > \sum_{i=0}^{d-2} \binom{n}{i} (q-1)^i := V_q(n-1, d-2).$$

Preuve [*Borne de Gilbert Varshamov*] Soit $\mathcal{C}$ un code comme décrit ci-dessus. Construisons sa matrice de parité $\mathcal{H}$. On choisit pour la première colonne n'importe quel vecteur de $\mathbb{F}_q^{n-k}$ non nul, puis pour la seconde un vecteur non multiple scalaire de la première, et ainsi de suite jusqu'à obtenir $j-1$ colonnes dont $d-1$ sont toujours linéairement indépendantes. Par combinaison linéaire des $d-2$ ou moins de ces $j-1$ colonnes, on peut former au plus :

$$\sum_{i=0}^{d-2} \binom{j-1}{i} (q-1)^i = V_q(j-2, d-2)$$

colonnes.

Comme par combinaison linéaire de $d-2$ ou moins de ces $j-1$ colonnes, on peut former au plus :

$$\sum_{i=0}^{d-2} \binom{j-1}{i} (q-1)^i = V_q(j-2, d-2)$$

colonnes, on obtient une condition suffisante pour l'existence d'une j-ième colonne cherchée. Elle est donnée pour un choix de $j-1 \leq n-1$ colonnes :

$$q^{n-k} > \sum_{i=0}^{d-2} \binom{n}{i} (q-1)^i = V_q(n-1, d-2)$$

(on obtient le nombre maximal possible des combinaisons linéaires de $d-2$ colonnes choisies parmi $n-1$ colonnes). Ainsi l'inégalité de la proposition 1.2.3 est vérifiée, on peut trouver une j-ième colonne linéairement indépendante de toutes $d-2$ des $j-1$ premières colonnes, et ce jusqu'à obtenir $\mathcal{H}$ de rang $n-k$, ce qui fournit un code de distance $\geq d$. □

Les idées de cette démonstration seront utilisées par la suite.

Pour un tel code on suppose qu'il n'est pas possible de trouver un mot c en dehors de $\mathcal{C}$ à distance $\geq d$ de tous mots du code. Alors :

$$q^n \leq \sum_{i=0}^{d-1} \binom{n}{i} (q-1)^i M = V_q(n, d-1).$$

(c'est-à-dire, le rayon de recouvrement est $\leq d-1$). Dans le cas contraire on trouve un tel mot c et on l'ajoute au code $\mathscr{C}$ (non nécessairement linéaire), à partir du code trivial comportant un seul mot (on raisonne par récurrence sur M). Ceci dit, il existe un code de cardinal M pour le nombre :

$$M = \frac{q^n}{V_q(n, d-1)} - 1 \Rightarrow M < \frac{q^n}{V_q(n, d-1)} \leq M + 2.$$

La borne de Gilbert-Varshamov est une minoration de la distance minimale des codes. On suppose habituellement, bien que cela n'a jamais été prouvé, que les codes linéaires binaires générés par une matrice aléatoire satisfont cette borne. Elle a une valeur voisine de $0,11n$ lorsque $n = 2k$ (valeurs que nous emploierons généralement par la suite), ce qui permet de dire qu'il y a de fortes chances qu'il n'y ait pas de mots non nuls du code de poids inférieur à $0,11n$.

Pour un code linéaire quelconque sur $\mathbb{F}_q$, il est montré que le nombre moyen de mots de poids w d'un code était proche de :

$$\binom{n}{w}(q-1)q^{n-k} \tag{1.1}$$

mais cette formule n'a pas été prouvée pour les codes binaires (cas $q = 2$), bien qu'elle ait des chances de ne pas être trop éloignée de la vérité. En effet, pour x aléatoire, les événements $(\mathcal{H}x = i)$ sont équiprobables, et en supposant que les mots du code soient répartis aléatoirement, suivant une loi binomiale de probabilité élémentaire $p = 1/2$ (ce qui est loin d'être prouvé), on a :

$$card\{x, \mathcal{H}x = 0\} = card\{Ker\mathcal{H}\} = 2^{n-k}$$

$$P(|x| = w) = \frac{\binom{n}{w}}{2^n}.$$

On pourrait en déduire (1.1) pour $q = 2$. On remarque, expérimentalement, que, pour un code binaire aléatoire, cette formule donne un nombre non nul de mots de poids w si w est supérieur à la borne de Gilbert-Varshamov (ce nombre croît alors extrêmement rapidement avec w), et un nombre nul si w est inférieur à celle-ci.

Au coeur de la plupart des cryptosystèmes utilisant des codes correcteurs d'erreurs, il y a un problème difficile : *le problème de décodage par syndrome* (Syndrome Decoding problem noté SD).

Berlekamp, Mac Eliece et Van Tilborg ont montré en 1978 [15] que le problème de la recherche de mots de poids et de syndrome fixé était un problème NP complet. C'est en fait la résolution du système :

$$\mathcal{H}x = i, |x| = w$$

où $\mathcal{H}$ est une matrice binaire, i un vecteur donné (syndrome de x) et w un entier fixé (poids de x), x étant l'inconnue.

Définition 1.2.10 (*Problème de décodage par syndrome (SD)*)

Entrée : Soit $\mathcal{H}$ une matrice $(n-k,n)$ binaire et w un entier et un syndrome $s \in {\mathbb{F}_2}^{n-k}$
Sortie : Un mot $e \in \mathbb{F}_2^n$ telle que $\mathsf{wt}(e) \leq w$ et $\mathcal{H}e^T = s$

Ce problème a été utilisé par Stern dans son protocole (que nous verrons dans le chapitre suivant), mais quelques années plus tard, une variation de ce problème appelée problème de distance minimale (Minimum Distance) a été prouvé NP-complet dans [134] :

Définition 1.2.11 (*Problème de distance minimale (MD)*)

Entrée : Soit $\mathcal{H}$ une matrice $(n-k,n)$ binaire et w un entier > 0.
Question : Existe-t-il un vecteur x non nul de $\mathbb{F}_2^n$ de poids $\leq w$ tel que $Hx^T = 0$?

Le problème du décodage borné de codes de Goppa (Goppa Parameterized Bounded Decoding problem (GPBD)) est un cas particulier du problème SD. Ce problème a aussi été prouvé NP-difficile dans [47].

Définition 1.2.12 (*Problème du Décodage Borné des Codes de Goppa (GPBD)*)

Entrée : Soit $\mathcal{H}$ une matrice $(n-k,n)$ binaire (la matrice de parité d'un code de $\mathsf{Goppa}(n,k)$) et un syndrome $s \in {\mathbb{F}_2}^{n-k}$
Sortie : un mot $e \in \mathbb{F}_2^n$ tel que $\mathsf{wt}(e) \leq \frac{n-k}{\log_2 n}$ et $\mathcal{H}e^T = s$

Un autre problème utilisé en cryptographie basée sur les codes est le problème de distinguabilité des codes de Goppa (Goppa Code Distinguishing problem (GD)).

Définition 1.2.13 (*Problème de Distinction de Codes de Goppa (GD) [122]*)

Entrée : Soit $\mathcal{H}$ une matrice $(n-k,n)$ binaire (la matrice de partié d'un code de $\mathsf{Goppa}(n,k)$) ou une matrice $(n-k,n)$ binaire aléatoire.
Sortie : $b = 1$ si $\mathcal{H} \in \mathsf{Goppa}(n,k)$, $b = 0$ sinon.

Nous proposons dans le chapitre 7 un autre problème décisionnel appelé *Décodage par Syndrome Quasi-Cyclique*. Nous prouvons que ce nouveau problème est aussi NP- complet.

Définition 1.2.2 (*Décodage par Syndrome Quasi-Cyclique*)

Étant donné $\ell > 1$ (on évite le cas $l = 1$ qui correspond au cas dégénéré) des matrices $A_1, \ldots, A_\ell$ de taille $r^* \times n^*$ sur $\mathbb{F}_q$, un entier $w < \ell n^*$ et un mot $\boldsymbol{z}$ de $\mathbb{F}_q^{\ell r^*}$. Soit A une matrice $\ell r^* \times \ell n^*$ définie de la manière suivante :

$$A = \begin{bmatrix} A_1 & \cdots & \cdots & A_\ell \\ A_\ell & A_1 & \cdots & A_{\ell-1} \\ \vdots & \ddots & \ddots & \vdots \\ A_2 & \cdots & A_\ell & A_1 \end{bmatrix}$$

Existe-t-il $\boldsymbol{e}$ dans $\mathbb{F}_q^{\ell n^*}$ de poids $\mathsf{wt}(\boldsymbol{e}) \leq w$ tel que $A \times \boldsymbol{e}^T = \boldsymbol{z}$?

Chapitre 2

Les codes en cryptographie

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

2.1 Chiffrer avec des codes

2.1.1 Le cryptosystème de McEliece

La sécurité du système repose donc sur deux problèmes distincts (voir chapitre précédent) :
– l'indistingabilité entre un code structuré et un code aléatoire ;
– le problème du décodage d'un code aléatoire.

Le cryptosystème de base

Soit $n = 2^m$, x le message, G la matrice génératrice sous forme systématique d'un code, $x' = xG$ est de la forme $(x|y)$ où y est $(n-k)$-bits. Ce sont ces $n-k$ bits (*redondance*) qui permettent de corriger des erreurs. Il faut m bits pour corriger une erreur, soit, pour corriger t erreurs, mt bits au plus (généralement moins car les erreurs sont interchangeables). Comme on a $2^m - k$ redondances, on peut corriger t_0 erreurs, avec $t_0 \geq (2^m - k)/m$. Par exemple, avec $m = 10$ (donc $n = 1024$) et $k = 512$, on peut corriger 52 erreurs au moins. Un tel code est généré par un polynôme de degré t de $\mathbb{F}_{2^m}$.

Dans le cryptosystème de McEliece, la matrice génératrice G est la clé secrète de Bob. Pour fabriquer une clé publique, on multiplie G à gauche par une matrice binaire non singulière (i.e. inversible) S de dimension (k, k). Cette nouvelle matrice $G'' = SG$ sera donc toujours une matrice génératrice d'un code : le produit de S par G revient, en effet, à prendre une combinaison linéaire des lignes de G. Cependant, il est facile de retrouver G à partir de G''. Il suffit de faire une élimination gaussienne pour obtenir la matrice identité en première partie de G. Pour parer cette attaque, on multiplie à droite G par une matrice de permutation P de dimension (n, n). La clé publique de Bob sera alors :

$$G' = SGP.$$

1. [Génération de clés] Soit G la matrice génératrice sous forme systématique d'un code, on choisit aléatoirement S une matrice binaire inversible de dimension (k, k) et une matrice de permutation P de dimension (n, n). On calcule la clé ublique :
$$G' = SGP.$$
2. [Chiffrement] Bob publie $G' = SGP$. Si Alice veut envoyer le message x (constitué de k bits) à Bob. Alice génère un mot de n bits aléatoires e de poids t, elle calcule et envoie à Bob
$$y = xG' + e.$$
3. [Déchiffrement] Pour $y = xG' + e$, la connaissance des secrets permet :
   (a) de calculer $u = yP^{-1}$, u est alors un mot du code de Goppa de matrice génératrice SG contenant t erreurs ;
   (b) de déterminer u' en corrigeant les erreurs de u ;
   (c) calculer $x = u'S^{-1}$.

FIGURE 2.1 – Le protocole de McEliece

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

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

– la sécurité croît beaucoup plus avec la taille des clés que pour le système RSA ;
– la rapidité du chiffrement.

Un autre avantage est de reposer sur un problème très différent des algorithmes asymétriques usuels. En conséquence de quoi une percée théorique dans le domaine de la factorisation, qui ruinerait RSA, n'affecterait en rien ce cryptosystème.

Le cryptosystème de McEliece résiste à ce jour à toute tentative de cryptanalyse, mais est rarement utilisé en pratique du fait de la grande taille des clés. On peut cependant noter qu'il a été utilisé pour le chiffrement dans Entropy, une alternative à Freenet [1]

2.1.2 La variante de Niederreiter

En 1986, Harald Niederreiter a proposé un autre cryptosystème fondé sur la théorie des codes. Le cryptosystème de Niederreiter a été prouvé équivalent à celui de McEliece en 1994 par Y.X. Li, R.H. Deng et X.M. Wang [75].

1. Entropy est un réseau décentralisé de communication peer-to-peer, conçu pour résister à la censure, tout comme Freenet. Entropy est une base de données écrite en C. Il rassemble les contributions de bande passante et d'espace de stockage de membres pour permettre aux utilisateurs de publier anonymement ou récupérer des informations de toutes sortes. Le nom Entropy est l'acronyme de *Emerging Network To Reduce Orwellian Potency Yield*, en référence à George Orwell auteur de 1984 et de sa police de la pensée totalitaire esclavagisant des personnes par le contrôle de leurs informations.

1. [Génération de clés] Soit $\mathcal{C}$ un code linéaire q-aire t-correcteur de longueur n et de dimension k. Soit H une matrice de parité de $\mathcal{C}$. On choisit aléatoirement S inversible et P une matrice de permutation. On calcule H' telle que :

$$H' = SHP$$

où H' sera publique, et sa décomposition constituera le secret, avec la connaissance d'un algorithme de décodage par syndromes efficace dans $\mathcal{C}$.

2. [Chiffrement] Pour un texte clair x choisi dans l'espace $E_{q,n,t}$ des mots de $\mathbb{F}_q^n$ de poids de Hamming t (l'espace des erreurs) : y est le cryptogramme correspondant à x ssi :

$$y = H'x^T.$$

3. [Déchiffrement] Pour $y = H'x^T$, la connaissance des secrets permet :
   (a) de calculer $S^{-1}y (= HPx^T)$;
   (b) de trouver Px^T à partir de $S^{-1}y$ grâce à l'algorithme de décodage par syndromes de $\mathcal{C}$;
   (c) de trouver x en appliquant P^{-1} à Px^T.

FIGURE 2.2 – Le protocole de Niederreiter

2.2 Signer avec des codes

2.2.1 Le schéma de signature de Courtois, Finiasz et Sendrier

La construction d'un schéma de signature pour les codes correcteurs n'est pas aussi naturelle que pour RSA. En effet, l'opération de décodage n'est pas inversible, c'est-à-dire qu'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. Le schéma que Courtois, Finiasz et Sendrier proposent [39], utilise des codes de Goppa avec une faible capacité de correction t, dans ce cas, la proportion de mots décodables est $\frac{1}{t!}$. Le schéma de signature consiste alors en la concaténation d'un haché du message $h(M)$, une suite d'éléments croissants 0,1,2,. . . et qui calcule la valeur du haché jusqu'à ce que la valeur du haché $h(M|i_0)$ corresponde à un mot décodable. La signature est alors le mot décodé associé à la valeur du haché $h(M|i_0))$. Les paramètres proposés par les auteurs sont $n = 2^{16} - 1$ et $t = 9$.

Soit M un message à signer. Soit h une fonction de hachage à valeurs dans $\{0,1\}^{n-k}$. On cherche un moyen de trouver un $s \in E_{q,n,t}$ tel que $h(M) = H's^T$. Il s'agit donc de déchiffrer $h(M)$. Le principal problème est que $h(M)$ n'est pas *a priori* dans l'espace d'arrivée de $x \to H'x^T$. C'est à dire que $h(M)$ n'est pas *a priori* dans l'espace des chiffrés par le système de Niederreiter. Pour contourner cette difficulté on utilise le protocole proposé par Courtois, Finiasz et Sendrier dans [39].

1. $i \leftarrow 0$
2. tant que $h(M|i)$ n'est pas un syndrome décodable faire $i \leftarrow i + 1$
3. calculer $s = D(h(M|i))$

FIGURE 2.3 – Protocole de Courtois-Finiasz-Sendrier

Soit $D()$ un algorithme de décodage par syndrome :
On obtient en sortie un couple $\{s, j\}$ tel que $h(M|j) = H's^T$. On peut remarquer qu'on a nécessairement s de poids t.

Dans [41], une preuve de sécurité est donnée dans le modèle de l'oracle aléatoire. Nous utiliserons le nouveau schéma que l'auteur décrit, appelé mCFS, pour donner une preuve de sécurité de notre schéma. Nous reviendrons sur cette construction dans le chapitre 5.

2.2.2 Le schéma d'identification de Stern

Le schéma de base

Le schéma de Stern est un schéma interactif à divulgation nulle de connaissance qui a pour protagonistes un *prouveur* noté P et un *vérifieur* noté V. Le prouveur cherchera à s'identifier auprès du vérifieur.

Soient n et k deux entiers tels que $n \geq k$. Le schéma de Stern nécessite une $(n-k) \times n$ matrice publique H' définie sur $\mathbb{F}_2$.

Soit $t \leq n$ un entier. Pour des raisons de sécurité (discutées dans [129]) il est recommandé que t soit choisi juste au dessus de la borne de Gilbert-Varshamov (voir [85]). La matrice H' et le poids t sont des paramètres du protocole et seront utilisés par plusieurs prouveurs différents.

Chaque prouveur P reçoit une clé secrète de n-bits s_P (aussi notée s s'il n'y a pas d'ambiguité sur le prouveur) de poids de Hamming t et calcule un *identifiant public* i_P tel que $i_P = H's_P^T$. Cet identifiant est calculé une fois pour toute pour H' donnée et peut alors être utilisé pour de nombreuses authentifications. Quand un utilisateur P a besoin de prouver à V qu'il est la personne associée à son identifiant public i_P, alors les deux protagonistes suivent le protocole suivant où h est une fonction de hachage standard :

Remarque 1

Lorsque b vaut 1, durant la quatrième étape du schéma de Stern, on peut noter que $H'y^T$ vient directement de la relation $H'(y \oplus s)^T$ car nous avons :

$$H'y^T = H'(y \oplus s)^T \oplus i_P = H'(y \oplus s)^T \oplus H's^T \quad .$$

Comme prouvé dans [129], le protocole est à divulgation nulle de connaissance. Pour un tour d'exécution, la probabilité qu'une personne malhonnête réussisse à tricher est de $2/3$. Afin d'obtenir un degré de sécurité β, le protocole doit être exécuté un nombre de fois k tel que l'on ait la relation $(2/3)^k \leq \beta$. Quand le nombre d'itérations vérifie la dernière condition, alors la sécurité du problème repose sur le problème NP-complet de décodage par syndrome (SD).

1. [Commitment Step] P choisit aléatoirement $y \in \mathbb{F}^n$ et une permutation σ définie sur $\mathbb{F}_2^n$. Alors P envoie à V les engagements c_1, c_2 et c_3 tels que :

$$c_1 = h(\sigma | H' y^T);\ c_2 = h(\sigma(y));\ c_3 = h(\sigma(y \oplus s)).$$

2. [Challenge Step] V envoie $b \in \{0, 1, 2\}$ à P.
3. [Answer Step] Trois possibilités :
   - si $b = 0$: P révèle y et σ.
   - si $b = 1$: P révèle $(y \oplus s)$ et σ.
   - si $b = 2$: P révèle $\sigma(y)$ et $\sigma(s)$.
4. [Verification Step] Trois possibilités :
   - si $b = 0$: V vérifie que c_1, c_2 sont corrects.
   - si $b = 1$: V vérifie quec_1, c_3 sont corrects.
   - si $b = 2$: V vérifie que c_2, c_3 sont corrects, et que le poids de $\sigma(s)$ est bien t.
5. Itère les étapes 1,2,3,4 jusqu'à ce que le niveau de sécurité recherché soit atteint.

FIGURE 2.4 – Le protocole de Stern

La probabilité de triche est obtenue de la façon suivante

- Le tricheur prend un vecteur y et une permutation σ. Il remplace le secret s par un vecteur arbitraire t de poids p et calcule les engagements qu'il transmet au vérifieur. Le tricheur ne réussira que si b est égal à 0 ou à 2, car le vérifieur calculera les engagements à vérifier uniquement à partir des données que le prouveur a fourni. Sa probabilité de triche est donc de $(2/3)^r$. Dans le cas ou $b = 1$, le vérifieur utilisera l'identifiant i pour retrouver $H(y) = H(y \oplus s) \oplus i$, comme le tricheur a fourni $(y \oplus t)$ au lieu de $(y \oplus s)$, le vérifieur ne retrouvera donc pas $H(y)$, et la vérification sera fausse.
- Si le tricheur procède comme précédement mais calcule $c_1 = \langle \sigma, H(y \oplus t) \oplus i \rangle$, il pourra tricher pour $b = 1$ et $b = 2$, mais pas pour $b = 0$, car dans ce cas, le vérifieur calculera $c_1 = \langle \sigma, H(y) \rangle$.
- Le tricheur prépare σ, y et $(y \oplus t)$, avec t tel que $H(t) = i$, mais avec t de poids différent de p. Le tricheur réussira quand $b = 0$ ou $b = 1$, car lorsque $b = 2$, la vérification du poids de $t.\sigma$ échouera. Il y a donc dans ce cas la même probabilité de triche $(2/3)^r$.

Remarque 2

En utilisant l'heuristique de Fiat-Shamir (vu au chapitre 1), il est possible théoriquement de convertir le protocole d'identification de Stern en un protocole de signature, mais la signature obtenue est très longue (quelque 150-kbit pour une sécurité de 2^{80} opérations binaires). Remarquons que cela est conséquent par rapport aux schémas de signature classique, mais cette taille reste raisonnable compte tenu des capacités de mémoire actuelles. De plus, nous présentons dans la section 3.5 page 66 une méthode qui permet de réaliser cette signature sur carte à puce sans avoir besoin de stocker la liste de tous les engagements. Ceci coûte un temps plus long d'exécution mais permet d'utiliser le protocole d'identification en signature sans ajout de mémoire.

Le schéma de Stern possède de nombreux avantages. Le premier de tous, sa sécurité repose sur un problème qui n'est pas un des deux problèmes utilisés traditionnellement en cryptographie, à

savoir la factorisation et le problème du logarithme discret. Ceci donne une bonne alternative aux nombreux cryptosystèmes dont la sécurité repose sur les deux problèmes précédents, spécialement dans l'éventualité ou un ordinateur quantique puisse exister. Le second avantage du schéma de Stern est qu'il ne nécessite que des opérations très simples (tel que : x-or ou décalage binaire) et ne nécessite aucun crypto-processeur contrairement au cas d'une identification basée sur des problèmes d'arithmétique.

En dépit de ces avantages, le schéma de Stern n'a que rarement été utilisé depuis sa publication en 1993. En effet, le schéma présente les deux inconvénients suivants, qui à eux deux rendent le schéma impraticable dans de nombreuses applications :

1. de nombreux tours sont nécessaires (typiquement 28 si nous voulons que le tricheur ait une probabilité de succès inférieur à 2^{-16}),
2. la clé publique H' est très grande (typiquement 150-kbit).

Le premier point est inhérent au protocole interactif et ne constitue pas un réel inconvénient. Par exemple, si le prouveur et le vérifieur peuvent être connectés ensemble pendant une longue période, alors l'identification peut être effectuée graduellement. Dans ce cas, le processus d'identification complet est réalisé en exécutant, de temps en temps durant une période prescrite (*i.e.* une heure), une itération de l'algorithme 2.2.2 page 23 jusqu'à ce que le niveau de sécurité espéré soit atteint. De tels schémas d'authentification *graduels* peuvent être de grand intérêt dans la télévision à péage ou dans les systèmes où une machine (*i.t.* une photocopieuse ou une machine à café) veut identifier un matériel physique (*i.t.* une cartouche d'encre ou de café).

Le deuxième inconvénient a été récemment traité par Gaborit et Girault dans [50]. Dans leur article, les auteurs proposent d'utiliser des *matrices doublement circulantes* pour décroître la taille de la clé publique sans apparemment dégrader le niveau de sécurité du protocole. Nous décrivons cette idée dans la section suivante :

Construction quasi-cyclique

L'idée de [50] est de remplacer la matrice aléatoire H' par la matrice de parité d'un type particulier de codes aléatoires : les *codes doublement circulants*.

Cette idée vient du fait que dans le cryptosystème NTRU [2] la clé publique consiste en le quotient $\frac{f}{g}$ où f et g sont des polynômes tronqués de l'anneau quotient $R = \mathbb{Z}_q[x]/(x^n - 1)$ pour n et q des paramètres du système (typiquement 251 et 128). Pour chiffrer, nous devons multiplier ces deux polynômes dans l'anneau R. Ce produit peut être vu comme la multiplication de deux matrices circulantes.

Soit ℓ un entier. Une matrice aléatoire doublement circulante $\ell \times 2\ell$ H est une matrice de la forme :

$$H = (I|A) \ ,$$

où A est une *matrice circulante*, qui est de la forme suivante :

2. NTRU= Number Theorists aRe Us ou encore N-th degree TRUncated polynomial ring. Le schéma de chiffrement a été proposé en 1996, le schéma de signature en 2002 mais partiellement cassé par Nguyen et Regev en 2006 (voir www.ntru.com)

$$A = \begin{pmatrix} a_1 & a_2 & a_3 & \cdots & a_\ell \\ a_l & a_1 & a_2 & \cdots & a_{\ell-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_2 & a_3 & a_4 & \cdots & a_1 \end{pmatrix} ,$$

où $(a_1, a_2, a_3, \cdots, a_\ell)$ est un vecteur de $\mathbb{F}_2^\ell$.

Comme nous pouvons facilement le vérifier, représenter H ne nécessite pas de stocker tous les coefficients de la matrice (comme c'est le cas dans le schéma de Stern originel) mais nécessite seulement le vecteur de ℓ-bits $(a_1, a_2, a_3, \cdots, a_\ell)$ (qui constitue la première ligne de A). Soit n égal à 2ℓ, les paramètres du nouveau schéma ont les tailles suivantes :

- **Données privées :** le secret s de n-bits.
- **Données publiques :** le syndrome public i_P de taille $\frac{n}{2}$ et la première ligne de A de taille $\frac{n}{2}$, soit n bits.

Comme expliqué dans [50] le comportement de ces matrices en terme de distance minimale est le même que celui des matrices aléatoires. De même, il semble que le problème du décodage par syndrome reste difficile dans le cas particulier des codes doublement circulants (voir [50] pour plus de détails).

Comme la nouvelle version du schéma de Stern nécessite des paramètres de plus petite taille et n'utilise toujours que des opérations élémentaires. Cette version devient très attractive pour des implémentations. Ceci est spécialement vrai dans des environnements à mémoire (RAM,PROM,etc.) limitée.

Dans le but de résister aux attaques par décodage, les auteurs de [50] préconisent des paramètres d'au moins $\ell = 347$ et $t = 74$. Ces choix mènent en effet à un niveau de sécurité d'au moins 2^{85} opérations binaires avec une clé (publique et secrète) de 694 bits.

Remarque 3

Comme dans l'article originel proposé par Stern, tous les utilisateurs peuvent utiliser la même matrice H, et ce sont seulement les paramètres secrets x et le syndrome s qui sont individuels. Dans certaines applications, H peut aussi être considérée comme un paramètre secret.

2.2.3 Le schéma de signature de Kabatianskii, Krouk et Smeets

Kabatianskii Krouk et Smeets ont mis au point un schéma que nous détaillerons dans la section 4 page 71.

2.3 Les attaques : cryptanalyse du schéma de McEliece

Dans cette section inspirée de [95], nous décrivons les attaques connues du cryptosystème de McEliece (voir McEliece [86], Kobara et Imai [58, 59], Adams et Meijer [1], Rao et Nam [115], Park [100], Gibbon [51], Korzhik et Turkin [70], van Tilborg [133], Lee et Brickell [73], Simmons [127], Beth et al. [17], Loidreau et Sendrier [78], Berson [16], Sidelnikov and Shestakov [126], et Engelbert, Overbeck et Schmidt [44]). La classification est de Kobara et Imai [58].

Bien qu'aucun algorithme efficace pour décomposer G' en (S, G, P) n'ait été découvert [91], une attaque structurelle a été découverte dans [78]. Cette attaque révèle une part de la structure de G' qui est généré à partir d'un code de Goppa binaire. Cependant, cette attaque peut être évitée simplement en n'utilisant pas de clé publique *faible*.

Le cas suivant que nous devons considérer est qu'un code de Goppa équivalent à G' (qui n'est pas nécessairement G), dont l'algorithme de décodage est connu, soit trouvé. Cette probabilité qui est estimée dans [1] et [51], est très petite. Toutes les autres attaques connues sont utilisées pour déchiffrer le message chiffré sans casser la clé publique. Nous classons ces attaques en deux grandes catégories : attaques critiques et attaques non-critiques.

Les attaques non-critiques sont celle qui dépendent fortement des paramètres et peuvent alors être contrecarrées seulement en augmentant la valeur de ces paramètres.

Les attaques critiques sont plus rapides que les attaques non-critiques et réalisables pour des paramètres réalistes. Elles peuvent seulement être évitées en utilisant des conversions adaptées (voir Kobara et Imai [58]) et en évitant les codes ayant une faiblesse structurelle.

De manière intéressante, toutes les attaques critiques exploitent des faiblesses structurelles des codes utilisés ou nécessitent des informations additionnelles, comme par exemple une connaissance partielle des textes en clair. Sans ces informations additionnelles, aucun algorithme efficace n'est connu pour déchiffrer un texte chiffré arbitraire à l'aide du cryptosystème de McEliece.

2.3.1 Attaques non-critiques

Les deux attaques suivantes peuvent être rendues infaisables simplement en augmentant la taille des paramètres. Kobara et Imai suggèrent dans [58] d'appliquer les modification de P. Loidreau [77] pour accroître la sécurité. Dans [59], ils montrent que cette modification, bien qu'augmentant la sécurité contre les attaques à clairs connus (CPA), est vulnérable à de nouvelles attaques de type CPA qu'ils ont développées. Il est alors intéressant d'utiliser ces modifications dans le cas du cryptosystème de McEliece.

La cryptanalyse de McEliece

McEliece proposa aussi une attaque contre son protocole. Elle consiste à choisir aléatoirement k bits du mot reçu x en espérant qu'il n'y ait pas d'erreurs sur ces bits là. Soient x_k le vecteur uniquement formé par les k bits choisis et G'_k la matrice G' dont on ne garde que les k colonnes correspondants aux k bits choisis. S'il n'y a effectivement pas d'erreurs dans les k bits choisis, alors :

$$x_k G'^{-1}_k = u.$$

L'inversion de la matrice G'_k nécessite $\mathcal{O}(k^3)$ opérations. La probabilité de n'avoir aucune erreur en choisissant aléatoirement k bits est :

$$T = \frac{\binom{n-t}{k}}{\binom{n}{k}}.$$

Le coût de l'algorithme est donc k^3/T.

En 1988, Lee et Brickell améliorèrent cette attaque. Comme G' est la matrice génératrice d'un code de distance minimale supérieure à $2t$, si $x_k G_k^{'-1}$ n'est pas le véritable message u, alors $uG' + x_k G_k^{'-1} G'$ est de poids au plus $2t$ (c'est la différence entre le vrai message et le message trouvé par le cryptanalyste). De là, si $x + x_k G_k^{'-1} G'$ est de poids inférieur ou égal à t, alors on peut dire que $x_k G_k^{'-1} = u$.

On peut généraliser cet algorithme de la manière suivante en permettant d'avoir quelques erreurs dans le vecteur x_k :
Soit j un paramètre dont la valeur sera discutée par la suite :

1. Soit x_k un k-bit choisis aléatoirement parmi les bits du message reçu x. Soit G'_k la matrice formée par les colonnes de G' correspondant aux k bits pris de x. On calcule $G_k^{'-1} G'$ et $x + x_k G_k^{'-1} G'$.
2. On choisit un k-bit e_k de poids inférieur ou égal à j (e_k modélisera l'erreur). Si $(x + x_k G_k^{'-1} G') + e_k(G_k^{'-1} G')$ est de poids inférieur ou égal à t, on s'arrête : $u = +x_k G_k^{'-1}$.
3. Si l'on a utilisé tous les vecteurs possibles e_k tels que $|e_k| \leq j$, on recommence à l'étape 1, sinon à l'étape 2.

Soit Q_i la probabilité qu'il y ait exactement i erreurs dans x_k,

$$Q_i = \binom{t}{i}\binom{n-t}{k-i} / \binom{n}{k}.$$

On doit exécuter T_j fois l'étape 1 avec :

$$T_j = 1 / \sum_{i=0}^{j} Q_i.$$

Soit N_j le nombre de vecteurs e_k distincts, de poids inférieur ou égal à j :

$$N_j = \sum_{i=0}^{j} \binom{k}{i}.$$

Étant donné que l'étape 1 nécessite à peu près ak^3 opérations, avec a petit, et l'étape 2 nécessite bk opérations avec b petit, le coût de cet algorithme est de l'ordre de :

$$T_j(ak^3 + N_j bk)$$

On choisit alors j minimisant ce coût. Dans la plupart des cas, avec des dimensions raisonnables, il faudra prendre $j = 2$. Par exemple, si l'on prend $n = 1024, k = 644, t = 38$, l'attaque nécessite $2^{73,4}$ opérations élémentaires.

Attaque par ensemble d'informations

De toutes les attaques générales (i.e., pas sur des codes spécifiques) cette attaque semble être celle qui possède la plus petite complexité. On essaie de retrouver les k symboles d'informations de la manière suivante :

1. La première étape consiste à choisir k des n coordonnées aléatoirement en espérant avoir sélectionné toutes les t positions d'erreurs.
2. On essaie alors de retrouver le message en résolvant le système linéaire de taille $k \times k$ (binaire sur $\mathbb{F}_q$).

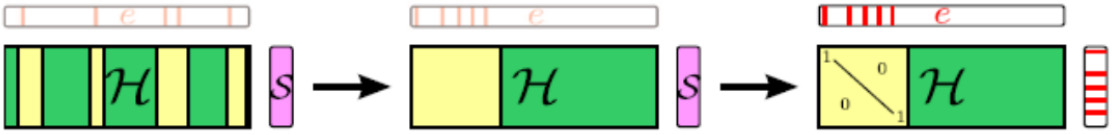


FIGURE 2.5 – Décodage par ensemble d'informations

Soit G'_k, c_k et z_k représentant les k colonnes prisent dans G', c et z, respectivement. Elles vérifient la relation suivante :

$$c_k = mG'_k + z_k.$$

Si $z_k = 0$ et G'_k est non singulier, m peut être retrouvé par

$$m = c_k {G'}_k^{-1}.$$

Le coût de calcul de cette version (appelée *attaque par ensemble d'informations originale*) est $T(k) \times P_{n,k,t}$, où

$$P_{n,k,t} = \Pi_{i=0}^{k-1} \left(1 - \frac{t}{n-i}\right).$$

La quantité $T(k)$ est le coût moyen d'opérations nécessaires pour résoudre un système linéaire $k \times k$ sur $\mathbb{F}_q$. Comme mentionné dans [86], la résolution d'un système binaire de taille $k \times k$ nécessite environ k^3 opérations. Sur $\mathbb{F}_q$, cela nécessite au moins $(k \times \log_2 q)^3$ opérations.

De plus, si $z_k \neq 0, m$ peut être retrouvé en devinant z_k positions parmis les éléments de petits poids de Hamming [73, 33]. On appelera cette attaque *attaque par ensemble d'informations généralisée* (*generalized information-set-decoding (GISD) attack*). Une itération de l'algorithme s'effectue de la manière suivante :

1. Permutation les colonnes de la matrice génératrice de manière aléatoire.
2. Faire une élimination gaussienne des colonnes de la matrice pour obtenir la forme $G = (I_k|A)$, avec les permutés correspondants du chiffré $c = (c_1 + e_1|c_2 + e_2)$.
3. Deviner que l'erreur e_1 est de poids au plus p et vérifier que l'erreur $e = (e_1|e_2)$ est de poids t.

La probabilité π qu'une permutation des colonnes mène à une configuration favorable est

$$\pi(p, n, k, t) = \sum_{i=0}^{p} \frac{\binom{n-t}{k-i}\binom{t}{i}}{\binom{n}{k}}.$$

Pour chaque itération, une estimation du nombre d'opérations est :

1. $\frac{k^2 n}{2}$ pour l'élimination gaussienne ;
2. environ $\frac{k}{2} + \sum_{i=1}^{p} \binom{k}{i} i$ additions sur les mots de $(n-k)$-bits de A.
   Alors, une estimation du coût de cet algorithme est

$$W(p,n,k,t) = \frac{\frac{k^2 n}{2} + (n-k)[\frac{k}{2} + \sum_{i=1}^{p} \binom{k}{i} i]}{\pi(p,n,k,t)}.$$

Actuellement la version donnée par A. Canteaut et F. Chabaud dans [23] est la plus aboutie dans ce cadre. La version généralisée de cet algorithme et son coût en nombre de calculs ont été étudiés par F. Chabaud [33], par exemple. Cet algorithme est sensiblement plus rapide que la version originale (où z_k est supposé être 0), mais reste inapplicable pour des paramètres appropriés (voir [58]).

Recherche de mot de petits poids

Cette attaque utilise un algorithme qui trouve un mot de petit poids parmi les mots de code générés par une matrice génératrice arbitraire en utilisant une base de données obtenue par pré-calcul [128, 23]. Comme le mot de code de poids minimum de la matrice génératrice $(k+1) \times n$ suivante :

$$[G'|c]^t$$

est le vecteur erreur z de c où $c = mG' + z$, cet algorithme peut être utilisé pour retrouver m à partir d'un message chiffré c.

Le coût précis de cette attaque a été évalué par A. Canteaut et F. Chabaud dans [23]. Il est montré impossible d'inverser c (retrouver m) pour des paramètres appropriés, i.e. $n \geq 2048$ et des paramètres k et t optimisés. Même en utilisant les paramètres originaux $(n,k,t) = (1024, 524, 50)$ suggérés dans [86], cette attaque a un coût de l'ordre de $2^{64.2}$ opérations binaires.

2.3.2 Attaques critiques

Les attaques suivantes ne peuvent pas être évitées en augmentant la taille des paramètres. Elles utilisent des faiblesses structurelles ou nécessitent de l'information supplémentaire.

Attaque à texte clair partiel

Avoir une connaissance partielle du texte clair réduit de manière drastique le coût de calculs des attaques contre le cryptosystème de McEliece [23, 57]. Par exemple, soient m_ℓ et m_r représentant les k_ℓ bits de gauche et les k_r bits restant du message clair m, i.e. $k = k_l + k_r$ et $m = (m_l|m_r)$. Supposons qu'un adversaire connaisse m_r. Alors, la difficulté de retrouver le message clair inconnu m_ℓ dans le cryptosystème de McEliece PKC avec pour paramètres (n,k) est équivalent au fait de retrouver le message clair en entier dans le cas d'un chiffrement à l'aide de McEliece de paramètres (n,k_l), puisque

$$\begin{aligned} c &= mG' + z \\ c &= m_l G'_l + m_r G'_r + z \\ c + m_r G'_r &= m_l G'_l + z \\ c' &= m_l G'_l + z \end{aligned}$$

où ${G'}_\ell$ et ${G'}_r$ sont les k_ℓ lignes supérieures et k_r les autres lignes de G', respectivement.

Attaque par renvoi de message

Supposons maintenant que, à cause d'un accident, ou du fait de l'action du cryptanalyste, à la fois

$$c_1 = mG' + e_1$$

et

$$c_2 = mG' + e_2,$$

$e_1 \neq e_2$, sont envoyés. Ceci est appelé un *renvoi de message*. Dans ce cas, il est facile pour le cryptanalyste de retrouver m à partir du système des c_i. Nous examinerons seulement le cas où $i = 2$. L'attaque est encore plus facile si $i > 2$. Remarquons que $c_1 + c_2 = e_1 + e_2 (\mathrm{mod} 2)$.

Un renvoi de message peut être facilement détecté en observant le poids de Hamming de la somme de deux cryptogrammes. Quand les messages sont différents, le poids attendu de la somme est d'environ 512 (pour les paramètres originaux de McEliece, en général le poids attendu est k). Alors que quand les deux messages sont identiques, le poids de la somme ne peut excéder 100 (ou en général $2t$). Heiman [56] a montré que le renvoi de message peut être détecté. Nous verrons comment l'exploiter.

Dans la suite, nous utiliserons les paramètres originaux proposés par McEliece $(n, k, t) = (1024, 524, 50)$. Les résultats pour d'autres paramètres sont essentiellement les mêmes. Tout d'abord nous calculons deux ensembles à partir de $(c_1 + c_2)$. On définit L_0 et L_1 de la manière suivante :

$$L_0 = \{l \in \{1..n\} : c_1(\ell) + c_2(\ell) = e_1(\ell) + e_2(\ell) = 0\}$$

$$L_1 = \{l \in \{1..n\} : c_1(\ell) + c_2(\ell) = e_1(\ell) + e_2(\ell) = 1\}$$

Nous voulons tirer un avantage du fait que

- $\ell \in L_0$ est plus probable si ni $c_1(\ell)$ ni $c_2(\ell)$ ne sont altérés que par une erreur, alors que
- $\ell \in L_1$ implique qu'exactement un de $c_1(\ell)$ ou $c_2(\ell)$ est altéré par une erreur.

Supposons que les vecteurs d'erreurs e_1 et e_2 sont choisis indépendamment, alors pour n'importe quel $\ell \in \{1..n\}$, la probabilité que les deux vecteurs erreurs soient à 1 sur la position d'indice ℓ est

$$P(e_1(\ell) = e_2(\ell) = 1) = \left(\frac{50}{1024}\right)^2 \approx 0.0024.$$

En d'autres termes, pour la plupart des $\ell \in L_0$ on a $e_1(\ell) = e_2(\ell) = 0$. Le cryptanalyste peut essayer de deviner les 524 colonnes non perturbées de celles indexées par L_0.

Quelle est la pertinence de cette stratégie ?

Soit p_i la probabilité que précisément i coordonnées sont perturbées simultanément par e_1 et e_2. Alors

$$p_i = P(|\{\ell : e_1(\ell) = 1\} \cap \{\ell : e_2(\ell) = 1\}| = i) = \frac{\binom{50}{i}\binom{974}{50-i}}{\binom{1024}{50}},$$

alors, e_2 doit choisir i positions d'erreurs parmi les 50 perturbées par e_1 et les $50 - i$ restantes parmi celles inchangées par e_1. De plus, le nombre attendu de L_1 est

$$E(|L_1|) = \sum_{i=0}^{50} (100 - 2i) p_i \approx 95.1,$$

comme tout ℓ pour lequel $e_1(\ell) = e_2(\ell) = 1$ divise $|L_1|$ par deux.

Par exemple, supposons que l'on ait $|L_1| = 94$. Alors $|L_0| = 930$, où seulement 3 positions sont perturbées. Nous observons que la probabilité de deviner les 524 colonnes non perturbées parmi celles indexées par L_0 est

$$\frac{\binom{927}{524}}{\binom{930}{524}} \approx 0.0828,$$

le cryptanalyste peut alors espérer un succès dans ce cas avec seulement 12 suppositions. Ces résultats sont meilleurs d'un facteur de 10^{15} que ceux obtenus en devinant k colonnes non perturbées sans renvoi de message.

Attaque par messages reliés

Nous allons maintenant généraliser l'attaque par renvoi de message. Nous supposons que nous avons deux cryptogrammes :

$$c_1 = m_1 G' + e_1$$

et

$$c_2 = m_2 G' + e_2,$$

où $e_1 \neq e_2$. Les messages m_1 et m_2 peuvent être différents, mais nous supposons que le cryptanalyste connaît une relation linéaire, par exemple, $m_1 + m_2$, entre ces deux messages. Ce que nous appellerons la condition de *messages reliés*. Dans ce cas, l'attaquant peut retrouver les m_i à partir de l'ensemble des c_i en en chiffrant un et en utilisant la méthode précédente. En combinant les deux cryptogrammes précédents, nous obtenons :

$$c_1 + c_2 = m_1 G' + m_2 G' + e_1 + e_2.$$

Remarquons que $m_1 G' + m_2 G' = (m_1 + m_2) G'$, une valeur que le cryptanalyste peut calculer sous la condition de messages reliés à partir de la relation connue et la clé publique. Il résout alors :

$$c_1 + c_2 + (m_1 + m_2) G' = e_1 + e_2$$

et réalise une attaque par renvoi de messages, en utilisant $(c_1 + c_2 + (m_1 + m_2)G')$ au lieu de $(c_1 + c_2)$.

Remarque

L'attaque par renvoi de messages est un cas particulier de l'attaque par messages reliés où $m_1 + m_2 = 0$.

Nous pouvons nous prémunir de ce genre d'attaques en utilisant des fonctions de hachage.

Hormis, ces attaques générales, il existe d'autres types d'attaques contre le cryptosystème de McEliece qui utilise des codes spécifiques. Parmi elles, nous montrerons deux attaques importantes, une contre les codes de Reed-Solomon Généralisés (GRS) et une contre les codes de Goppa générés par un polynôme générateur binaire. Nous verrons ces attaques là plus en détail dans le chapitre 7 page 121.

L'efficacité des attaques sur le cryptosystème de McEliece est essentiellement basée sur le fait que pour un succès il est nécessaire de pouvoir corriger les erreurs de poids inférieur ou égal à

t. L'attaque devient beaucoup plus difficile, si le décodage nécessite une capacité de correction totale, c'est à dire la capacité de corriger tous les mots du code. De manière générale, le décodage total n'est pas une fonction trappe mais une fonction *one-way* car un tel décodage est un problème NP-complet. Le schéma de signature de KKS utilise cette idée voir chapitre 4 page 71.

Deuxième partie

Signatures avec des codes

Chapitre 3

Implémentation sûre du schéma de Stern

Ce chapitre a donné lieu à l'article suivant [31] :

Secure Implementation of the Stern Authentication and Signature Schemes for Low-Resource Devices
avec Philippe Gaborit et Emmanuel Prouff
Eighth Smart Card Research and Advanced Application Conference CARDIS 2008
In G. Grimaud and F.-X. Standaert, editors, Lecture Notes in Computer Science,
Vol. 5189, pages 191-205.

Dans ce chapitre, nous décrivons la première implémentation sur carte à puce d'un protocole d'authentification basé sur les codes correcteurs d'erreurs, le schéma de Stern section 2.2.2 page 22 et nous donnons une sécurisation de ce schéma contre les attaques par canaux cachés. Nous obtenons alors une implémentation sûre d'un schéma d'authentification (et de signature) très pratique et très intéressante d'un point de vue de cryptographie à bas coût.

3.1 Introduction

L'idée d'utiliser les codes correcteurs d'erreurs pour l'authentification est due à Harari, suivi par Stern (premier protocole) et Girault. Les protocoles d'Harari et Girault ont été successivement cassés, pendant que le schéma de Stern avait 5 passes et était impraticable. Le premier protocole pratique et sûr basé sur les codes correcteurs d'erreurs a été proposé à Crypto'93 par Stern [129]. Le schéma d'authentification à divulgation nulle de connaissance proposé par Stern est basé sur un problème de théorie des codes : le problème de décodage par syndrome, *Syndrome Decoding problem* (*SD*). Il s'agit d'un protocole à la Fiat-Shamir mais avec une probabilité de triche de $2/3$ au lieu des $1/2$ de Fiat-Shamir. Il est considéré comme une bonne alternative aux nombreux schémas dont la sécurité est liée à des problèmes de théorie des nombres classique tels que la factorisation ou le problème du logarithme discret.

Bien que le schéma de Stern ait été proposé il y a 15 ans, il n'a (à ma connaissance) jamais été implémenté sur carte à puce. Il souffre d'un inconvénient classique due à l'utilisation de codes correcteurs d'erreurs : la taille de la clé publique est grande. En effet, comme le prouveur et le vérifieur doivent connaître une grande matrice aléatoire d'au moins 100-kbits, il est difficile d'utiliser le schéma sur des supports à ressources limitées tels que les cartes à puce ou les tags RFID.

Cette faiblesse a été récemment résolue par Gaborit et Girault dans [50] (voir aussi section 2.2.2 page 24) où ils proposent d'utiliser des matrices de parité de codes aléatoires quasi-cycliques au lieu de matrices purement aléatoires. Cette solution permet de préserver la sécurité du schéma et décroît la description de la matrice aléatoire à seulement quelques centaines de bits. Cette nouvelle avancée ouvre les portes d'une utilisation du schéma de Stern dans un milieu à ressources limitées.

Contribution Dans ce chapitre, nous donnons pour la première fois une description précise de l'implémentation du protocole de Stern (qui est différent de celle d'un schéma classique basé sur la théorie des nombres) et nous montrons comment protéger les principales étapes de l'algorithme contre les attaques par canaux cachés. On obtient pour notre implémentation une authentification en 5.5 secondes et une signature en 22 secondes, les deux sans crypto-processeur. Il s'agit d'un résultat prometteur quand on le compare à une implémentation de RSA qui prend plus de 30 secondes dans un contexte similaire sans crypto-processeur. Le protocole de Stern a une application naturelle dans des contextes où le temps d'exécution n'est pas nécessairement court tels que : l'authentification pour la pay-TV ou l'authentification pour contrefaire des moyens expensifs (*i.e.* des cartouches d'encre pour des machines à photocopier). Au delà de ça, le protocole a aussi les quatre avantages suivants :

- il peut être une alternative aux protocoles basés sur la théorie des nombres dans le cas où un ordinateur quantique pourrait exister ;
- comme les opérations utilisées sont linéaires, le protocole est plus facile à protéger contre des attaques par canaux cachés que les protocoles basés sur la théorie des nombres ;
- les opérations linéaires (produits scalaires et permutations de bits) sont facilement implémentables en hardware et sont très efficaces dans ce contexte ;
- la clé secrète est plus petite que celle des autres protocoles (quelques centaines de bits) pour le même niveau de sécurité.

Organisation du chapitre Ce chapitre est organisé de la manière suivante. Dans la section 3.2, nous décrivons le schéma d'identification et de signature de Stern et nous précisons les quatres

étapes principales du protocole. Dans la section 3.3, nous présentons les attaques par canaux cachés dans le contexte de la théorie des codes et dans la section 3.4 nous proposons une version sécurisée de notre implémentation contre les attaques par canaux cachés. Dans la section 3.5, nous donnons les résultats de notre implémentation du schéma et concluons.

3.2 Schéma d'authentification de Stern

Nous reprenons brièvement dans cette section le chapitre 2 page 19 et en particulier la section 2.2.2 page 22. Nous rappellerons ici simplement le schéma (figure 3.1 page 37).

Le schéma de Stern est un schéma interactif à divulgation nulle de connaissance qui a pour protagonistes un utilisateur (appelé le *prouveur* noté P) qui s'identifie auprès d'un autre (appelé le *vérifieur* noté V).

Soient n et k deux entiers tels que $n \geq k$. Le schéma de Stern utilise une $(n-k) \times n$ matrice publique H' définie sur $\mathbb{F}_2$.

Soit $t \leq n$ un entier. Pour des raisons de sécurité (discutées dans [129]) il est recommandé que t soit choisi juste au dessus de la borne de Gilbert-Varshamov (voir [85]). La matrice H' et le poids t sont des paramètres du protocole et seront utilisés par plusieurs prouveurs différents.

Chaque prouveur P reçoit une clé secrète de n-bits s_P (aussi notée s s'il n'y a pas d'ambiguité sur le prouveur) de poids de Hamming t et calcule un *identifiant public* i_P tel que $i_P = H's_P^T$. Cet identificant est calculé une fois pour toute pour H' donnée et peut alors être utilisé pour de nombreuses authentifications. Quand un utilisateur P a besoin de prouver à V qu'il est la personne associée à son identifiant public i_P, alors les deux protagonistes suivent le protocole suivant où h est une fonction de hachage standard :

1. [Commitment Step] P choisit aléatoirement $y \in \mathbb{F}^n$ et une permutation σ définie sur $\mathbb{F}_2^n$. Alors P envoie à V les commitments c_1, c_2 et c_3 tels que :
$$c_1 = h(\sigma | H'y^T);\ c_2 = h(\sigma(y));\ c_3 = h(\sigma(y \oplus s)),$$
2. [Challenge Step] V envoie $b \in \{0,1,2\}$ à P.
3. [Answer Step] Trois possibilités :
   - si $b = 0$: P révèle y et σ.
   - si $b = 1$: P révèle $(y \oplus s)$ et σ.
   - si $b = 2$: P révèle $\sigma(y)$ et $\sigma(s)$.
4. [Verification Step] Trois possibilités :
   - si $b = 0$: V vérifie que c_1, c_2 sont corrects.
   - si $b = 1$: V vérifie que c_1, c_3 sont corrects.
   - si $b = 2$: V vérifie que c_2, c_3 sont corrects, et que le poids de $\sigma(s)$ est bien t.
5. Itérer les étapes 1,2,3,4 jusqu'à ce que le niveau de sécurité recherché soit atteint.

FIGURE 3.1 – Le protocole de Stern

Remarque 4

Durant la quatrième étape, lorsque b vaut 1, on peut noter que $H'y^T$ vient directement de la

relation $H'(y \oplus s)^T$ car nous avons :

$$H'y^T = H'(y \oplus s)^T \oplus i_P = H'(y \oplus s)^T \oplus H's^T \quad .$$

Opérateurs principaux

Une analyse rapide du protocole de Stern montre que les différentes étapes sont composées essentiellement de quatre opérateurs principaux :

1. Produit matrice-vecteur : la multiplication d'un vecteur par une matrice aléatoire ;
2. Fonction de hachage : l'action d'une fonction de hachage sur des données sensibles ;
3. Permutation : la génération et l'action d'une permutation aléatoire ;
4. PRNG : un générateur pseudo-aléatoire utilisé pour générer des permutations aléatoires et des vecteurs aléatoires.

Dans les deux sections suivantes, nous discutons de la problématique des attaques par canaux cachés. Nous précisons pour chaque opérateur (donné ci dessus) un moyen de l'implémenter et de le protéger contre les attaques par canaux cachés.

3.3 Attaques par canaux cachés

Un canal caché est un canal de communication qui utilise la bande passante d'un autre canal dans l'objectif de transmettre des informations sans l'autorisation ou la connaissance du propriétaire de l'information ou de l'administrateur du réseau. Les attaques par canaux cachés ont pour but de retrouver de l'information concernant les *variables sensibles* apparaissant dans la description de l'algorithme attaqué. Nous pouvons dire qu'une variable est sensible si elle est fonction de la clé publique et d'un paramètre secret (ou privé) de l'algorithme (la fonction ne doit pas être constante par rapport au paramètre secret).

Dans la figure 3.1 page 37, seulement les étapes 1 et 3 (celles reliées au prouveur) sont implémentées dans le système embarqué. Les autres étapes, 2 et 4 (qui correspondent au vérifieur) peuvent être effectuées sur un PC. Après un bref listing des variables utilisées aux étapes 1 et 3, nous obtenons la liste suivante de variables sensibles qui peuvent être potentiellement attaquées par SCA :

- Menace A. le vecteur aléatoire y durant les calculs de c_1 (dans le calcul de $\tilde{H}y^T$) et de c_2 (dans le calcul de $\sigma(y)$) : si un attaquant est capable de retrouver y durant une de ces attaques, alors avec une probabilité de $1/3$ il est capable de retrouver s lorsque A répond $y \oplus s$ au challenge $(b = 1)$.
- Menace B. le vecteur privé s durant le calcul de $\sigma(s)$: dans ce cas, l'attaquant retrouve le paramètre privé de A.
- Menace C. le vecteur privé s durant le calcul de $\sigma(y \oplus s)$: dans ce cas, l'attaquant retrouve le paramètre privé de A.
- Menace D. la permutation σ durant le calcul de $\sigma(s)$ ou $\sigma(y \oplus s)$: si un attaquant est capable de retrouver σ durant une de ces étapes, alors avec probabilité $1/3$ il est capable de retrouver s lorsque A répond $\sigma(s)$ au challenge $(b = 2)$.

- Menace E. la permutation σ durant le calcul du haché $h(\sigma|\tilde{H}y^T)$: si un attaquant est capable de retrouver σ, alors avec probabilité $1/3$ il est capable de retrouver s lorsque A répond $\sigma(s)$ au challenge $(b = 2)$.

Trois catégories d'attaques par canaux cachés peuvent être définies, elles dépendent du mode opératoire :

1. Les attaques *template* consistent en une longue étape hors ligne, qui autorise des attaques en ligne futures rapides.
2. Les attaques par *simple analyse* (SPA) sont des attaques en ligne qui consistent en l'interprétation directe de la consommation d'énergie et en l'identification de la séquence exécutée.
3. Les attaques par *corrélation* fonctionnent comme des algorithmes *greedy* : l'information est analysée jusqu'à ce que le secret soit extrait.

Nous allons détailler ces différents types d'attaques dans la section suivante.

Les attaques template

Elles correspondent à un adversaire très puissant. Elles requièrent la possession d'un clone de l'entité attaquée (ou l'entité elle-même). Ce clône est utilisé comme un accessoire d'entraînement, dans le but de construire une base de données du canal caché. Une fois que cette base de données est construite, l'attaque en ligne consiste en le traitement des informations récupérées sur l'accessoire d'entraînement concernant le canal caché dans la base de données et à les comparer à celles sur le matériel attaqué. La clé correcte se distingue alors des mauvaises, en faisant une analyse des variations du profil de la base de données. L'attaque est alors la comparaison des mesures effectuées. Dans une telle attaque, l'attaquant n'a pas besoin d'un modèle pour estimer la relation entre la consommation d'énergie du matériel et la valeur de la donnée manipulée.

Les attaques SPA et par corrélation

Cette partie est inspirée de l'article de Nadia El Mrabet [43].

Ces attaques fonctionnent différemment et concernent un adversaire plus faible que dans le cas des attaques template. Un problème physique connu ou suspecté est analysé dans le cadre d'acquisition d'information par canal caché. L'attaque peut alors commencer.

Pour les attaques de type SPA, les mesures sont analysées et éventuellement moyennées dans le but de réduire les effets du bruit. De cette analyse, (c'est-à-dire des différences temporelles) l'attaquant essaie de déduire des informations concernant les valeurs sensibles. Une telle attaque peut être très efficace contre des implémentations de RSA non protégées (voir [68]).

Attaque SPA : l'exemple de RSA Le protocole RSA du nom de ses inventeurs Rivest, Shamir et Adleman est un protocole asymétrique à clé publique. Il est utilisé à tous les niveaux de la société. Par le gouvernement et bien sûr par les armées, le système bancaire, les cartes à puce, sur internet pour le commerce et l'échange de données, pour le téléphone mobile, la télévision payante, etc . . .

La sureté du protocole RSA réside dans la difficulté à factoriser des grands nombres. La clé publique est un couple (N, e) où N est le produit de deux grands nombres premiers p, q et e un entier compris entre 0 et $\phi(N) = (p-1) \times (q-1)$. On considère qu'un message M est un chiffre, la

Algorithme d'exponentiation rapide
On veut calculer C^{d_A}
1 - Calculer la décomposition binaire de d_A, $d_A = \overline{d_n d_{n-1} \dots d_1 d_0}^2$
2 - $T \leftarrow C$
3 - **Pour** $i = n-1$ **à** 0 **faire**
4 - $T \leftarrow T \times T$
5 - **Si** $d_i = 1$ **Alors** $T \leftarrow T \times C$
6 - **Renvoyer** T

FIGURE 3.2 – Algorithme d'exponentiation rapide

méthode de chiffrement consiste alors à élever le message à la puissance e, modulo N, le message chiffré $C = M^e \mod (N)$. La méthode de déchiffrement utilise des propriétés sur les entiers et en particulier, le petit théorème de Fermat.

L'entier e choisi, n'est pas vraiment pris au hasard, il doit être possible d'associer à e, un entier d compris entre 1 et $\phi(N)$ qui soit l'inverse de $e\ mod(\phi(N))$. Ce qui se traduit par le fait que $e \times d \equiv 1\ mod(\phi(N))$.

La méthode de déchiffrement est alors simplement une exponentiation à la puissance d. Pour retrouver le message d'origine, il suffit d'élever le message crypté C à la puissance d, puisque $C^d = (M^e)^d = M^{(ed)} \equiv M\ mod(N)$. L'entier d est appelé la clé privée associée à la clé publique (N, e).

L'attaque SPA est une attaque simple à canaux cachés. Eve (l'attaquante) peut retrouver des informations par l'analyse de la courbe représentative de la consommation de courant lors de l'exponentiation à la puissance d_A. Cette courbe de consommation est différente suivant les instructions exécutées et les données manipulées. La méthode la plus utilisée pour élever à une puissance donnée est l'algorithme d'exponentiation rapide.

L'algorithme est détaillé dans la figure 3.2.

Lorsque Eve analyse de près l'algorithme, elle s'aperçoit qu'il n'y a jamais deux multiplications consécutives, en effet, à chaque itération, un carré est calculé, mais la multiplication n'est pas toujours faite. Il y a clairement deux comportements différents de la courbe. La courbe de consommation suit un tracé régulier, qui traduit les opérations effectuées pendant l'exponentiation, soit un carré, soit une multiplication. Comme ces deux opérations ne nécessitent pas le même nombre d'opérations élémentaires, la consommation pour effectuer un carré et une multiplication n'est pas la même. Utilisant le fait qu'il ne peut y avoir deux multiplications consécutives, Eve peut déterminer les parties de la courbe correspondant à un carré et les parties de la courbe correspondant à une multiplication. Ce qui directement lui permet de savoir si le bit de l'exposant, lors de l'itération lue, est à 0 ou à 1.

Par une lecture attentive de la courbe de consommation, Eve peut déterminer l'exposant utilisé, c'est-à-dire retrouver d_A.

Contre-mesures Les contre-mesures sont les modifications à apporter à l'algorithme pour rendre l'attaque décrite inefficace.

L'attaque SPA que nous venons de décrire utilise une faille cachée dans la courbe de consommation de courant.

Afin d'empêcher cette attaque, la parade est simple. Il suffit de faire en sorte que la courbe de consommation de courant ne puisse apporter d'information sur l'exposant. Ce qui équivaut à ce que la courbe de consommation soit uniforme. Si la consommation de courant est la même pour toutes les itérations, il ne sera plus possible de différencier les itérations entre : celles où un carré seul est fait et celles où un carré et une multiplication sont faits. Ceci est très facilement réalisable en ajoutant des opérations factices dans l'algorithme d'exponentiation rapide. On modifie l'algorithme d'exponentiation rapide de la manière suivante :

Modification des étapes 4 et 5
4 - **Si** $d_i = 0$ **Alors** $T \leftarrow T \times T \,\&\, V \leftarrow T \times C$
5 - **Si** $d_i = 1$ **Alors** $T \leftarrow T \times T \,\&\, T \leftarrow T \times C$

A chaque itération de cet algorithme, un carré et une multiplication sont faits. La courbe de consommation a un comportement constant quel que soit le bit de l'exposant lu. L'accumulateur V permet de stocker le résultat de l'opération factice, sans que cela interfère dans le résultat final.

Les attaques par corrélation consistent en des tests statistiques pour différentes entrées de l'algorithme et se terminent lorsqu'une corrélation entre le biais physique prédit et la mesure de celui ci se revèle. Pour réaliser des attaques par corrélation, l'adversaire a besoin de modèles et de relations entre la consommation d'énergie du matériel et la valeur de la donnée manipulée.

Sinon, le calcul de corrélation ne peut jamais résulter d'un grand rapport signal à bruit même si les prédictions faites par l'attaquant sont correctes.

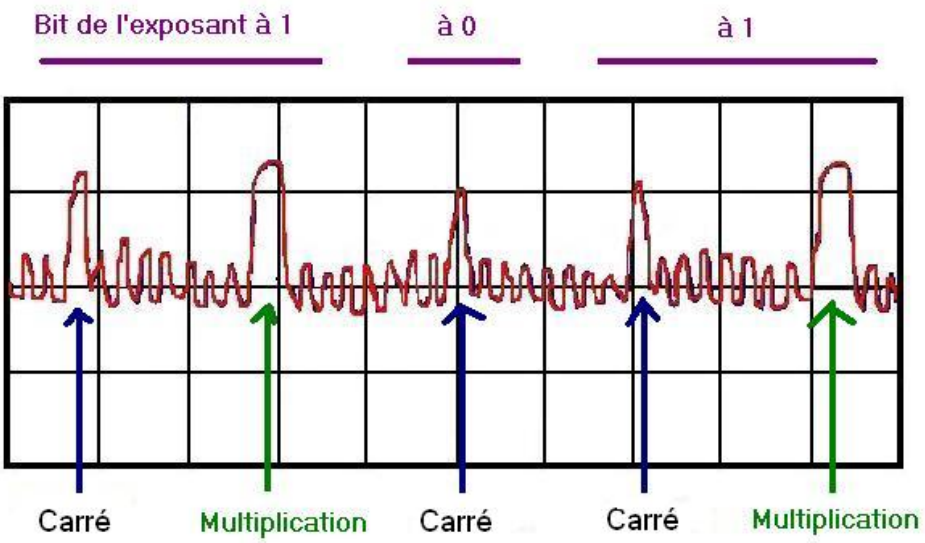


FIGURE 3.3 – Consommation de l'exponentiation rapide

DPA *versus* HODPA

L'exemple le plus connu d'attaque par corrélation est l'attaque par analyse de différences de potentiels : Differential Power Analysis attack (DPA). Dans une attaque DPA, l'attaquant se concentre sur la consommation d'énergie d'une seule instruction et réalise des tests statistiques pour révéler des corrélations entre la distribution de ces mesures et la *donnée sensible* (*i.e.* dépendant d'une valeur secrète) manipulée pendant l'instruction. Depuis l'introduction de DPA dans [69], de nombreuses améliorations ont été publiées (voir [5, 21, 36, 93] par exemple). Parmi elles, les *Higher order DPA attacks* (HODPA) sont d'un intérêt particulier. Elles étendent la DPA en considérant simultanément un ensemble de plusieurs instructions au lieu d'une seule. Le nombre d des instructions considérées par l'attaquant sont appelées *ordre* de la DPA. Depuis que les HODPA analysent le comportement d'un matériel à différents moments, leur efficacité est fortement influencée par le bruit induit durant les phases de mesures et par la désynchronisation (voir par exemple [121]). En fait, réaliser des attaques HODPA efficaces sur du matériel utilisé aujourd'hui dans des applications sensibles (*i.e.* banques et GSM) est toujours un problème et la réalisation nécessite un très grand nombre de mesures ciblées, car les appareils sont généralement fournis avec des mécanismes de sécurité du matériel (tel que désynchronisation, contrôleur de la consommation d'énergie etc) qui cachent l'information derrière un bruit matériel. Ce n'est pas la même situation pour les attaques DPA qui sont beaucoup moins sensibles au bruit produit par le matériel.

Pour les raisons ci-dessus, dans ce chapitre, nous nous concentrerons uniquement sur un pirate qui essaie de réaliser une SPA ou une DPA de premier ordre contre notre implémentation. Même si nous pensons que la stratégie que nous décrivons ci-après pour contrecarrer la DPA de premier ordre rend également très difficile les attaques template, nous n'allons pas discuter de ce fait et nous consacrerons notre analyse à la SPA et DPA (aussi appelé DPA de premier ordre et notée par 1O-DPA).

Nous allons présenter, dans la section suivante, les grandes lignes de la stratégie de défense que nous allons appliquer à la protection de l'algorithme décrit dans la figure 3.1 page 37.

Stratégie de défense

La façon la plus courante pour contrecarrer les SPA et DPA implique des valeurs aléatoires (appelées *masques*) utilisées pour décorréler le signal de la fuite des données sensibles qui sont manipulées [36, 54, 5]. Cette méthode de protection est habituellement appelée *masquage de premier ordre*. Il a été dit dans plusieurs documents récents (*i.e.* [35, 114, 97]) que cette méthode est judicieuse (en combinaison avec les protections habituelles) pour protéger un algorithme contre la SPA et toutes sortes de 1O-DPA.

Dans une méthode de masquage de premier ordre d'un algorithme, toute variable sensible y, apparaissant dans l'algorithme, n'est jamais directement manipulée par l'appareil et est représentée par deux valeurs $\widetilde{y}$ (la *donnée masquée*) et M (le *masque*). Pour assurer la résistance aux DPA, le masque M prend des valeurs aléatoires et pour s'assurer de l'exhaustivité, $\widetilde{y}$ satisfait :

$$\widetilde{y} = y \oplus M \quad . \tag{3.1}$$

Comme y est sensible, toute fonction S de y est aussi sensible si S est connue par l'attaquant. Soit z cette nouvelle valeur sensible $S(y)$. Pour masquer le traitement de z, sans révéler d'informations sur y, deux nouvelles valeurs de $\tilde{z}$ et N doivent être calculées à partir de $(\widetilde{y}, M)$ (ce qui

représente y dans l'implémentation), telles que

$$\tilde{z} \oplus N = z = S(y) \ . \tag{3.2}$$

Le point critique de cette méthode est de déduire la nouvelle paire de (valeur masquée)/masque $(\tilde{z}, N)$ à partir de la paire précédente (y, M) sans compromettre la sécurité du schéma en restant sûr vis à vis de la 1O-DPA. Ce problème est souvent évoqué comme le *problème de correction de masque*. Quand S est linéaire, il peut être résolu de façon très efficace puisque nous avons :

$$z = S(y) = S(y \oplus M \oplus M) = S(\widetilde{y}) \oplus S(M) \ , \tag{3.3}$$

Par conséquent, nous avons tout simplement à définir $\tilde{z}$ et N tels que $\tilde{z} = S(\widetilde{y})$ et $N = S(M)$.

Utiliser le masque de correction des problèmes quand S est non-linéaire est beaucoup plus difficile. De nombreux articles qui ont été publiés ont pour but de s'attaquer à cette question (une vue d'ensemble des méthodes existantes est proposée dans [114]). Ainsi qu'il est démontré dans [114], lorsque les dimensions de l'entrée et de la sortie n et m de la fonction S sont petites, alors la méthode de *Re-Calcul* (REC) est la plus appropriée car elle n'exige qu'un seul transfert mémoire et le pré-traitement d'une table de la RAM de 2^n éléments (une seule fois par exécution de l'algorithme) :

Méthode de Re-Calcul : Soient M et N deux variables aléatoires et supposons que la RAM look-up table S^* associée à la fonction $y \mapsto S(y \oplus M) \oplus N$ a été pré-calculée. Ensuite, pour calculer $S(y) \oplus N$ avec $\tilde{y} = y \oplus M$, la méthode REC effectue une opération unique : la table look-up $S^*[\tilde{y}]$.

Comme nous le verrons dans les prochaines sections, appliquer le masquage de premier ordre au protocole de Stern induit seulement un très petit coût en temps et un coût acceptable en mémoire, car presque toutes les opérations effectuées sont linéaires (et donc la relation (3.3) est appliquée la plupart du temps). En outre, pour les quelques opérations non-linéaires qui doivent être protégées (en particulier lorsque la permutation σ est calculée), nous pouvons appliquer efficacement la méthode REC puisque les dimensions des sous-fonctions concernées sont de petite taille.

Remarque 5

*Le masquage de premier ordre peut théoriquement être attaqué par une DPA de second ordre visant à la fois le masque et la valeur masquée. Pour contrecarrer la 2O-DPA (et, plus généralement, les dO-DPA), toutes les variables sensibles doivent être masquées avec 2 (resp. d) valeurs aléatoires. Cela implique que le compromis temps-mémoire des contre-mesures existantes à base de masquage augmente fortement avec l'ordre de la DPA, qu'ils visent à contrer (*i.e. *[121]). Pour les applications où les contraintes de temps sont très fortes (comme les applications sans contact), un seul masquage est généralement considéré comme suffisant pour masquer toutes les données sensibles.*

Dans ce chapitre, nous allons suivre une stratégie de défense basée sur un seul masque pour rendre la consommation d'énergie, liée à la manipulation de toutes les données sensibles, non corrélée à la valeur des données.

3.4 Spécification de l'algorithme

Dans la suite, on se concentre sur les quatre opérateurs définis dans la section 3.2. Pour chacun d'entre eux, nous exhibons une implémentation efficace et discutons de la manière de la protéger contre des attaques SPA et DPA.

3.4.1 Produit matrice-vecteur

Description de l'algorithme pour une matrice quasi-cyclique

L'algorithme 3.4 calcule le produit matrice-vecteur où la matrice $(\frac{n}{2}, n)$ $\tilde{H} = (I|A)$ est quasi-cyclique. Supposons que A est de taille $\frac{n}{2} \times \frac{n}{2}$ et est générée par un $\frac{n}{2}-$vecteur ici noté $matrix$.

Soit $result$ un $\frac{n}{2}$-vecteur et soit v le n-vecteur représentant le vecteur à multiplier, divisé en deux moitiés $v1$ et $v2$.

Soit$|reg|$ le nombre de bits contenus dans un registre du processeur et soit $nblocs$ le nombre de blocs de $matrix$: $nblocs = \lceil \frac{n}{2\times|reg|} \rceil$.

Discussion sur la sécurité vis à vis des attaques SCA

Comme nous l'avons vu dans la section 3.3, de l'information concernant la donnée sensible y peut fuir au cours du produit matrice-vecteur $\tilde{H}y^T$ (Menace A section 3.3 page 38) et un masque de premier ordre doit être appliqué. Comme ce produit est linéaire pour l'addition bit à bit et en raison de la relation (3.3), masquer le calcul est simple et implique un surcoût temps/mémoire acceptable.

Avant de calculer $result = \tilde{H}y^t$, le vecteur y est masqué avec un masque de n-bits M (généré aléatoirement). Alors l'algorithme 3.4 est utilisé avec en entrée $matrix$ (*i.e.* la première ligne de A) et $\tilde{y} = y \oplus M$. La sortie correspondante est $\tilde{H}\tilde{y}^t = result \oplus N$, où N représente la valeur $\tilde{H}M^t$.

Comme toutes les coordonnées de y sont masquées avec une valeur distribuée de manière uniformément aléatoire, l'analyse SPA ou 1O-DPA du produit matrice-vecteur ne donne pas d'information sur y. Pour rendre le futur démasquage de $\tilde{y}$ possible, un second produit matrice-vecteur $N = \tilde{H}M^T$ est réalisé pour être stocké en mémoire avec $\tilde{y}$.

Le temps de calcul est multiplié par deux (car il y a deux produits matrice-vecteurs au lieu d'un) et $2n$ additions sont nécessaires pour stocker les masques M et N.

Discussion de la complexité : version non sûre

Pour une matrice quasi-cyclique de taille $\frac{n}{2} \times n$ dont la première ligne est de poids $p+1$, les étapes suivantes sont effectuées :

- $n/2$ bits pour extraire et tester la première ligne de la matrice ;
- $\lceil \frac{n}{2\times|register|} \rceil$ décalages binaire du vecteur ;
- $p\lceil \frac{n}{2\times|register|} \rceil$ registres à ajouter au résultat.

Discussion de la complexité : version sûre

Pour une matrice quasi-cyclique de taille $\frac{n}{2} \times n$ dont la première ligne est de poids $p+1$, les étapes suivantes sont effectuées :

- masquer la matrice $\tilde{y} = y \oplus M$;
- calculer $\tilde{H}\tilde{y}^t = result \oplus N$, où $N = \tilde{H}M^t$;
- un second produit matrice-vecteur $N = \tilde{H}M^T$ est réalisé et stocké en mémoire avec $\tilde{y}$;

```
 Pour i de 1 à k
   // résultat initialisé avec la première
   // moitié du vecteur à multiplier
   resultat[i] = v1[i]
FinPour

Pour i de 1 à |reg|
   Si i > 1 Alors
     La deuxième moitié (v2) du vecteur à multiplier
     subit une rotation d'un bit vers la gauche
   FinSi
   Pour j de 1 à nblocs (i.e. pour chaque bloc de la matrice)

    Si le i-ème bit de matrice[j] est égal à 1 Alors
       // ajout du vecteur v2 au résultat en
       // commençant par le j-ième bloc
       br = 1 // indice du bloc dans le résultat
       Pour jj de j à nblocs
          resultat[br] = resultat[br] ⊕ v2[jj]
          br = br + 1
       FinPour
       Pour jj de 1 à j - 1
          resultat[br] = resultat[br] ⊕ v2[jj]
          br = br + 1
       FinPour
     FinSi

   FinPour
FinPour
```

FIGURE 3.4 – Produit matrice quasi-cyclique / vecteur

- $n/2$ bits pour extraire et tester la première ligne de la matrice pour le produit $\tilde{H}\tilde{y}^t$;
- $n/2$ bits pour extraire et tester la première ligne de la matrice pour le produit $\tilde{H}M^t$;
- $\lceil \frac{n}{2\times|register|} \rceil$ décalages binaire du vecteur $\tilde{y}$;
- $\lceil \frac{n}{2\times|register|} \rceil$ décalages binaire du vecteur M ;
- $2p \times \lceil \frac{n}{2\times|register|} \rceil$ registres à ajouter aux 2 résultats.

La version sûre nécessite deux produits matrice-vecteurs : un pour le masque et un pour le produit à déterminer. Le coût est alors le double de celui de la version non sûre.

3.4.2 Fonction de hachage

Pour contrer la Menace E section 3.3 page 39, l'implémentation du protocole de Stern doit utiliser une fonction de hachage dont l'implementation est sûre contre les 1O-DPA. Jusqu'à présent, la sécurisation des implémentations de fonctions de hachage contre les attaques SCA n'avait que rarement été étudiée, essentiellement parce que ces fonctions opèrent sur des données non-sensibles (souvent publiques).

Préliminaires

De façon générale, les fonctions de hachage prennent en entrée une partie M_i du message M à hacher, ainsi que le haché H_{i-1} de la partie précédente M_{i-1}. On note H_0 une valeur initiale arbitrairement fixée, utilisée pour calculer le haché H_1 de la première partie M_1.

On peut considérer différents types d'attaques :

1. Attaque par préimage : connaissant la valeur initiale H_0 et le haché $h(H_0; M)$ d'un message M, trouver un message M' tel que $h(H_0; M') = h(H_0; M)$.
2. Attaque par seconde préimage : connaissant H_0 et M, trouver M' tel que $M' \neq M$ mais $h(H_0; M') = h(H_0; M)$.
3. Attaque par free-start préimage : connaissant H_0 et $h(H_0; M)$, trouver H'_0 et M' tels que $h(H'_0; M') = h(H_0; M)$.
4. Attaque par free-start seconde préimage : connaissant H_0 et M, trouver H'_0 et M' tels que $(H'_0; M') \neq (H_0; M)$ mais $h(H'_0; M') = h(H_0; M)$.
5. Attaque par collision : connaissant H_0, trouver M et M' tels que $M' \neq M$ mais $h(H_0; M') = h(H_0; M)$.
6. Attaque par semi-free-start collision : trouver H_0, M et M' tels que $M' \neq M$ mais $h(H_0; M') = h(H_0; M)$.
7. Attaque par free-start collision : trouver H_0, H'_0, M et M' tels que $(H'_0; M') \neq (H_0; M)$ mais $h(H'_0; M') = h(H_0; M)$.

Le taux (*rate*) d'une fonction de hachage est le nombre de blocs du message qu'elle traite pour un seul appel à la fonction de chriffrement. Une fonction avec un taux de $\frac{1}{2}$ fera donc deux chiffrements pour calculer le haché d'un seul bloc. Pour considérer qu'une fonction de hachage est sûre, il faut que la meilleure attaque ait une complexité supérieure à 2^{80}. Les attaques par collision en free-start et semi-free-start sont difficilement exploitables pour les fonctions de hachage itératives qui ont des valeurs initiales prédéfinies.

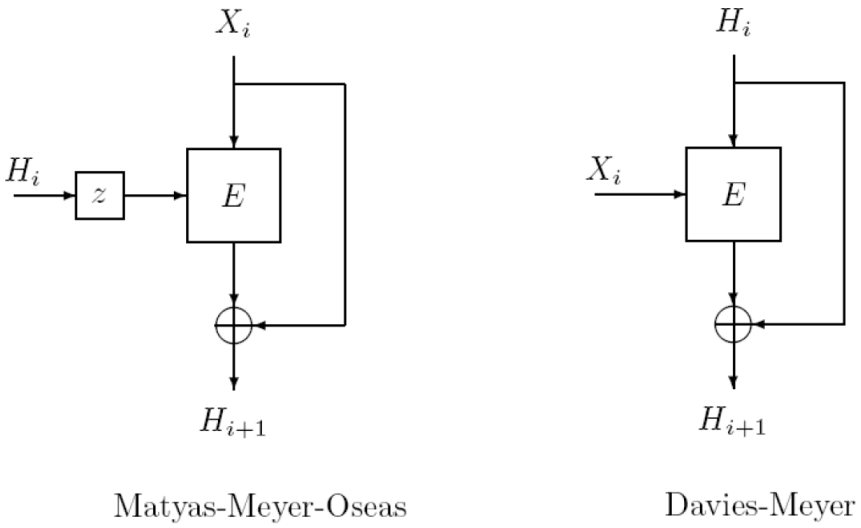


FIGURE 3.5 – Matyas-Meyer-Oseas et Davies-Meyer

Haché de la taille d'un seul bloc

Ces fonctions prennent des blocs de taille m et produisent un haché de taille m. De nombreux schémas de hachage existent mais peu semblent vraiment sécurisés, Preneel, Govaerts et Vandewalle [112] relatent que douze schémas apparaissent sécurisés.

Parmis ces douze schémas, on peut citer le Davies-Meyer et le Matyas-Meyer-Oseas.

$$H_{i+1} = f(H_i; X_i) = E_{z(H_i)}(X_i) \oplus X_i$$

(Matyas-Meyer-Oseas)

$$H_{i+1} = f(H_i; X_i) = E_{X_i}(H_i) \oplus H_i$$

(Davies-Meyer)

Avec f la fonction de hachage, X_i le i-ème bloc à hacher, z une fonction qui adapte H_i à la taille de la clé que prend la fonction de chiffrement E_K de clé K. Une preuve de sécurité pour ces schémas a été donnée en 2002 par J. Black et al. dans [99].

Les attaques contre ces shémas ont cependant les complexités suivantes : $\mathcal{O}(2^m)$ pour les attaques par préimage et $\mathcal{O}(2^{m/2})$ pour les attaques par collision. Ils ne sont donc envisageables que pour des fonctions de chiffrement utilisant des blocs d'au moins 160 bits, afin d'avoir une attaque par collision en $\mathcal{O}(2^{80})$, ce qui n'est pas le cas pour DES et Triple-DES qui utilisent des blocs de 64 bits, ni pour AES qui utilise des blocs de 128 bits.

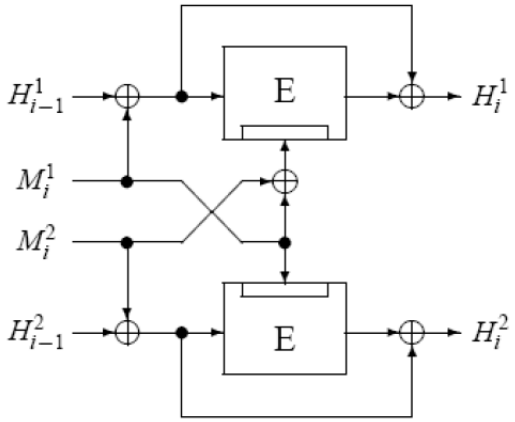
Haché de longueur double

Ces méthodes permettent d'obtenir un haché de la taille de deux blocs, il existe plusieurs méthodes, avec différents taux de hachage, par exemple un Parallel Davies-Meyer ou un LOKI-DBH. Ces deux blocs sont cependant insuffisants dans le cas de DES et Triple-DES, car à partir de blocs de 64 bits, il faut obtenir un haché de 160 bits.

Preneel et Knudsen [64] ont présenté des attaques sur différents schémas :

Parallel Davies-Meyer :

$$\begin{cases} H_i^1 = E_{M_i^1 \oplus M_i^2}(H_{i-1}^1 \oplus M_i^1) \oplus H_{i-1}^1 \oplus M_i^1 \\ H_i^2 = E_{M_i^1}(H_{i-1}^2 \oplus M_i^2) \oplus H_{i-1}^2 \oplus M_i^2 \end{cases}$$



PBGV :

$$\begin{cases} H_i^1 = E_{M_i^1 \oplus M_i^2}(H_{i-1}^1 \oplus H_{i-1}^2) \oplus M_i^1 \oplus H_{i-1}^1 \oplus H_{i-1}^2 \\ H_i^2 = E_{M_i^1 \oplus H_{i-1}^1}(M_i^2 \oplus H_{i-1}^2) \oplus M_i^2 \oplus H_{i-1}^1 \oplus H_{i-1}^2 \end{cases}$$

LOKI-DBH :

$$\begin{cases} H_i^1 = E_{H_{i-1}^2 \oplus M_i^2}(H_{i-1}^2 \oplus M_i^1) \oplus H_{i-1}^1 \oplus H_{i-1}^2 \oplus M_i^1 \\ H_i^2 = E_{H_{i-1}^1 \oplus M_i^1}(H_{i-1}^2 \oplus M_i^2 \oplus H_i^1) \oplus H_{i-1}^1 \oplus H_{i-1}^2 \oplus M_i^2 \end{cases}$$

FIGURE 3.6 – PDM, PBGV et LOKI-DBH

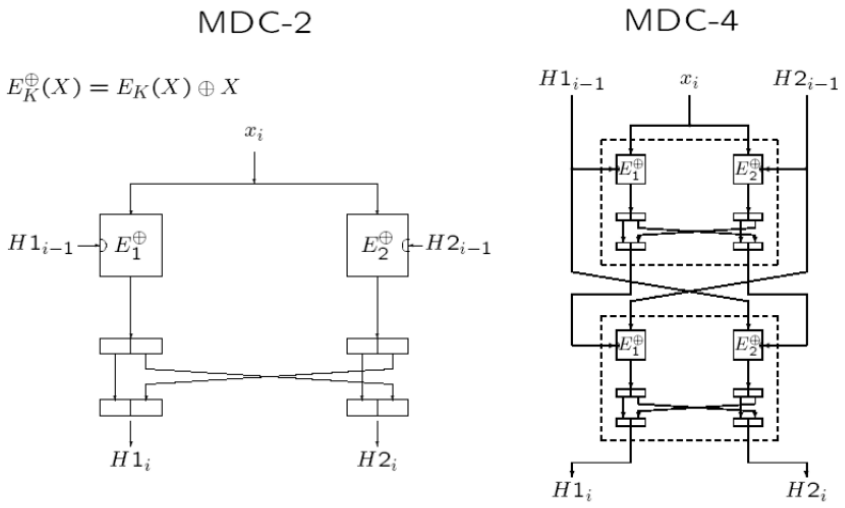


FIGURE 3.7 – MDC2 et MDC4

	taux	collision	préimage	coll(f)	préimage(f)
MDC-2	1/2	2^{55}	2^{83}	2^{28}	2^{54}
MDC-4	1/4	2^{56}	2^{109}	2^{41}	2^{90}

FIGURE 3.8 – MDC-2, MDC-4 : complexité

$$H_i^1 = E_A(B) \oplus C$$
$$H_i^2 = E_R(S) \oplus T$$

Pour un schéma de taux $\frac{1}{2}$, A, B, C sont des combinaisons linéaires des vecteurs de taille m : H_{i-1}^1, H_{i-1}^2, M_i ; et R, S, T sont des combinaisons de : H_{i-1}^1, H_{i-1}^2, M_i et H_i^1. Pour un schéma de taux 1, c'est la même chose sauf que l'on considère que l'on prend également un second bloc du message M à inclure dans les combinaisons. Les trois schémas cités précédemment sont donc de la forme décrite ci-dessus. Knudsen et Preneel ont montré qu'il existe une attaque par free-start préimage en $\mathcal{O}(2^m)$ et par free-start collision en $\mathcal{O}(2^{m/2})$, pour les fonctions de taux 1 ou $\frac{1}{2}$. Pour des schémas de taux 1 et ayant au moins une fonction de type $H_i^1 = E_A(B) \oplus C$, des attaques par préimage se font en $\mathcal{O}(2^m)$ et par collision en $\mathcal{O}(2^{m/2})$ (sauf deux schémas en $\mathcal{O}(2^{3m/4})$). Deux autres fonctions, avec un taux de $\frac{1}{2}$ et $\frac{1}{4}$, MDC-2 et MDC-4, sont elles aussi assez insécurisées.

La figure 3.8, tirée de [110], montre la complexité d'attaques sur les hachés et sur la fonction de hachage, en utilisant DES comme fonction de chiffrement.

La meilleure attaque contre MDC-4 a une complexité en $\mathcal{O}(2^{\frac{3m}{4}})$ pour des collisions sur la fonction, il conviendrait donc pour AES avec des collisions sur la fonction en $\mathcal{O}(2^{96})$ et une complexité supérieure à $\mathcal{O}(2^{128})$ pour les attaques par préimage.

Longueur double mais clé de longueur double

Ces schémas utilisent une fonction de chiffrement dont la clé est de longueur $2m$ pour des blocs de longueur m. Les deux principaux schémas [71] sont basés sur Davies-Meyer, le Tandem-DM

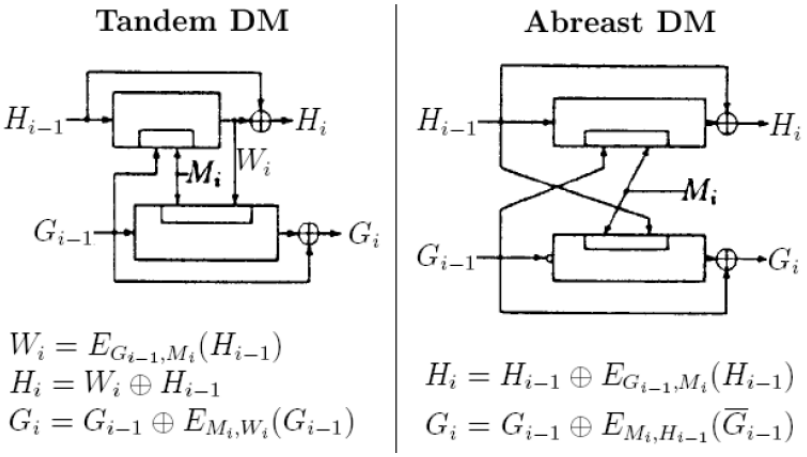


FIGURE 3.9 – TandemDM et AbreastDM

et le Abreast-DM. Les attaques se font en $\mathcal{O}(2^{2m})$ par préimage et $\mathcal{O}(2^m)$ par collision ce qui est donc suffisament sécurisé pour AES-256 qui utilise une clé de longueur double.

Haché de taille supérieure

Des schémas triples et quadruples ont été conçus à partir de fonctions produisant des hachés de la taille d'un bloc. Mais, comme précisé dans une publication de Preneel [66], ils ne sont pas beaucoup plus résistants que les schémas produisant un haché de taille $2m$, une attaque par collision est possible en $\mathcal{O}(2^m)$ en considérant que les sous-fonctions utilisées ont elles aussi une résistance aux collisions en $\mathcal{O}(2^m)$. Knudsen et Preneel [66, 65] ont également conçu des fonctions de hachage basées sur des instanciations multiples de fonctions Davies-Meyer. Des codes linéaires sont utilisés pour définir les entrées de chaque fonction. En décembre 2005, Preneel [110] a cependant dit que la proposition de preuve de sécurité nécessitait une légère correction. Ainsi Preneel annonce 2^{72} au lieu de théoriquement 2^{81}, comme meilleure attaque sur une fonction basée sur DES utilisant un code dont la distance est de 4. Ce schéma produit des hachés de taille au moins $5m$, il est donc nécessaire de les réduire, Preneel et Knudsen proposent une méthode basée sur le mode CBC.

Fonctions de chiffrement alternatives

Pour augmenter la sécurité des schémas de hachage, on peut également utiliser des fonctions de chiffrement construites avec des fonctions ayant des blocs plus petits. On peut citer par exemple $DEAL$ [63, 83, 61] et son amélioration $DEAL^{KX}$ [84], qui utilisent des blocs de 128 bits et sont construites à partir de DES. Ces deux fonctions sont cependant très coûteuses car il faut 6 DES pour le chiffrement et 6 DES pour l'extension de clé. Les travaux de Luby et Rackoff sur les permutations et fonctions pseudo-aléatoires [81, 82], suivis de nombreuses autres publications, permettent de transformer un générateur d'aléas en fonction de chiffrement basée sur le schéma de Feistel et utilisant une taille de blocs choisie arbitrairement. De récents travaux [88] étudient la sécurité de ce chiffrement en fonction de la qualité du générateur d'aléa. Une autre alternative permet de doubler la taille des blocs d'une fonction, suivant les travaux de Aiello, Haber et Venkatesan [4, 3] qui

|hachage|	Fonction	PR	CL	Taux	|Clé|	DES	TDES	AES
m	12 schémas	2^m	$2^{\frac{m}{2}}$	1	m	2^{27}	2^{32}	2^{64}
$2m$	$E_A(B) \oplus C$	2^m	$2^{\frac{m}{2}}$	1 ou $\frac{1}{2}$	m	2^{27}	2^{32}	2^{64}
	sauf 2 schémas	2^m	$2^{\frac{3m}{4}}$	1	m	2^{41}	2^{48}	2^{96}
	Tandem-DM Abreast-DM	2^{2m}	2^m	$\frac{1}{2}$	$2m$	-	2^{64}	2^{128}
	MDC-2	2^m	$2^{m/2}$	$1/2$	m	2^{28}	2^{32}	2^{64}
	MDC-4	$2^{\frac{3m}{2}}$	$2^{\frac{3m}{4}}$	$1/4$	m	2^{41}	2^{48}	2^{96}
$n.m$	codes $\mathbb{F}_{2^2}$ $[n, k, d]$							
	[9,5,4]	$2^{(d-1)m}$	$2^{\frac{(d-1)m}{2}}$	$1/9$	m	2^{81} ou 2^{72}	2^{96}	2^{192}
	[16,12,4]	$2^{(d-1)m}$	$2^{\frac{(d-1)m}{2}}$	$1/2$	m	2^{81} ou 2^{72}	2^{96}	2^{192}
	codes $\mathbb{F}_{2^4}$ $[n, k, d]$							
	[9,6,4]	$2^{(d-1)m}$	$2^{\frac{(d-1)m}{2}}$	$1/3$	m	2^{81} ou 2^{72}	2^{96}	2^{192}
	[16,13,4]	$2^{(d-1)m}$	$2^{\frac{(d-1)m}{2}}$	$5/8$	m	2^{81} ou 2^{72}	2^{96}	2^{192}

FIGURE 3.10 – Récapitulatif fonctions de hachage

utilisent le schéma Benes pour biaiser le paradoxe des anniversaires et une autre pour remplacer le cadencement de clé du DES. Des travaux récents ont également été faits sur Benes [103, 102].

Récapitulatif

La figure 3.10 donne les complexités des meilleures attaques en fonction du chiffrement utilisé, les attaques en free-start ou semi-free-start collision n'ont pas été prises en compte.

Le tableau suivant, extrait de [110], présente également la taille qu'un haché doit avoir pour résister à des attaques en force brute dont la durée est d'un an de calcul. Pour les attaques en seconde préimage, on considère que $65\,536$ messages sont attaqués en parallèle.

attaquant	coût en $	outil	2nde preim 2006	2nde preim 2015	coll. 2006	coll. 2015
Attaquant de base	400	FPGA	74	80	115	127
Petite entreprise	10 000	FPGA	79	85	125	137
Moyenne entreprise	300 000	ASIC	90	96	147	159
Grande entreprise	10×10^6	ASIC	95	101	158	169
Services secrets	300×10^6	ASIC	100	106	162	174

FPGA = Field Programmable Gate Array
ASIC = Application Specific Integrated Circuit

On peut interpréter une ligne du tableau de la façon suivante : en 2006, pour 400$ et avec une durée d'un an, on peut réaliser une attaque par collision d'une complexité de $2^{115/2} = 2^{57}$ et une attaque par préimage d'une complexité de 2^{74}.

Conclusion

Les schémas intéressants pour AES sont MDC-4 avec un taux de 1/4, ainsi que les Tandem-DM et Abreast-DM avec un taux de 1/2. Il n'y a pas de solution vraiment intéressante pour le DES et TDES, en raison de la taille trop réduite de leurs blocs. On peut quand même envisager une solution basée sur des codes de distance 4 ou 5, il y aura alors beaucoup de tours à effectuer comparé au nombre de blocs à hacher qui est de 8. Il faut donc compter au moins quinze chiffrements pour hacher ces blocs et réduire le haché à une taille de 192 bits, il faut aussi ajouter l'extension des morceaux de blocs afin de préparer, à chaque itération, les entrées de la fonction de hachage.

Un autre problème est la taille des hachés qui doit être idéalement de 160 bits, mais qui sera un multiple de la taille d'un bloc, 256 bits pour AES et 192 bits pour DES. Réduire ces tailles en coupant un morceau risque de faciliter les attaques par collision.

Toutefois, Lemke *et al.* [74] ou McEvoy *et al.* [87] ont montré que, dans certaines applications comme l'authentification HMAC, réaliser des attaques DPA contre les fonctions de hachages a un sens quand le secret (ou données privées) doit être haché avec des données publiques. Dans [87], les auteurs exhibent un moyen de protéger une implémentation matérielle de la fonction de hachage SHA-256 contre les 1O-DPA.

Dans le reste du chapitre, nous supposons que nous possédons une version sûre de SHA-256. On peut remarquer que la méthode de masquage utilisée par McEvoy *et al.* pour l'implémentation matérielle peut aussi être suivie pour construire une implémentation software de SHA-256. Toutefois, dans ce cas, le surcoût temps-mémoire devient très grand. En fait, si l'appareil ne possède pas d'implémentation de SHA-256 sécurisée, il peut être pertinent d'utiliser une fonction de hachage basée sur des constructions à base d'algorithmes de chiffrement par blocs (l'état de l'art des fonctions de hachages publié par Preneel dans [111] donne de nombreux exemples de telles fonctions). En effet, dans un cas pareil la fonction de hachage peut hériter de la sécurité vis à vis des attaques DPA qui est liée au schéma de chiffrement par blocs sous jacent. De nos jours les appareils embarqués possèdent pour la plupart, une version matérielle de DES et parfois de l'AES ce qui inclut des mécanismes contre les DPA. Dans le cas où ni une fonction de hachage sûre ni un algorithme de chiffrement par blocs ne sont implémentés dans l'appareil alors il est toujours possible d'utiliser une des nombreuses versions softwares proposées dans la littérature de DES ou AES sûres contre les attaques DPA (voir par exemple [5, 35, 55, 113, 119, 131]) et de les utiliser dans une fonction de hachage basée sur des algorithmes de chiffrement par blocs (comme par exemple MDC-4).

Remarque 6

> *Pour qu'une fonction de hachage garantisse une certaine sécurité, la longueur des valeurs des hachés produits doit être d'au moins* 160 *bits. Quand on utilise une fonction de hachage basée sur des algorithmes de chiffrement par blocs, il peut être difficile d'avoir des hachés d'une telle longueur. Dans ce cas, une solution peut être de concaténer plusieurs blocs de sortie jusqu'à obtenir une longueur de* 160 *bits ou plus et, si nécessaire de tronquer dans le but d'obtenir une longueur d'exactement* 160 *bits (par exemple deux chiffrements par AES donnent* 256 *bits qui peuvent être tronqués à* 160 *bits).*

Peu de cartes à puce disposent d'une fonction de hachage codée en hardware, là aussi, il est

intéressant d'utiliser une fonction de chiffrement par blocs. Le problème de ce type de fonction est également d'obtenir un haché de 160 bits à partir de fonctions de chiffrement utilisant des blocs de taille inférieure, comme DES, Triple-DES et AES. Preneel a publié en 2005 un état de l'art des fonctions de hachage [111] et [110].

Description de l'algorithme : implémentation matérielle sûre de SHA-256

Dans [87], McEvoy et al. décrivent un masque de premier ordre pour une implémentation de HMAC basée sur SHA-256. En utilisant une implémentation sur FPGA, ils présentent une implémentation qui est conçue pour résister aux attaques 1O-DPA.

Discussion sur la sécurité vis à vis des attaques SCA

Dans [87], la résistance de l'implémentation de SHA-256 est formellement analysée et démontrée.

Discussion de la complexité

Il est montré dans [87] que le processeur et que l'interface de circuit correspondant à la version masquée de SHA-256 nécessitent 1734 slices (37% des ressources du FPGA) et que le chemin critique dans le matériel (i.e. le plus long chemin combinatoire) est 18.6. Comme expliqué dans [87], la place a quasiment doublée comparée à la version non protégée mais la vitesse n'a pas vraiment été affectée.

3.4.3 Permutation

Sécurité des permutations

Attaque à chiffré connu Ne connaissant que le permuté, l'attaquant souhaite retrouver le vecteur non-permuté. Bien qu'il y ait $n!$ permutations possibles, pour un vecteur de taille n et de poids p, il n'y a que $\binom{n}{p} = \frac{n!}{p!(n-p)!}$ permutés possibles, cette valeur doit donc être supérieure ou égale au nombre de permutations que peut indexer le germe.

En considérant que la permutation soit prise aléatoirement dans les $n!$ possibles, la meilleure solution est d'essayer toutes les valeurs possibles du germe, puis de tester chaque vecteur obtenu pour voir si c'est le bon. Si le vecteur attaqué est le permuté du secret, le test peut être une multiplication avec la matrice H pour vérifier si le résultat est égal à l'identifiant public.

Attaque à germe connu Si l'attaquant connait le germe et le permuté, il peut retrouver le vecteur initial si la permutation n'est pas à sens unique.

Attaque à clair connu Si l'attaquant connait le clair et le chiffré, il peut vouloir connaître le germe afin d'attaquer le générateur d'aléa qui en est à l'origine, ou en vue de connaître un autre vecteur permuté avec ce germe. Le vecteur initial ayant le même poids p que son permuté, l'attaque peut se diviser en deux parties : $p!$ possibilités à essayer afin de trouver les permutations sur les bits à 1 et $(n-p)!$ essais pour trouver les permutés des bits à 0. La complexité totale de l'attaque sera

donc de l'ordre de $p!(n-p)!$ au lieu de $n!$ idéalement. La complexité des attaques à chiffré ou clair connu reste très élevée, elles ne sont pas réalisables même pour $n = 256$.

Pour un vecteur de n bits, il existe $n!$ permutations possibles. On peut représenter une permutation par une suite de n valeurs, chacune indiquant, pour le bit correspondant à son indice, la position à laquelle se trouve ce bit dans le permuté du vecteur. $\{ x_1, ..., x_i, ..., x_n \}$ avec $i =$ indice des bits du vecteur à permuter et x_i = position dans le permuté du bit d'indice i.

On représente donc une permutation sur $n.log_2(n)$ bits, avec une complexité de calcul de $\mathcal{O}(n)$ car il faut effectuer n déplacement de bits. La suite des nombres x_i est assujettie à la contrainte de contenir chaque nombre de 1 à n.

$n.\log_2(n)$ est cependant une valeur élevée, il serait intéressant de réduire cette taille pour accélérer l'identification. La taille optimale qui permet de définir une permutation est $\log_2(n!)$ bits, ce qui reste encore élevé. Stern suggère d'utiliser un germe (*seed*) de 120 bits, ce qui permet de coder 2^{120} permutations. Or $32! < 2^{120} < 33!$, il faut donc que ces 2^{120} permutations semblent prises aléatoirement parmi les $n!$ afin que la sécurité ne corresponde pas à celle d'un vecteur de 32 ou 33 bits. La puissance de calcul ayant fortement augmenté depuis ces recommandations, un germe de 160 bits est plus approprié.

Comme pour le produit matrice-vecteur, il est possible d'optimiser le calcul de la permutation en ne permutant que les 1 dans un résultat initialisé à 0, ou inversement. On peut envisager différents générateurs de permutations aléatoires.

Matrice de permutation

Une solution simple consiste à utiliser une matrice de permutation, contenant un seul bit à 1 par ligne et par colonne. L'inconvénient est la complexité du produit de matrice en $\mathcal{O}(n^2)$ et la taille de la matrice qui est du même ordre. Le germe de permutation définirait alors la position des bits dans chaque ligne de la matrice. L'utilisation d'une forme compressée de la matrice est possible, cette solution s'apparente au générateur $n!$ décrit à la sous-section suivante. Une autre solution est d'utiliser un code correcteur générant des mots du même poids que le vecteur à permuter. On a alors une matrice de taille $160n$, elle aussi sûrement compressible. On multiplie alors le germe avec cette matrice pour obtenir un mot de n bits. Un problème posé par cette solution est que cette matrice doit être construite en fonction de chaque vecteur à permuter. L'inverse est aussi envisageable, une matrice construite à partir du germe, multipliée par le vecteur à permuter.

Générateur $n!$

Un tel générateur permet de générer toutes les permutations possibles à partir d'un nombre compris entre 1 et $n!$. Knoth et Knuth ont conçu des générateurs de ce type [62, 67]. L'algorithme décrit dans cette section s'apparente à celui de Knuth dont le nombre compris entre 1 et $n!$ est représenté en notation "mixed-radix", c'est-à-dire un vecteur de n nombres : $\{x_0, ..., x_{n-1}\}$ avec $x_i = \sum_{i=0}^{n-1} x_i.i!$ et $0 \leq x_i \leq i$. On peut donc générer les $n!$ permutations à partir de $\mathcal{O}(n.\log_2(n))$ bits, avec les algorithmes suivants :

- algorithme 1 :

```
Pour chaque bit i de 1 à n
  On prend j compris entre i et n
  On échange les bits i et j
FinPour
```

- algorithme 2 :

```
Pour chaque bit i de 1 à n
  On prend j compris entre 0 et n-i
  On échange les bits i et i+j
FinPour
```

Complexité en stockage pour les n valeurs j :

- $n.\log_2(n)$ pour l'algorithme 1
- $\sum_{i=1}^{n} \log_2(i)$ pour l'algorithme 2 (valeur légèrement inférieure à $n.\log_2(n)$)

Complexité en calcul : n échanges. Si on utilise un germe et qu'on l'étend sur $n.\log_2(n)$ bits, la complexité de calcul passe alors au mieux à $\mathcal{O}(n.\log_2(n))$, mais la complexité en stockage peut être réduite à $\mathcal{O}(1)$ si il n'y a qu'une seule permutation à faire, ou si toutes les permutations sont faites simultanément.

Le problème de ce schéma est qu'il faut utiliser des opérations modulaires pour le calcul de j compris entre i et n. Une solution peut consister à générer les nombres aléatoires j par addition de nombres aléatoires compris entre 0 et chaque puissance de 2 qui compose j. Ce qui nécessite une génération supplémentaire d'aléas. Il est également possible de faire un calcul modulaire sécurisé en générant deux aléas, on fait un calcul modulaire normal sur le premier et le calcul modulaire "complémentaire" sur le second, puis on ajoute les deux résultats pour avoir la même probabilité d'obtenir chaque nombre.

La sécurité de tels schémas repose entièrement sur la sécurité des générateurs d'aléas utilisés pour générer le germe et de la fonction qui l'étend à la longueur voulue, par exemple un générateur d'aléas initialisé avec le germe.

générateur de permutations aléatoires

Luby et Rackoff ont initié des travaux sur des générateurs de permutations aléatoires basés sur DES [81, 82]. Dans ce schéma, les S-box du DES sont remplacées par des fonctions pseudo-aléatoires. La sécurité augmente avec le nombre de tours.

Ce type de schéma permet de calculer le permuté d'un nombre compris entre 0 et 2^{x-1}. Si l'on veut permuter les n bits d'un vecteur, le bit de position i devra être déplacé en position permuté(i), i étant codé sur $\lceil \log_2(n) \rceil$ bits.

Si le nombre n de bits à permuter n'est pas une puissance de 2, alors, lorsque l'on tombe sur un permuté de i inférieur à n, on associe à i le permuté de $i+1$, jusqu'à trouver un permuté inférieur à n. Dans le pire des cas, la complexité de calcul est de $\mathcal{O}(2^{\lceil \log_2(n) \rceil}.\log_2(n))$, la complexité est donc toujours optimale quand n est une puissance de 2.

On note $I_n = \{0,1\}^n$ l'ensemble des 2^n chaînes binaires de longueur n. On note F_n l'ensemble des fonctions de I_n vers I_n : $F_n = \{f | f : I_n \rightarrow I_n\}$

F_n contient 2^{n2^n} éléments. On note P_n l'ensemble des permutations de I_n vers I_n , on a donc $P_n \subset F_n$.

Soit $f_1 \in F_n$ et $L, R, S, T \in I_n$. $\forall (L,R) \in (I_n)^2$, on définit la fonction $\Psi(f_1)$ de la façon suivante : $\Psi(f_1)[L,R] = [R, L \oplus f_1(R)]$

La fonction Ψ correspond donc à un tour du schéma Feistel, que l'on peut également noter D_{f_1}. $\Psi(f_1, ..., f_k) = \Psi(f_k) \circ ... \circ \Psi(f_1)$

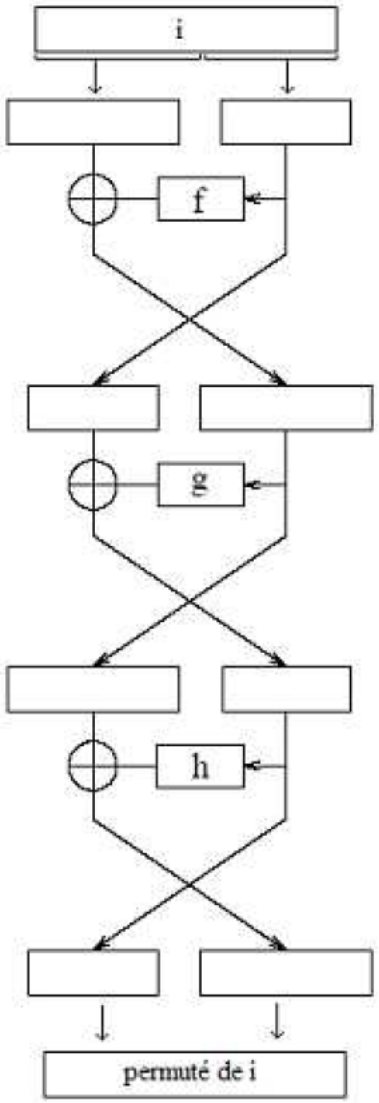


FIGURE 3.11 – Luby - Rackoff

$\Psi(f^i, f^j, f^k)$ n'est pas pseudo-aléatoire, avec $i, j, k \geq 1$.
$\Psi(f^i, f^j, f^k, f^l)$ n'est pas fortement pseudo-aléatoire, avec $i, j, k, l \geq 1$.
$\Psi(f, f, f, f^2)$ est pseudo-aléatoire.
$\Psi(f^2, I, f, f^2, I, f)$ est fortement pseudo-aléatoire, avec I la fonction identité appartenant à F_n.
$\Psi(f, f, f \circ \zeta \circ f)$ est pseudo-aléatoire avec ζ une fonction appartenant à F_n et ayant certaines propriétés, par exemple une rotation de 1 bit.
$\Psi(f, f, f, f \circ \zeta \circ f)$ est fortement pseudo-aléatoire.

FIGURE 3.12 – Constructions à partir d'une seule fonction

$\Psi(f, f, g)$ et $\Psi(f, g, g)$ sont pseudo-aléatoires mais pas fortement pseudo-aléatoires.
$\Psi(f, g, f)$ n'est pas pseudo-aléatoire car symétrique.
$\Psi(f, f, f, g)$ et $\Psi(f, g, g, g)$ sont fortement pseudo-aléatoires.
$\Psi(f, f, g, g)$ est fortement pseudo-aléatoire.
$\Psi(f, g, g, f)$ n'est pas pseudo-aléatoire car symétrique.
$\Psi(f, g, f, g)$ est fortement pseudo-aléatoire.
$\Psi(f, f, g, f)$ et $\Psi(f, g, f, f)$ sont fortement pseudo-aléatoires.

FIGURE 3.13 – Constructions à partir de deux fonctions

On note Ψ^k pour considérer $\Psi(f_1, ..., f_k)$ avec $f_1, ..., f_k$ k fonctions indépendantes choisies aléatoirement dans F_n. Une permutation pseudo-aléatoire (pseudo random permutation) est une permutation qui est indistinguable d'une distribution uniforme à un observateur qui a une puissance de calcul réaliste et qui a accès à la permutation aux points de son choix. On définit aussi une permutation pseudo-aléatoire forte (strong pseudo random permutation ou super pseudo random permutation), qui est indistinguable d'une distribution uniforme lorsque l'observateur a également accès à la permutation inverse.

D'après Luby et Rackoff [82], Ψ^3 est pseudo-aléatoire et Ψ^4 est fortement pseudo-aléatoire.

D'après les travaux de Patarin [101] et de Pieprzyk [107], il est possible d'utiliser plusieurs fois la même fonction, ce qui permet de diminuer les calculs à faire pour générer les fonctions aléatoires à partir du germe.

Le nombre de bits à permuter peut être impair, ce qui force l'utilisation d'au moins deux fonctions. Ces fonctions doivent également pouvoir indexer autant de permutations que le permet le germe. Elles doivent donc être définies sur au moins autant de bits que celui-ci, c'est-à-dire 160. Pour $n = 512$, la permutation s'effectuera sur des nombres de $\log_2(512) = 9$ bits, il faut donc prendre un schéma à au moins deux fonctions, l'une dont les valeurs sont définies sur 4 bits et l'autre sur 5. On devra donc générer $4.2^4 + 5.2^5 = 224$ bits à partir du germe. Pour $n = 256$, les deux fonctions seront définies sur $2.4.2^4 = 128$ bits, ce qui est insuffisant pour générer 2^{160} permutations possibles. On peut donc prendre quatre fonctions et le schéma Ψ^4 proposé par Luby et Rackoff. Il faudra alors générer 256 bits à partir du germe. Pour $n = 1024$, un schéma à une fonction est suffisant car il faut permuter deux blocs de 5 bits, ce qui définit une fonction à utiliser sur 160 bits, qui est la taille du germe. On peut donc utiliser directement celui-ci pour définir les valeurs de la fonction. Là encore, la sécurité repose en grande partie sur la qualité du générateur d'aléa utilisé pour créer le germe et de la fonction utilisée pour étendre le germe, qui doivent donner une suite de bits qui semble la plus aléatoire possible, afin que l'on ne puisse pas déduire le permuté d'un bit à partir du permuté d'un autre. Par rapport aux générateurs $n!$ décrits précédemment, l'avantage de ce schéma est que l'on peut précalculer la valeur des fonctions, en raison de

leur faible taille. Il n'y a donc besoin de générer que très peu de bits à partir du germe, c'est donc un gain en temps de calcul si la fonction d'extension du germe est lente. Les travaux de Luby et Rackoff ont donné lieu à de nombreuses publications, on peut citer, entre autres, [102] et [94] qui proposent des variations sur le schéma et [88] qui étudie la sécurité pour des fonctions faibles.

Nous reviendrons plus tard sur ce type de construction.

Autre schéma

On peut envisager un autre schéma, associant lui aussi une position i d'un bit dans le vecteur initial avec sa position $permuté(i)$ dans le vecteur permuté. Cette méthode consiste à appliquer à la position i d'un bit, une alternance de deux transformations bijectives et non commutatives entre elles, préalablement calculées à partir du germe. Ces transformations sont :

- $fp(i)$: une permutation des bits.
- $fa(i)$: l'addition d'un nombre, ce n'est pas un OU-exclusif.

Le précalcul de la permutation fp doit se faire avec la contrainte de ne pas permuter deux bits à la même position. On peut également diviser en deux la représentation binaire de la position i. On définit ainsi deux fonctions fp_0 et fp_1 qui associent à chaque partie i_0 et i_1 un vecteur de même taille que i, chacun de ces vecteurs contient la moitié du permuté de i par la fonction fp. Cette division en deux parties permet de réduire le stockage nécessaire à cette transformation. Pour calculer $fp(i)$ on fait donc un OU-exclusif de $fp_0(i_0)$ avec $fp_1(i_1)$. La figure 3.14 présente le calcul de $permuté(i)$, avec à droite, le détail de la fonction fp quand elle est divisée en deux parties.

On peut envisager d'utiliser plusieurs fonctions fp et fa différentes afin d'améliorer la sécurité. Par rapport au schéma de Feistel, cette solution présente le désavantage d'avoir une contrainte dans la construction de la fonction fp, deux bits ne peuvent pas être permutés vers la même position. Si on utilise plusieurs fonctions fp différentes, il faut également prévoir un espace de stockage plus important et un temps plus important pour fabriquer toutes les fp. La fonction fa présente l'avantage de pouvoir être calculée directement par le processeur car il s'agit d'une addition, son espace de stockage est donc d'une seule valeur de la même taille que i. Cette solution peut se révéler très intéressante quand la taille du vecteur à permuter est de 256 bits, facilitant ainsi l'addition et le stockage de fp. La position i d'un bit est alors codée sur 8 bits, qui est la taille de registre la plus répandue sur les processeurs de carte à puce. La taille de la fonction fa sera alors de 8 bits et celle de la fonction fp sera de 8.2^8 bits. Si fp est divisée en deux sous-fonctions, alors leur taille sera chacune de 8.2^4 bits, soit 8.2^5 bits pour les deux.

Permutations à sens unique

On peut vouloir des permutations à sens unique pour éviter que l'on retrouve le vecteur initial à partir du germe et du vecteur permuté. Le schéma de Benes semble pouvoir faire ce genre de permutations [4, 102, 103]. Le sens unique se ferait donc sur la fonction qui associe une position dans le vecteur initial à une position dans le vecteur permuté.

On peut également imaginer le schéma suivant :
Soit P une fonction qui permute un vecteur n-bit à partir d'un germe g. Pour un n-bit x on obtient son permuté $y = P_g(x)$. Soit h une fonction de hachage générant un haché de la même taille que le germe et $|$ une fonction qui concatène deux séquences de bits.

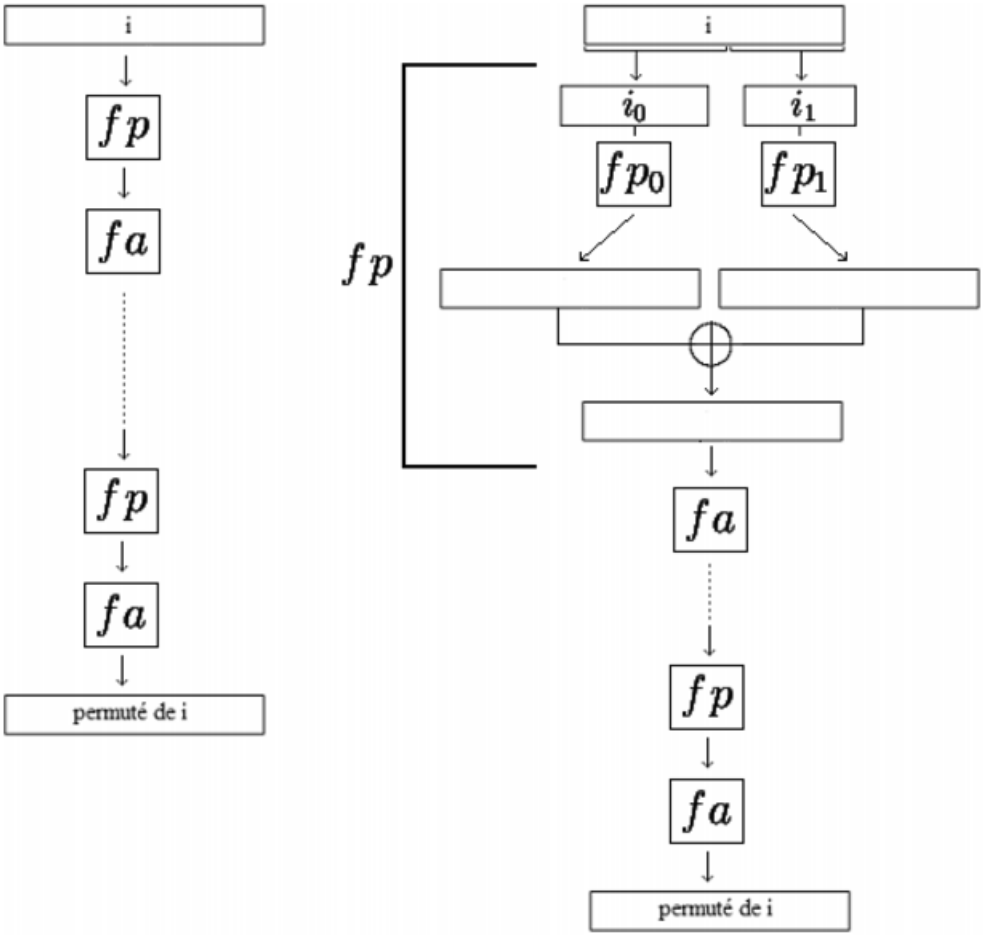


FIGURE 3.14 – Permuté de i

Générateur	**#bits**	n**=512**	**#alea64**	**#alea128**
Algo 1	$2n.\log_2(n)$	9216	144	72
Algo 2	$2.\sum_{i=1}^{n}\log_2(i)$	8194	129	65
Schéma de Feistel	$\phi(n)$	288	4,5	2,25
Autre schéma	$\phi(n)+\lceil\log_2(n)\rceil$	297	4,64	2,32

FIGURE 3.15 – Récapitulatif permutations

- #bits = nombre de bits à générer en fonction de n;
- "n=512" = nombre de bits à générer pour $n = 512$;
- #alea64 = nombre d'aléas de 64 bits nécessaires ;
- #alea128 = nombre d'aléas de 128 bits nécessaires ;
- $\phi(n) = \lceil\log_2(n)\rceil.2^{\lceil\log_2(n)\rceil} + \lfloor\log_2(n)\rfloor.2^{\lfloor\log_2(n)\rfloor}$

Pour obtenir une fonction à sens unique on calcule y de la façon suivante : $y = P_{h(x|g)}(x)$ ainsi, un attaquant qui connaît g et y devra trouver une collision sur $h(x|g)$ avant de pouvoir calculer la fonction P à partir du germe.

Si l'on considère P comme une fonction de chiffrement, on peut difficilement construire une fonction à sens-unique avec l'un des douze schémas de hachage apparaissant comme sûrs [112], en raison de l'ajout d'une valeur en sortie de la fonction de chiffrement, ce qui modifierait le poids du résultat et ne serait donc plus une permutation.

Conclusion

Le tableau 3.15 présente le nombre de bits à générer à partir du germe en fonction des différents schémas envisagés. Le schéma de Feistel est considéré comme utilisant deux fonctions et l'autre schéma utilisant seulement une fonction fp et une fonction fa. Pour les deux algorithmes, on considère une opération modulaire sécurisée prenant deux aléas pour ne donner qu'un résultat.

Le schéma de Feistel semble le plus intéressant, car il réunit une facilité de construction des fonctions, ainsi qu'un faible volume de bits à générer à partir du germe. Il serait intéressant d'évaluer précisément sa sécurité notamment en s'intéressant aux études récentes publiées sur ce sujet. Si l'on considère des attaques traditionnelles prévues pour des fonctions de chiffrement, alors ces permutations sont toujours faibles à partir du moment où l'on peut attaquer la fonction qui associe la position d'un bit dans le vecteur initial, à sa position dans le vecteur permuté. Par exemple, on considère une boîte noire initialisée avec une fonction créée à partir du germe, on peut demander à cette boite, autant de fois que l'on veut, de calculer une position dans le vecteur permuté à partir d'une position dans le vecteur initial. Pour un vecteur n-bit à permuter, la boîte noire prend $\lceil\log_2(n)\rceil$ bits en entrée et la fonction qu'elle contient est définie sur $n.\lceil\log_2(n)\rceil$ bits. Pour $n = 512$, on a donc 9 bits en entrée et une attaque à clair choisit en faisant 2^9 appels à la boîte noire. La sécurité de ces méthodes de permutation s'apparente donc plutôt à l'allure aléatoire de la permutation générée, afin que, disposant d'un vecteur n-bit permuté, on ne puisse pas déduire la position initiale d'un bit à partir de la position d'un autre. Il faut également que chaque bit du vecteur initial ait la même probabilité d'être déplacé à chacune des positions possibles.

Définir une permutation vectorielle σ sur $\mathbb{F}_2^n$ nécessite de définir une *permutation d'indices* ψ

sur $\{0, \cdots, n-1\}$ telle que pour tout $y = (y[0], \cdots, y[n-1]) \in \mathbb{F}_2^n$ on ait :

$$\sigma(y) = (y[\psi(0)], \cdots, y[\psi(n-1)]).$$

Dans ce chapitre, nous choisissons de construire une permutation ψ en suivant l'approche proposée par Luby et Rackoff dans [81, 82] et améliorée dans [101, 107]. Le coeur de l'idée de cette approche réside dans le fait d'utiliser des fonctions pseudo-aléatoires dans un schéma de Feistel. Comme expliqué par les auteurs de [101, 107], une telle méthode rend possible la construction de permutations aléatoires très efficace à l'aide d'uniquement quelques tours de Feistel et les dimensions d'entrée/sortie des fonctions utilisées sont plus ou moins logarithmiques en la taille des mots sur lesquels la permutation opère.

Rappelons d'abord quelques propriétés de base des schémas de Feistel.

Schéma de Feistel

Pour toute fonction f définie de $\mathbb{F}_2^l$ dans $\mathbb{F}_2^m$, le tour de Feistel d'une fonction f, noté $\psi(f)$, est définie pour toute paire $(L, R) \in \mathbb{F}_2^m \times \mathbb{F}_2^l$ par $\psi(f)[L, R] = [R, L \oplus f(R)]$. Le calcul de k tours de Feistel, c'est-à-dire la fonction $\psi(f_k) \circ \ldots \circ \psi(f_1)$, est notée $\psi(f_1, \ldots, f_k)$ ou par ψ s'il n'y a pas d'ambiguïtés sur les fonctions utilisées.

Si f et g sont deux fonctions aléatoires générées indépendamment et définies de $\mathbb{F}_2^m$ dans lui même, alors il a été montré dans [101, 107] que la fonction $\psi(g, f, g, f)$ est indistinguable d'une distribution aléatoire par un observateur, même s'il a accès à la permutation inverse. Cela a pour conséquence, la construction d'une permutation d'indice ψ sur $\{0, \cdots, 2^{2m} - 1\}$ (et induit une permutation vectorielle σ sur $\mathbb{F}_2^n$ avec $\log_2(n) = 2m$), nous avons besoin simplement de générer les deux fonctions aléatoires indépendantes f et g.

Une fois que le schéma de Feistel a été réalisé pour deux fonctions f et g, il y a deux possibilités pour calculer la permutation $\sigma(y)$, $\sigma(y \oplus s)$ et éventuellement $\sigma(s)$ dans le protocole de Stern. La première consiste en le pré-calcul $\psi(i)$ pour tout $i \leq n$ et alors le stockage de la suite $(\psi(i))_{i \leq n}$ comme une représentation de σ. Dans ce cas, à chaque fois que σ doit être multipliée par un vecteur, alors on a accès à sa table de représentation n fois (une fois pour chaque bit). Cette stratégie nécessite une allocation de RAM de $n \times \lceil \log_2(n) \rceil$ bits, qui est assez grande dans un contexte de faible ressource. La seconde stratégie consiste en le calcul de $(\psi(i))_{i \leq n}$ chaque fois que l'on a besoin de déterminer l'indice du bit correspondant à i dans σ. Cette stratégie, que nous avons choisie pour notre implémentation, est plus consommatrice en temps que la précédente mais ne nécessite pas d'allocation en mémoire RAM.

Par construction, le schéma de Feistel permet seulement de constuire des permutations d'indices ψ telles que n est une puissance de 2. Pour la taille du paramètre n nous considérons pour le schéma de Stern $n = 347$ (voir section 3.2 page 37), nous ne pouvons pas utiliser directement le schéma de Feistel mais une version légèrement modifiée.

Description de l'algorithme

Dans la section 3.2 nous montrons que la taille des paramètres (de s et y) doit être d'au moins $694 = 2 \times 347$. Le schéma de permutation de Feistel a été construit pour opérer sur des vecteurs dont la taille est une puissance de 2. Soit m la valeur $\lceil \log_2(n) \rceil / 2$. Pour implémenter une permutation sur des vecteurs de n'importe quel taille n telle que $\lceil \log_2(n) \rceil$ soit pair, nous suggérons de générer

Algorithm 2 Permutation de bit pour n tel que $\lceil \log_2(n) \rceil$ soit paire

INPUT: le vecteur v à permuter, la longueur en bit n de v, la valeur $n' = 2^{\lceil \log_2(n) \rceil}$, un schéma de Luby-Rackoff $\psi(g, f, g, f)$ avec f et g défini de $\mathbb{F}_2^{\lceil \log_2(n) \rceil / 2}$ dans lui même.
OUTPUT: le vecteur $result = \sigma(v)$

1. **for** i **from** 0 **to** $n' - 1$ **do**
2. $\quad T[i] \leftarrow 0$
3.
4. **for** i **from** 0 **to** $n' - 1$ **do**
5. $\quad new\_index \leftarrow \psi(i)$
6. $\quad result[new\_index] \leftarrow v[i]$
7. $\quad T[new\_index] \leftarrow 1$
8. $j \leftarrow n$
9. **for** i **from** 0 **to** $n - 1$ **do**
10. $\quad$ **if** $(T[i] = 0)$
11. $\quad\quad$ **while** $(T[j] = 1)$ **do**
12. $\quad\quad\quad j \leftarrow j + 1$
13. $\quad\quad result[i] \leftarrow result[j]$
14. **return** $result$

FIGURE 3.16 – Permutation à l'aide du schéma de Feistel

aléatoirement deux fonctions f et g définies de $\mathbb{F}_2^m$ dans lui même et d'utiliser le schéma de Feistel de la manière décrite dans la figure 3.16.

Remarque 7

La table T nécessite d'être calculée seulement une fois pour chaque permutation σ. Une fois calculée, elle peut être utilisée pour toutes les permutations impliquant σ.

Dans l'algorithme 3.16, chaque itération de la seconde boucle calcule l'indice du bit new_index dans $result$ où est stockée la valeur du bit $v[i]$. Durant ce processus, la table T garde la trace des coordonnées de $result$ qui sont mises à jour durant ce processus. Quand la boucle est terminée, une troisième boucle est itérée pour remplir les coordonnées du bit d'index $i < n$ qui n'a pas été initialisé par la seconde boucle (qui sont celles telles que $T[i] = 0$), avec les coordonnées des bits d'index $n \leq j < n'$ qui ont été initialisés (qui sont tels que $T[j] = 1$).

Discussion sur la sécurité vis à vis des attaques SCA

Afin de contrecarrer les Menaces C et D (voir section 3.3 page 38), nous choisissons de masquer les calculs de σ et ψ. Comme σ est utilisée au plus 3 fois avant d'être regénérée, elle peut être attaquée par une SPA mais ne souffre pas de faiblesse vis à vis d'une DPA. La linéarité de σ rend facile le masquage de ce calcul : nous masquons l'entrée y à l'aide d'un masque aléatoire M ($\tilde{y} = y \oplus M$) et pour démasquer la sortie $\sigma(\tilde{y})$ nous effectuons un simple $x - or$ avec $\sigma(M)$.

Dans l'algorithme 3.16, la même fonction ψ est appliquée $2^{n'}$ fois sur des entrées connues avant d'être regénérée. Elle peut alors être attaquée par des attaques DPA. Pour les contrer, nous choisissons de masquer les variables intermédiaires qui apparaissent durant le calcul de ψ et d'appliquer la méthode REC pour traiter le problème de correction du masque quand les fonctions f et g sont

utilisées. A chaque fois, de nouvelles fonctions f et g sont générées (qui définie alors une nouvelle fonction ψ), nous générons deux masques aléatoires $r, s \in \mathbb{F}_2^m$ et nous définissons deux nouvelles fonctions f^* et g^* telles que $f^*(x) = f(x \oplus r) \oplus s$ et $g^*(x) = g(x \oplus s) \oplus r$. Nous décrivons la version classique et sécurisée du calcul de ψ dans la figure 3.17.

Discussion de la complexité

La fonction ψ doit être regénérée à chaque exécution de l'algorithme 3.16. Ceci nécessite la génération de $2 \times m \times 2^m$ bits aléatoires pour définir les fonctions f et g. Le processus non sûr de calcul de ψ nécessite 4 accès en table et 4 additions de m bits. La version sécurisée nécessite le pré-calcul (en RAM) de nouvelles tables lookup pour f^* et g^* (de complexité $\mathcal{O}(2^m)$) et 8 additions de vecteurs de m bits de plus (4 pour la correction du masque et 2×2 pour l'étape de masquage/démasquage de la paire entrée/sortie) comparé au coût de calcul de la version non sécurisée.

L'algorithme 3.16 nécessite deux variables locales de n' bits : T et $result$. Il utilise n' fois la fonction ψ et exécute deux boucles nécessitant respectivement $2 \times n'$ et $4 \times n'$ opérations binaires.

Ce qui donne, pour le calcul de σ :

$$14n' = 8 \times n' + 2 \times n' + 4 \times n' \text{ opérations binaires}$$

et la génération de $2 \times m \times 2^m$ bits aléatoires pour le calcul de ψ sans contre-mesure contre les SCA.

Dans le mode sûr contre les SCA, le calcul de σ nécessite $22n' = (8+8) \times n' + 2 \times n' + 4 \times n'$ opérations binaires et la génération de $2 \times m \times 2^m$ bits aléatoires pour f et g, un vecteur de n bits M pour masquer l'entreé y de σ et un vecteur de $2m$ bits (m_r, m_l) pour masquer toutes les entrées de ψ. Pour rendre le démasquage final de $\sigma(y \oplus M)$ possible, le vecteur $\sigma(M)$ doit aussi être calculé, ce qui ajoute $22n'$ opérations binaires (remarquons que le masque (m_r, m_l) n'a pas besoin d'être regénéré pour protéger le calcul de ψ pour $\sigma(M)$).

Finalement, nous avons pour le processus en mode sécurisé, environ $44n'$ (*i.e.* $44 \times 2^{\lceil \log_2(n) \rceil}$) opérations élémentaires et la génération de $n + 2 \times m \times 2^m + 2m$ bits aléatoires (*i.e.* $n + \lceil \log_2(n) \rceil \times 2^{\lceil \log_2(n) \rceil / 2} + \lceil \log_2(n) \rceil$).

Exemple 1

*Pour le choix de paramètres fait dans la section 3.2 page 37 (*i.e. $n = 694$*), nous avons $m = \lceil \log_2(n) \rceil / 2 = \lceil \log_2(694) \rceil / 2 = 5$ et $n' = 2^{\lceil \log_2(n) \rceil} = 2^{10} = 1024$. Dans un tel cas, le calcul de σ sans aucune sécurité nécessite environ $14 \times 10^3 \equiv 14 * 1024$ opérations élémentaires et la génération de $320 = 10 \times 32$ bits aléatoires. Pour une méthode sécurisée contre les attaques par canaux cachés, cela nécessite environ $45 \times 10^3 \equiv 44 * 1024$ opérations élémentaires et la génération de $994 = 694 + 10 \times 32 + 10$ bits aléatoires.*

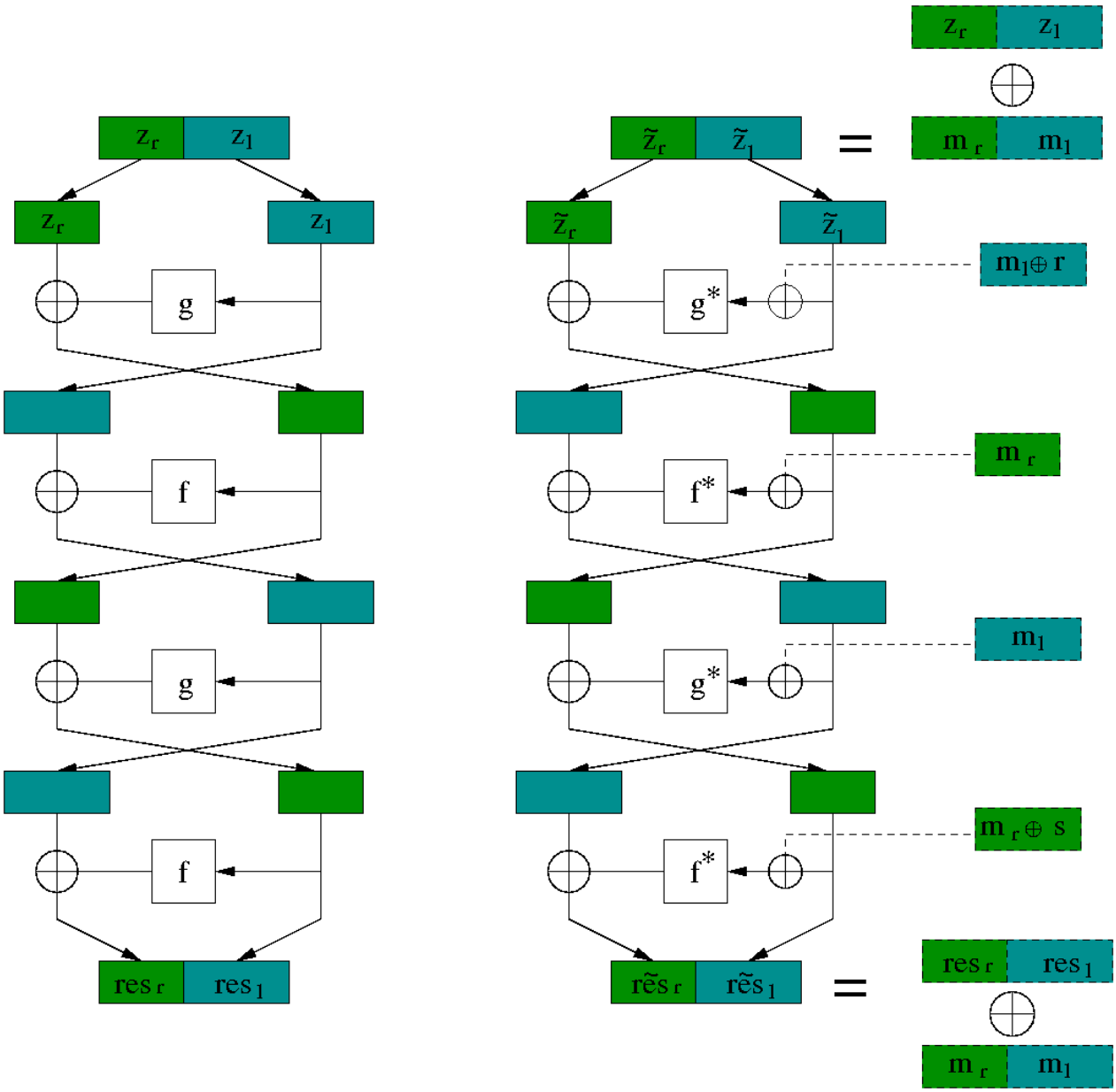


FIGURE 3.17 – Permutation à l'aide du schéma de Feistel - versions non-sûre et sûre.

Exemple 2

Pour $n = 512$ (qui constitue le choix de paramètres effectué dans la section 3.5), le calcul de σ sans aucune sécurité nécessite environ $7 \times 10^3 \equiv 14 * 512$ opérations élémentaires et la génération de $224 = 4 \times 2^4 + 5 \times 2^5$ bits aléatoires (remarquons que dans ce cas, $\log_2(512) = 9$ est impair, les fonctions f et g ne peuvent avoir les mêmes dimensions et nous choisissons f de $\mathbb{F}_2^4$ dans $\mathbb{F}_2^5$ et g de $\mathbb{F}_2^5$ dans $\mathbb{F}_2^4$). Dans le mode sécurisé contre les SCA, cela nécessite environ $22 \times 10^3 \equiv 44 * 512$ opérations élémentaires et la génération d'environ 450 bits aléatoires.

3.4.4 Générateur pseudo-aléatoire

Nous avons besoin d'un générateur pseudo-aléatoire pour construire la graine du code ainsi que de la permutation. De nos jours, la plupart des générateurs pseudo-aléatoires utilisés dans le commerce sont basés soit sur des systèmes de chiffrement à flot soit par blocs. Les implémentations matérielles d'algorithme de chiffrement à flot sont souvent plus rapides que ceux basés sur de systèmes de chiffrement par blocs.

Toutefois, les schémas à flot ne sont disponibles que dans certains dispositifs (souvent disponibles dans les cartes à puce), tandis que les algorithmes par blocs tels que les algorithmes DES ou AES sont systématiquement implémentés en hardware. Nous considérons que le générateur pseudo-aléatoire (Pseudo Random Number Generator (PRNG)), que nous utiliserons pour générer les graines n'est pas biaisé et sûr contre les attaques DPA et SPA. Dans [105], les auteurs proposent un PRNG basé sur un algorithme de chiffrement par blocs sûr contre les attaques par canaux cachés, nous utiliseront cette proposition là.

3.5 Implémentation

3.5.1 Pseudo-code

Dans cette section, nous présenterons le pseudo-code de l'étape d'engagement (commitment step) du schéma de Stern.

```
/*** Initialisation Step ***/
alea_vector_p (vector_y); //creation of the random y of weight p
// creation of the two functions f and g for Luby-Rackoff
alea_function (function_f,dim_f_input,dim_f_output);
alea_function (function_g,dim_f_output,dim_f_input);

/*** Computation of the commitments ***/ result_prod_H_y =
Product(matrix_H,vector_y); // product Hy^t (c_1)
Permute(vector_y,vector_permut_y); // permutation of y (c_2)
for ( i=0 ; i < n ; i++ )
    vector_y_xor_s [i] = vector_y [i] ^ vector_s [i]; // compute y x-or s
vector_permut_y_s = Permute (vector_y_xor_s); // permutation of y x-or s (c_3)


/*** Computation of the hash values ***/
// compute hash to obtain c1
c_1 = SHA-256 (fucntion_f||function_g||result_prod_H_y,...);
// compute hash to obtain c_2
c_2 = SHA-256 (vector_permut_y,...);
// compute hash to obtain c_3
c_3 = SHA-256 (vector_permut_y_s,...);
```

FIGURE 3.18 – Pseudo-code

3.5.2 Résultats expérimentaux

Résultats implémentation en C

Nous présentons ici les résultats obtenus par une implémentation faites sur un processeur Athlon cadencé à 1.3 GHz en collaboration avec Bertrand Zanzotto. Nous distinguons trois implémentations : l'une faite avec DES, une avec T-DES et enfin une avec AES.

Le produit matrice-vecteur est *sécurisé* et réalisé sur une architecture 32 bits. Il faut compter trois fois plus de temps sur une architecture 8 bits. Les permutations ne sont pas sécurisées, elles sont faites en fonction des bits à 1 dans le vecteur à permuter. La fonction de hachage a un taux de 1/4. Le temps est mesuré en ms pour $n = 512$.

1. Matrice aléatoire
2. Matrice quasi-cyclique $(I|A)$ de poids $n/4$
3. Matrice quasi-cyclique $(I|A)$ de poids $n/9$

DES	1	2	3
hachage	2,021		
Produit matrice/vecteur	1,097	0,453	0,305
Permutations	1,277		
Générateur	0,086		
Total pour un tour	4,674	4,367	3,888
Total, 56 tours	233	218	194

FIGURE 3.19 – Résultats expérimentaux DES

T-DES	1	2	3
hachage	5,815		
Produit matrice/vecteur	1,093	0,457	0,294
Permutations	1,279		
Générateur	0,148		
Total	8,530	7,900	7,735
Total, 56 tours	426	395	386

FIGURE 3.20 – Résultats expérimentaux T-DES

AES	1	2	3
hachage	0,931		
Produit matrice/vecteur	1,100	0,451	0,295
Permutations	1,150		
Générateur	0,100		
Total	3,483	2,827	2,678
Total, 56 tours	174	141	133

FIGURE 3.21 – Résultats expérimentaux AES

Concernant les permutations, la construction $fgfg$ de Pieprzyk a montré une faiblesse au niveau statistique : certains bits ont beaucoup plus de chances de se retrouver permutés au même endroit. Les fonctions Ψ^4 et Ψ^8 de Feistel semblent ne pas souffrir de cette faiblesse.

Résultats implémentation sur carte à puce

Nous avons réalisé, en collaboration avec Emmanuel Prouff, l'implémentation du schéma d'identification de Stern avec des matrices doublements circulantes pour $l = 256$ (*i.e.* $n = 512$). On utilise une 8051-architecture **sans crypto-processeur** ni implémentation hardware de SHA-256 et avec un CPU cadencé à 8 MHz. Nous donnons ici les temps obtenus :

Temps et nb de cycles pour 1 tour	Temps (ms)
PRNG (vecteur y, fonction f et g)	16.7
Produit matrice-vecteur	22
Permutations	28.2
Fonction de hachage (SHA-256)	85
Total : Identification (35 tours)	5316.5

On obtient une identification en $5,5$ secondes et une signature en 22 secondes et pour une sécurité de 2^{85} opérations binaires. Le coût de communication est d'environ 40-kbits pour l'identification et environ 140-kbits pour la signature. On peut remarquer que les performances (en temps) peuvent être sérieusement améliorées en utilisant une version hardware de SHA-256 au lieu d'une implémentation software.

Cette implémentation n'inclut pas les contre-mesures. D'après l'étude de la version sécurisée faite dans la section 3.4, le surcoût temps/mémoire attendu est le suivant :

- Produit : $+100\%$ (temps et mémoire) ;
- Permutation : $+300\%$ (temps) et $+200\%$ (mémoire) ;
- Fonction de hachage : pas d'augmentation du temps de calcul par contre $+100\%$ du coût en mémoire.

La valeur de ce surcoût pour une version sécurisée $(\times 3)$ est vraiment faible comparée à une sécurisation de AES où le surcoût est de $\times 10$.

3.5.3 Une implémentation optimisée de la signature

En raison du coût de communication, nous présentons ici une méthode améliorée d'implémentation du schéma de Stern utilisé en signature. Le paradigme de Fiat-Shamir [46] (voir aussi chapitre 2 page 19 de cette thèse) permet de convertir un schéma d'identification à divulgation nulle de connaissance en un schéma de signature en générant tous les engagements pour un certain niveau de sécurité et ensuite en les hachant afin d'obtenir une série de défis.

Dans ce mode, la signature est constituée de la suite des engagements ainsi que des réponses aux défis induits par les engagements. Contrairement à l'identification, la signature nécessite la génération de la suite **entière** d'engagements qui peut être importante.

En fait, plutôt que de signer l'ensemble des suites d'engagements qui pourrait être de 50 kbits et nécessiterait beaucoup d'efforts de stockage, nous proposons d'utiliser une version légèrement modifiée de cette construction. Définissons la suite $(s_1, \cdots, s_r)$ d'engagements pour r tours. Au lieu d'utiliser la génération précédente des défis, nous proposons d'utiliser la suite suivante de hachés (pour h une fonction de hachage) : $h_{i+i} = h(h_i|s_i)$, Pour $1 \leq i \leq r+1,\ h_1 = h(s_1)$ et où $||$ correspond à la concaténation. En procédant de cette manière, la production de la version finale de hachage utilisée pour les défis se fait de façon séquentielle et n'a pas besoin de plus de mémoire que celle que l'allocation nécessite pour un tour classique du protocole. On doit générer deux fois les engagements mais cette méthode permet de diminuer la quantité de mémoire de stockage

nécessaire. La signature est alors faite de façon séquentielle tour après tour. Cette façon de procéder réduit considérablement la quantité de mémoire nécessaire, ce qui permet d'abaisser le coût du protocole.

3.6 Conclusion

Nous avons décrit dans le présent chapitre, la première implémentation du protocole de Stern sur carte à puce (en fait, cela constitue aussi, plus généralement, le premier cryptosystème basé sur des codes correcteurs d'erreurs implémenté sur carte à puce avec les ressources habituelles). Pour un niveau de sécurité satisfaisant, la taille de la clé publique est de seulement 694 bits en utilisant une représentation quasi-cyclique de la matrice. Les matrices doublement circulantes réalisent un bon compromis entre des matrices complètement aléatoires et des matrices fortement structurées.

Dans ce cas, les opérations sont en effet très simples à réaliser et peuvent être facilement mises en œuvre dans le matériel. En outre, le fait que le protocole effectue essentiellement des opérations linéaires rend facile la protection contre les attaques par canaux cachés.

Ainsi, nous pensons que ce protocole est une nouvelle option pour réaliser une identification forte et rapide sur carte à puce. De plus, nous pensons que l'utilisation d'un co-processeur d'algèbre linéaire dédié devrait améliorer sensiblement les performances de notre implémentation.

Des travaux futurs envisagent des attaques d'injection de fautes (ce qui était au-delà de la portée du présent chapitre) et la mise en oeuvre d'autres variantes du protocole de Stern qui peuvent avoir d'autres avantages (voir [50]).

Chapitre 4

Schéma de signature de Kabatianskii, Krouk et Smeets

Ce chapitre a donné lieu à l'article suivant [30] :

On Kabatianskii-Krouk-Smeets Signatures
avec Ayoub Otmani et Damien Vergnaud
(International Workshop on the Arithmetic of Finite Fields, WAIFI 2007)
Springer, Lect. Notes Comput. Sci. vol. 4547, 2007, p. 237-251

G. Kabastianskii, E. Krouk et B.J.M. Smeets ont proposé en 1997 un schéma de signature basé sur des codes correcteurs aléatoires. Dans ce chapitre, nous allons examiner la sécurité et l'efficacité de leur proposition. Nous montrerons qu'un attaquant passif qui peut intercepter quelques signatures peut récupérer la clé privée. Nous donnerons précisément le nombre de signatures nécessaires pour atteindre cet objectif. Ceci nous permet de prouver que tous les systèmes donnés dans le papier original peuvent être brisés avec au plus 20 signatures. Nous améliorerons l'efficacité de ces protocoles en donnant d'abord des paramètres qui permettent de signer 40 messages, et ensuite, en décrivant une façon d'étendre ces schémas de *few-time* signatures en des schémas de *multi-time* signatures. Enfin, nous donnerons un moyen de réduire les tailles des clés à l'aide de matrices quasi-circulantes.

4.1 Introduction

En 1978, McEliece (voir [86] et section 2.1.1 page 19 de cette thèse) a proposé le premier système de chiffrement à clé public basé sur des codes correcteurs d'erreurs. Son idée est de d'abord choisir un code particulier dont on connaît un algorithme de décodage efficace, et ensuite de masquer sa structure pour qu'*il est l'air* d'un code linéaire tiré aléatoirement. La description du code originel sert alors de clé privée, tandis que la description du code modifié constitue la clé publique. Plusieurs propositions ont été faites pour modifier le schéma originel de McEliece, mais malheureusement la plupart d'entre elles sont inefficaces ou non sûres. Toutefois, le tout premier schéma, qui utilise des codes de Goppa, n'a jamais été cassé (pour des paramètres appropriés).

En 1986, Niederreiter (voir [96] et section 2.1.2 page 20 de cette thèse) a proposé un autre schéma utilisant un code linéaire. Le schéma de McEliece utilise la matrice génératrice d'un code

tandis que le schéma de Niederreiter utilise la matrice de parité. Ces deux schémas ont été prouvés équivalents [75] en termes de sécurité pour les mêmes paramètres [1].

Comparés à d'autres systèmes cryptographiques à clé publique qui impliquent une exponentiation modulaire, ces systèmes présentent l'avantage [2] d'avoir une grande vitesse de chiffrement et de déchiffrement. Cependant, ils souffrent d'un problème de taille de la clé publique. En 2005, Gaborit ([49] et voir section 2.2.2 page 24) a proposé une méthode permettant de réduire grandement sa taille (elle est quasi-linéaire).

Les signatures numériques sont d'importantes primitives cryptographiques pour fournir l'authentification dans un monde électronique. Elles permettent à un signataire détenteur d'une clé secrète de signer un message de telle sorte que toute personne ayant accès à la clé publique correspondante soit capable de vérifier l'authenticité du message. Parallèlement aux efforts faits pour construire un bon système de chiffrement à clé publique basé sur les codes correcteurs d'erreurs, plusieurs tentatives ont été proposées pour la conception de systèmes de signature basés sur cette même primitive. Malheureusement, la plupart des protocoles proposés se sont révélés précaires (voir l'article [44] dans lequel on peut trouver des références pour plus de détails).

Il est bien connu que toute permutation à trappe permet de concevoir des signatures numériques à l'aide de la capacité unique du propriétaire de la clé publique d'inverser cette permutation. La fameuse approche *Full Domain Hash* (FDH) ne peut seulement être utilisée que pour signer des messages dont les hachés appartiennent aux images de la permutation à trappe. Par conséquent, un schéma de signature basé sur l'approche FDH et utilisant des codes correcteurs d'erreurs nécessite de parvenir à un décodage total. En 2001, Courtois, Finiasz et Sendrier ([39] et voir section 2.2.1 page 21) ont présenté un schéma de signature pratique dérivé d'une technique permettant le décodage complet des codes de Goppa (pour des paramètres bien choisis).

À Crypto'93, Stern ([129] et voir section 2.2.2 page 22) a proposé un schéma d'identification et de signature basé sur le problème de décodage par syndrome. Dans ce schéma tous les utilisateurs partagent une matrice aléatoire (vue comme matrice de parité d'un code) et chaque utilisateur choisit secrètement un vecteur $\vec{v}$ de poids de Hamming donné juste en dessous de la distance minimale du code. La clé publique d'identification est le syndrome correspondant. À l'aide d'un protocole zero-knowledge interactif, chaque utilisateur peut s'authentifier en prouvant qu'il connaît le vecteur secret $\vec{v}$ sans le révéler.

Une version duale du schéma d'identification de Stern qui utilise une matrice génératrice du code a été proposé par Véron [136]. Ces deux protocoles peuvent donner des signatures en utilisant l'heuristique de Fiat-Shamir.

Finalement, Kabastianskii, Krouk et Smeets [60] ont proposé, en 1998, un schéma de signature basé sur des codes aléatoires. Ils ont exploité le fait que pour tout code linéaire l'ensemble de ces syndromes corrigibles contient un sous-espace linéaire de relativement grande dimension. Kabatianskii *et al.* concluent leur article en demandant une analyse de la sécurité de leur schéma. Ce chapitre propose cette étude.

Organisation du chapitre Le chapitre se décompose de la manière suivante : nous présenterons tout d'abord le schéma KKS décrit proposé par Kabatianskii *et al.* (de KKS-1 à KKS-4).

1. Le schéma originel de Niederreiter utilisait des familles de codes tels que par exemple les codes de Reed-Solomon généralisés et les deux schémas sont équivalents si on remplace ces codes par des codes de Goppa.

2. Pour les mêmes paramètres, le système de Niederreiter révèle certains avantages, par exemple, la taille de la clé publique et le nombre d'opérations pour chiffrer sont plus réduits.

Nous présenterons ensuite l'analyse de sécurité dans le cadre d'une attaque à message connu. En particulier, nous montrerons qu'un attaquant passif, qui peut seulement récupérer des signatures, est capable de retrouver la clé secrète de signature. Pour des valeurs de paramètres concrètes, seulement 20 signatures sont nécessaires pour révéler une grande partie de la clé secrète. Dans un troisième temps, nous donnerons un moyen d'étendre ce schéma de *few-times* signature en un schéma de *multi-time* signature. Nous étudierons ensuite les tailles de clés des schémas de signature de KKS et proposerons un moyen de les réduire. Enfin, nous concluerons sur l'effectivité du nouveau schéma.

4.2 Comment signer avec des codes correcteurs d'erreurs ?

Nous avons vu dans le chapitre 2 page 19 que l'on peut en général dériver un schéma de signature d'un schéma de chiffrement. Lorsque l'on essaye d'appliquer ce schéma de construction au cryptosystème de McEliece (voir section 2.1.1 page 19) on se heurte à un obstacle. En effet, le cryptosystème McEliece consiste à transformer un message en mot d'un code de Goppa et à y ajouter une erreur décodable. La trappe T qui permet de décoder un nombre d'erreurs donné peut ainsi s'appliquer à tous les messages chiffrés par la fonction de chiffrement associée. À part pour un code parfait où tous les mots de l'espace seront décodables, rien ne garantit que T pourra s'appliquer à n'importe quel haché. De plus, dans la pratique, la probabilité qu'un haché aléatoire soit décodable est même très petite : avec les paramètres d'origine $[1024, 524, 101]$ pour le code de Goppa, cette probabilité est même de 2^{-216}.

L'idée du schéma de Courtois, Finiasz et Sendrier (voir section 2.2.1 page 21) est de prendre une famille de codes denses.

L'idée du schéma de Kabatianski, Krouk et Smeets est un petit peu différente car elle propose un décodage total. Décrivons maintenant cette méthode.

Soit k un entier et supposons que $\mathbb{F}_q^k$ est l'ensemble des messages à signer. Pour signer un message m à l'aide du cryptosystème de Niederreiter, on définit une matrice $k \times n$ de parité H représentant un code $\mathscr{C}[n, k, d]$. Le message m peut être n'importe quel mot, pour signer il doit exister z tel que $Hz^T = m$. Ceci n'est pas toujours possible car l'algorithme de décodage peut seulement décoder les éléments de l'ensemble des syndromes corrigibles qui est différent de $\mathbb{F}_q^r$. Nous avons donc besoin de trouver une application $\chi : \mathbb{F}_q^k \to H\mathscr{M}_{n,t}$, et alors décoder $\chi(m)$ pour produire $z \in \mathscr{M}_{n,t}$ vérifiant $\chi(m) = Hz^T$.

Kabatianskii *et al.* [60] ont présenté une technique pour produire des schémas de signature basés sur des codes en utilisant des *codes linéaires quelconques.* Ce qui veut dire qu'il n'est pas nécessaire d'avoir un algorithme de décodage pour signer les messages. L'idée est de définir directement une application *secrète* $f : \mathbb{F}_q^k \to \mathscr{M}_{n,t}$ dans le but de générer automatiquement une signature $f(m)$ pour $m \in \mathbb{F}_q^k$. Le signataire envoie alors $(m, f(m))$. Il publie aussi une application $\chi : \mathbb{F}_q^k \to H\mathscr{M}_{n,t}$ qui sera utilisée dans l'étape de vérification qui est définie pour tout $m \in \mathbb{F}_q^k$ de la manière suivante $\chi(m) \stackrel{\text{def}}{=} Hf(m)^T$. Le récepteur vérifie la validité du message signé (m, z) en testant que :

$$\mathsf{wt}(z) = t \text{ et } \chi(m) = Hz^T. \tag{4.1}$$

La partie la plus importante des schémas de *signature sans décoder* est la construction de la fonction f. D'un point de vue pratique, la description de f doit être meilleure qu'une énumération

des images de f pour lesquelles on aurait besoin de $q^k \log_2(\binom{n}{t}(q-1)^t)$ bits de stockage. Il en est de même pour la fonction χ.

Ainsi, en choisissant une application linéaire aléatoire on fait un bon choix en terme d'usage pratique mais un mauvais en terme de sécurité. Du point de vue de la sécurité, il est nécessaire que la matrice publique H et l'application publique χ *ne donnent aucune information* sur l'application secrète f. Si cette propriété est garantie alors la sécurité du schéma est équivalente à celle du cryptosystème de Niederreiter sur laquelle le système est construit. Par ailleurs, pour retrouver f à partir de H (et χ), un attaquant doit résoudre $\chi(m) = Hz^T$ pour un $m \in \mathbb{F}_q^k$ donné, ce qui constitue une instance du problème du décodage par syndrome.

De plus, on peut remarquer que la seule propriété nécessaire du code $\mathscr{C}$ réside dans la difficulté à résoudre l'équation (4.1). En d'autres termes, t doit être assez grand ou à la rigueur le code $\mathscr{C}$ doit être un code *aléatoire* avec une distance minimale suffisamment grande. La proposition suivante donnée dans [60] estime la distance minimale d'un code linéaire aléatoire généré par une *matrice de parité* aléatoire.

Proposition 4.2.1

La probabilité $\mathrm{Prob}\{d(\mathscr{C}) \geq d\}$ *qu'une matrice* $r \times n$ *de parité aléatoire* $[I_r|M]$ *sur* $\mathbb{F}_q$ *définisse un code* $\mathscr{C}$ *avec une distance minimale* $d(\mathscr{C})$ *supérieure ou égale à* d *vérifie l'inégalité suivante :*

$$\mathrm{Prob}\{d(\mathscr{C}) \geq d\} \geq 1 - q^{-r+nh_q(\frac{d-1}{n})},$$

où $h_q(x) = x\log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x)$.

4.3 Le schéma

Tout d'abord nous allons décrire le schéma de base puis par la suite nous donnerons des améliorations.

4.3.1 Protocole KKS-1

Soit $\mathscr{V}$ un code q-aire $(n, n-r)$ avec une distance de Hamming minimale $d(\mathscr{V}) > 2t$ et soit $\mathscr{C}$ un code (n', k') équidistant[3] avec pour distance de Hamming minimale $d(\mathscr{C}) = t$, où $n \geq n' = \frac{q^{k'}-1}{q-1}$ et $d(\mathscr{C}) = t = q^{k'-1}$. Soit $H = [h_1, \ldots, h_n]$ une matrice $(r \times n)$ de parité du code $\mathscr{V}$ et soit G une matrice $(k' \times n')$ génératrice du code $\mathscr{C}$. Définissons alors une matrice $(r \times k')$ $F = H(J)G^T$, où J est un sous-ensemble de l'ensemble $\{1, \ldots, n\}$ de cardinal n', et $H(J)$ est une sous-matrice de la matrice M consistant à prendre uniquement les colonnes h_j où $j \in J$. Rappelons que $S_t(H) = \{He^T : e \in \mathbb{F}_q^{n,t}\}$ est l'ensemble des syndromes correspondants aux erreurs de poids t.

Nous avons alors la proposition suivante :

Proposition 4.3.1

$Fx^T \in S_t(H)\ \forall x \in \mathbb{F}_q^{k'} \backslash \{0\}$

Définition 4.3.1 (*KKS-1*)

Soit $\mathscr{U}[n', k, t]$ un code équidistant *(tous les mots sont de poids t) sur $\mathbb{F}(q)$ tel que $n' \leq n$ défini par une matrice génératrice $G = [g_{i,j}]$. On sait [85] que pour un tel code : $n' = \frac{q^k-1}{q-1}$ et $t = q^{k-1}$.*

Malheureusement KKS-1 n'est pas praticable car il nécessite un code trop grand. Par exemple dans le cas binaire ($q = 2$) et avec le paradigme FDH, k doit être d'au moins 160. Ce qui implique que $n \geq n' = 2^{160} - 1$. Il est nécessaire de remplacer le code équidistant par un autre pour lequel $\omega_1 \neq \omega_2$. Deux solutions sont proposées dans [60] :

- soit en considérant le dual d'un code BCH binaire
- soit en considérant un code linéaire aléatoire grâce aux propositions 4.3.2 et 4.3.3.

Proposition 4.3.2 (*Borne de Carlitz-Uchiyama*)

Soit $\mathscr{U}$ le code dual d'un BCH binaire de longueur $n' = 2^m - 1$ et de distance construite $\delta = 2s + 1$. Alors pour n'importe quel $u \in \mathscr{U}$:

$$\left| wt(u) - \frac{n'+1}{2} \right| \leq (s-1)\sqrt{n'+1}.$$

Nous pouvons définir un schéma de signature de la manière suivante. Nous avons deux matrices publiques : H et F. L'ensemble J et la matrice G constituent la clé privée. Le signataire signe un message $m \in \mathbb{F}_q^{k'} \backslash \{0\}$ par la signature e définie de la manière suivante :

$$He^T = Fm^T : wt(e) = t \tag{4.2}$$

où le vecteur e vaut mG sur les positions de J et 0 en dehors de J.

3. Un code équidistant est un code dans lequel tous les mots sont de même poids

Problème 1

Un tel protocole de signature n'est pas résistant aux attaques par homomorphismes, c'est-à-dire qu'après l'observation de deux messages signés $[m_1, e_1]$ et $[m_2, e_2]$ un attaquant peut créer un mot faux, mais valide : $[m_1 + m_2, e_1 + e_2]$. Pour éviter ce genre d'attaques nous allons faire la même chose que ce qui est fait d'habitude dans de tels cas (la signature RSA par exemple).

Considérons une *bonne* (c'est-à-dire suffisamment non linéaire, facile à évaluer, difficile à inverser, sans collision, etc ...) fonction de hachage $f : M = \mathbb{F}_q^k \to \mathbb{F}_q^{k'} \backslash \{0\}$, et modifions la définition de la signature précédente en :

$$He^T = Ff(m)^T : wt(e) = t \tag{4.3}$$

Pour un message donné m le signataire évalue la signature $e = f(m)G$. L'opposant,qui veut résoudre l'équation (4.3), doit échouer à cause de la difficulté de décoder un code linéaire arbitraire. Trouver les fonctions trappes, c'est-à-dire l'ensemble J et la matrice G semble être aussi difficile que de décoder le code $\mathscr{V}$ car chaque colonne de la matrice publique F est une combinaison linéaire d'exactement t colonnes de la matrice H (venant des positions de J) et trouver cette combinaison linéaire revient à décoder $\mathscr{V}$ pour des valeurs particulières du syndrome.

Problème 2

Considérons des codes binaires. Nous voulons que le coût d'une attaque pour un opposant ne soit pas inférieur à 2^{80} opérations. Cela implique que le nombre des valeurs des signatures possibles doit être d'au moins 2^{80}. D'où $k' \geq 80$ et $n \geq n' = \frac{2^{80}-1}{80-1}$ mais ces valeurs sont trop grandes pour être applicables en pratique.

4.3.2 Protocole KKS-2

Définition 4.3.2 (*KKS-2*)

La construction se déroule de la manière suivante

1. *Le signataire choisit aléatoirement :*
   - *une matrice D de taille $r \times (n - r)$,*
   - *une matrice A non singulière de taille $k \times k$,*
   - *un sous ensemble $J \subset \{1, \ldots, n\}$ de taille $n' = 2^m - 1$.*
2. *Il construit une matrice de parité $H = [I_r|D]$ binaire de taille $r \times n$.*
3. *Il choisit une matrice génératrice G du code dual d'un code BCH binaire de longueur n' et de distance construite $2s + 1$.*
4. *Il forme la matrice $F = H(J)(AG)^T$ (pour masquer la matrice G).*

La clé publique est constituée des matrices H et F, et la clé secrète est l'ensemble J et la matrice A.

Les valeurs numériques suivantes viennent de [60] :

$$m = 10, s = 6, k = ms = 60, n' = 2^{10} - 1 = 1023,$$

$$\omega_1 = 352, \omega_2 = 672, r = 2808, n = 3000.$$

La distance minimale du code $\mathscr{C}$ est d'au moins 1024 avec une probabilité $\geq 1 - 10^{-9}$.

Comme pour le nombre de bits à stocker, nous remarquons que la clé privée est constituée de $nh_2(\frac{n'}{n})$ bits pour décrire J et k^2 bits pour la matrice A.

Pour la clé publique nous devons stocker $r(n-r)$ bits pour la matrice H et rk bits pour la matrice F.

KKS-2 peut être encore amélioré en prenant un code linéaire aléatoire pour $\mathscr{U}$. Grâce à la proposition 4.3.3, il est possible de connaître la probabilité qu'un code aléatoire ait tous ses mots de poids non nul à l'intérieur d'un intervalle donné.

Proposition 4.3.3

Soit $\mathscr{U}$ un code défini par une matrice génératrice sous forme systématique aléatoire de taille $k \times n'$. Soit δ un réel tel que $0 < \delta < 1$. Alors la probabilité que les mots (non nuls) d'un code linéaire binaire aléatoire aient leur poids dans l'intervalle $[\omega_1; \omega_2]$ (avec $\omega_2 = \delta + \omega_1$) est d'au moins :

$$\text{Prob}\{\omega_1 \leq wt(\mathscr{U}) \leq \omega_2\} \geq 1 - 2^{-2(n'-k)+n'h_2(\frac{\omega_1-1}{n'})+n'h_2(\frac{n'-\omega_2+1}{n'})}.$$

Dans le but d'améliorer les paramètres de ce schéma nous allons remplacer le code équidistant par n'importe quel code $\mathscr{C}$ (n', k'), dont les mots non nuls de $\mathscr{C}$ ont un poids compris entre ω_1 et ω_2. Cela mène à la modification suivante du schéma de signature :

$$He^T = Ff(m)^T : \omega_1 \leq wt(e) \leq \omega_2 \tag{4.4}$$

Remarquons que la condition $d(\mathscr{V}) \geq 2\omega_2$, qui garantit l'unicité de la signature, c'est-à-dire, une solution de l'équation (4.4), n'est pas aussi importante pour le schéma présent que pour les schémas de McEliece et Niederreiter (un message peut avoir plusieurs signatures, mais un texte chiffré doit correspondre à un et un seul message). Nous avons simplement besoin que la résolution de l'équation (4.4) soit difficile, c'est-à-dire de décoder un code $\mathscr{V}$ sachant que le nombre de ces erreurs appartient à un intervalle de la forme $[\omega_1, \omega_2]$.

Par un moyen utilisé dans la preuve de l'existence de la borne de Gilbert-Varshamov (voir section 1.2.2 page 16) nous avons :

Proposition 4.3.1

La probabilité qu'un code binaire aléatoire $\mathscr{V}$ $(n, n-r)$ ait une distance minimale $d_{\mathscr{V}} \geq d$ est :

$$\text{Prob}\{d(\mathscr{V}) \geq d\} \geq (1 - 2^{-(n-r)}(\sum_{i=0}^{d+1} \binom{n}{i}) \geq 1 - 2^{-r+nh_2(\frac{d-1}{n})},$$

où $h_2(x) = -x\log_2(x) - (1-x)log_2(1-x)$.

Maintenant la description du système est la suivante. Le signataire choisit aléatoirement : une matrice $r \times n$ H, une matrice non singulière $k' \times k'$ A, et un n'-sous ensemble $J \subset \{1, \ldots, n\}$. Il prend une matrice de parité G d'un code BCH binaire de distance $2s+1$ comme matrice génératrice et forme une matrice $r \times k'$ F de la forme $F = H(J)(AG)^T$ (c'est-à-dire que AG est une matrice génératrice arbitraire du code $\mathscr{C}$). La clé publique de ce système est composée des matrices H et F ce qui coûte $r(n+k')$ bits, et la clé privée est relativement petite, précisément : $\log_2 \binom{n}{n'} nh(\frac{n}{n'}) \leq n' \log_2(n)$ bits pour décrire l'ensemble J et $(k')^2$ bits pour la matrice A. Le signataire évalue la signature e comme $f(m)AG$ en les positions de l'ensemble J et 0 en dehors. Tous les arguments précédents concernant le schéma basé sur des codes équidistants sont valides pour les modifications considérées. Illustrons cela par quelques exemples :

Exemple 1

Prenons $l = 8$ *et* $s = 6$. *Alors le nombre de signatures possibles est suffisamment grand* $(2^{48} - 1), \omega_1 = 48, \omega_2 = 208$. *Choisissons une matrice binaire aléatoire* (927×1200) H. *Avec une probabilité* $p \geq 1 - 10^{-9}$ *le code* $\mathscr{V}$ *correspondant a une distance minimale* $d_{\mathscr{V}} > 256$. *La clé publique de ce système est de 253000 bits .*

Cela coûte plus de 2^{50} *opérations (environ* $\frac{\binom{n}{t}}{\binom{r}{t}} \approx 2^{nh(\frac{t}{n}) - rh(\frac{t}{r})} = 2^{51}$ *opérations élémentaires à l'aide du meilleur des algorithmes connus voir section 2.3 page 26) pour résoudre l'équation (4.3.1) avec un nombre d'erreurs appartenant à l'intervalle* $[48, 208]$.

Remarque

En fait pour stocker une matrice aléatoire de la forme $(k \times n)$, *on stocke la matrice duale de la forme* $(n - k \times n)$ *que nous mettons sous forme systématique, on obtient un stockage de* $(n - k) \times k$ *bits. Ici pour stocker une matrice* (927×1200) *nous avons besoin de* $(1200 - 927) \times 927$ *soit* 253071 *bits.*

Exemple 2

Prenons $l = 10$ *et* $s = 6$. *Alors le nombre de signatures possibles est suffisamment grand* $(2^{60} - 1), \omega_1 = 352, \omega_2 = 672$. *Choisissons une matrice binaire aléatoire* (2808×3000) H. *Avec une probabilité* $p \geq 1 - 10^{-9}$ *le code* $\mathscr{V}$ *correspondant a une distance minimale* $d_{\mathscr{V}} > 1024$. *La clé publique de ce système est de 538000 bits. Cela coûte bien plus de* 2^{54} *opérations pour résoudre l'équation (4.3.1) pour un nombre d'erreurs appartenant à l'intervalle* $[352, 672]$.

On choisit l, s pour déterminer $\mathscr{C}$ et r, n pour déterminer H (il s'agit de nos degrés de liberté pour avoir un compromis : complexité / taille des clés) puis on a :

- l tel que $2^l - 1$ soit la longueur du code BCH,
- s tel que $2s + 1$ soit la distance du code BCH,
- r représente le nombre de lignes de la matrice H,
- n représente le nombre de colonnes de la matrice H,
- k' représente la dimension du code BCH,
- n' représente la longueur du code BCH,
- ω_1 poids minimum des mots du code,
- ω_2 poids maximum des mots du code.

k'	n'	ω_1	ω_2
sl	$2^l - 1$	$\omega_2 - 2 \times (s-1)\sqrt{n'+1}$	$(s-1)\sqrt{n'+1} + \frac{n'+1}{2}$

compléxité	clé publique
$\frac{\binom{n}{t}}{\binom{r}{t}} \approx 2^{nh_2(\frac{t}{n}) - rh_2(\frac{t}{r})}$	$r(n + k')$

FIGURE 4.1 – Paramètres KKS-2

Maintenant nous allons améliorer ces schémas en utilisant une description, un petit peu plus compliquée, qui nous permet d'utiliser des codes $\mathscr{C}$ avec une petite distance minimale. Car pour l'instant les codes $\mathscr{C}$ que nous utilisons sont composés de mots de poids appartenants à un intervalle dont la borne supérieure est assez grande.

l	s	r	n	k'	n'	ω_1	ω_2	compléxité	clé publique
8	6	927	1200	48	255	48	208	2^{50}	253 000
10	6	2808	3000	60	1023	352	672	2^{54}	538 000
10	6	3120	3450	60	1023	352	672	2^{81}	10 951 200

FIGURE 4.2 – Valeurs des paramètres KKS-2

Schéma	Version	k	n'	ω_1	ω_2	r	n	$nh_2(\frac{\omega_1}{n}) - rh_2(\frac{\omega_1}{r})$
KKS-2		60	1023	352	672	2808	3000	36
KKS-3	#1	60	280	50	230	990	1250	17
	#2	160	1000	90	110	1100	2000	80
KKS-4		48	180	96	96	765	1100	53

FIGURE 4.3 – Paramètres KKS-2-3-4.

4.3.3 Protocole KKS-3

Définition 4.3.3 (*KKS-3*)

Le signataire suit les mêmes étapes que pour KKS-2 mais choisit une matrice sous forme systématique aléatoire $G = [I_k|B]$ de taille $k \times n'$.

La clé publique est constituée des matrices H et F, et la clé secrète est l'ensemble J et la matrice B.

Dans ce cas, la taille de la clé privée de KKS-3 est de $nh_2(\frac{n'}{n})$ bits pour J et $k(n'-k)$ bits pour la matrice B. La taille de la clé publique reste inchangée.

Nous donnons les valeurs numériques suivantes : $k = 160$ et $n' = 900$. Le code $\mathscr{U}$ généré par G a tous ces mots de poids entre $\omega_1 = 90$ et $\omega_2 = 110$ avec une probabilité $\geq 1 - 2^{-749}$. Le signataire choisit une matrice de parité aléatoire H du code $\mathscr{C}$ de taille (1100×2000). Alors $d(\mathscr{C}) > 220$ avec une probabilité $\geq 1 - 2^{-100}$.

La figure 4.3 page 79 donne les paramètres donnés par les auteurs de [60] pour KKS-3 #1. La version #2 correspond quant à elle à nos paramètres choisis afin de respecter le niveau de sécurité actuel.

Avec ce qui a été fait dans [60] où la sécurité du système a été évaluée avec $t = \frac{\omega_1+\omega_2}{2}$, nous donnons une valeur pour ω_1 telle que l'attaque de décodage par ensemble d'informations ([22] et section 2.3.1 page 28 de cette thèse) ne puisse s'appliquer.

Finalement, les auteurs proposent une modification qui permet de construire une signature KKS à partir de codes qui contiennent des mots de petits poids. L'idée est de prendre pour le code $\mathscr{U}$ le produit direct de P codes $\mathscr{U}_i[n^*, k^*, \omega_1^*]$ sur $\mathbb{F}(q)$ dont les mots de codes ont un poids $\leq \omega_2^*$. Bien sur, le code $\mathscr{U}$ a des mots de codes de petits poids. Nous devons trouver un moyen d'éliminer ces mots de petits poids. Supposons que chaque code $\mathscr{U}_i$ est défini par une matrice génératrice G_i. Le corps fini $\mathbb{F}(q^{k^*})$ est considéré comme un espace vectoriel de dimension k^* sur $\mathbb{F}(q)$ défini par une base fixée. Nous noterons M_β la matrice représentant l'application linéaire $x \mapsto x\beta$ avec $\beta \in \mathbb{F}(q^{k^*})$. Soit Q un entier non négatif. Pour n'importe quel $u = (u_1, \ldots, u_Q) \in \mathbb{F}(q^{k^*})^Q$ nous définissons pour tout $x \in \mathbb{F}(q^{k^*})$

$$u(x) = u_1 + u_2 x + \cdots + u_Q x^{Q-1}.$$

Soient $\beta_1, \ldots, \beta_P$ des éléments non nuls de $\mathbb{F}(q^{k^*})$, $A_1, \ldots, A_P$ des matrices de taille $k^* \times k^*$ non singulières et $J_1, \ldots, J_P$ des sous ensembles disjoints de $\{1, \ldots, n\}$ de cardinal n^* avec $Pn^* < n$. L'application $f : \mathbb{F}(q^{k^*})^Q \mapsto \mathbb{F}(q)^n$ envoie n'importe quel u sur z qui vaut 0 sur les positions $\{1, \ldots, n\} \setminus \cup_{i=1}^{P} J_i$ et vaut $u(\beta_i)A_iG_i$ sur les positions de J_i. La signature reste $Hz^T = Fu^T$ où la matrice publique est $F = (F_1, \ldots, F_Q)$ et où F_j est la matrice $r \times k^*$ avec :

$$F_j \stackrel{\text{def}}{=} \sum_{i=1}^{P} H(J_i) \left(M_{\beta_i^{j-1}} A_i G_i \right)^T.$$

L'ensemble des messages est maintenant $\mathbb{F}(q^{k^*})^{Q-1}$ et pour signer $(u_2, \ldots, u_Q)$, le signataire choisit $u_1 \in \mathbb{F}(q^{k^*})$ tel que $u_1 \notin \{\sum_{i=2}^{Q} u_i \beta_j^{i-1} \ : \ j = 1, \ldots, P\}$ (ce qui est toujours possible quand $P \leq q^{k^*}$) et on a $P\omega_1^* \leq \mathsf{wt}(f(u)) \leq P\omega_2^*$. Nous appelerons ce schéma : KKS-4.

4.3.4 Protocole KKS-4

Définition 4.3.4 (*KKS-4*)

Le signataire choisit P codes $\mathscr{U}_i[n^*, k^*, \omega_1]$ sur $\mathbb{F}(q)$ dont les mots de code ont un poids $\leq \omega_2$, des éléments non nuls $\beta_1, \ldots, \beta_P \in \mathbb{F}(q^{k^*})$, des matrices de taille $k^* \times k^*$ non singulières $A_1, \ldots, A_P$ et des sous ensembles disjoints $J_1, \ldots, J_P$ de $\{1, \ldots, n\}$. Ceci constitue la clé secrète. Il forme une matrice F comme décrit précédemment ce qui constitue avec la matrice H la clé publique.

Remarquons que dans ce schéma modifié

$$n' = Pn^*, k = Pk^*, \omega_1 = P\omega_1^* \text{ et } \omega_2 = P\omega_2^*.$$

Les auteurs ont donné les valeurs suivantes :

$$Q = 14, P = 12, k^* = 4, n^* = 15.$$

Les codes $\mathscr{U}_1, \ldots, \mathscr{U}_p$ sont tous égaux au code équidistant $\mathscr{U}[15, 4, 8]$. $\mathscr{C}$ est un code aléatoire de longueur $n = 1100$ et de dimension 335. La distance minimale $d(\mathscr{C}) \geq 193$ avec une probabilité d'au moins $1 - 10^{-9}$. La figure 4.3 page 79 récapitule ces valeurs.

4.4 Étude de la sécurité

La sécurité des schémas de signature de KKS réside dans la qualité du choix de la fonction f. Nous avons vu que si f ne donne pas d'information alors *a priori* KKS est aussi sûr que le cryptosystème de Niederreiter. En fait, à chaque utilisation de f, on obtient beaucoup d'information.

En effet, chaque signature z revèle $|\mathsf{supp}(z)|$ positions du secret J. De plus, un attaquant peut exploiter cela pour retrouver J. Lorsqu'un attaquant connaît J, il peut trouver la matrice secrète G juste en résolvant un système linéaire $F = H(J)G^T$ où G représente l'inconnue car avec une grande probabilité $H(J)$ est de rang plein puisqu'on a généralement $r > n'$.

Dans le cas de KKS-4, l'attaquant doit aussi trouver $A_1, \ldots, A_P$ et les éléments $\beta_1, \ldots, \beta_P$ qui sont des racines des polynômes $U(X) = \sum_{i=1}^{Q} u_i X^i$ définis par chaque message $(u_1, \ldots, u_Q)$. Nous ne traitons pas ce cas dans ce chapitre et préférons nous concentrer sur le premier schéma.

Supposons que l'attaquant possède $\ell \geq 1$ signatures (m_i, z_i) Chaque signature z_i peut être vue comme le résultat d'un oracle de choix indépendant, et ℓ signatures donnent $\left|\cup_{i=1}^{\ell} \mathsf{supp}(z_i)\right|$ éléments de J. Définissons les variables aléatoires $U_\ell \stackrel{\text{def}}{=} \left|\cup_{i=1}^{\ell} \mathsf{supp}(z_i)\right|$. Alors ℓ signatures révèlent en moyenne $\mathsf{E}[U_\ell]$ positions de J où $\mathsf{E}[X]$ est l'espérance de la variable aléatoire X. Pour chaque position $j \in \{1, \ldots, n'\}$, posons χ_j la variable aléatoire de Bernoulli définie par $\chi_j = 1$ si $j \in \cup_{i=1}^{\ell} \mathsf{supp}(z_i)$ et par $\chi_j = 0$ sinon. Par définition, $U_\ell = \sum_{j=1}^{n'} \chi_j$ et on a alors $\mathsf{E}[U_\ell] = \sum_{j=1}^{n'} \mathsf{E}[\chi_j]$. De plus, $\mathrm{Prob}\{\chi_j = 0\} = \prod_{i=1}^{\ell} \mathrm{Prob}\{j \notin \mathsf{supp}(z_i)\} = \mathrm{Prob}\{j \notin \mathsf{supp}(z_1)\}^\ell$ comme les signatures z_i sont considérées comme des variables aléatoires indépendantes.

Soit N_w le nombre de mots de code de $\mathscr{U}$ de poids w et pour tout $j \in \{1, \ldots, n'\}$ posons $N_w(j)$ le nombre de mots de code $u \in \mathscr{U}$ de poids w tels que $u_j = 0$. On a alors :

$$\begin{aligned} \mathrm{Prob}\{j \notin \mathsf{supp}(z_i)\} &= \sum_{t=\omega_1}^{\omega_2} \mathrm{Prob}\{j \notin \mathsf{supp}(z_i), \mathsf{wt}(z_i) = t\} \\ &= \sum_{t=\omega_1}^{\omega_2} \mathrm{Prob}\{j \notin \mathsf{supp}(z_i) | \mathsf{wt}(z_i) = t\} \, \mathrm{Prob}\{\mathsf{wt}(z_i) = t\}. \end{aligned}$$

Mais, $\mathrm{Prob}\{\mathsf{wt}(z_i) = t\} = \frac{N_t}{q^k}$ et $\mathrm{Prob}\{j \notin \mathsf{supp}(z_i) | \mathsf{wt}(z_i) = t\} = \frac{N_t(j)}{N_t}$.

Un moyen de simplifier ces calculs est d'approximer la dernière égalité en posant :

$$\frac{N_t(j)}{N_t} = \mathrm{Prob}\, j \notin \mathsf{supp}(z_i) | \mathsf{wt}(z_i) = t \simeq \frac{\binom{n-1}{t}}{\binom{n}{t}} = (1 - \frac{t}{n'}).$$

Ce qui donne le résultat suivant :

$$\mathrm{Prob}\, j \notin \mathsf{supp}(z_i) \;=\; 1 - \frac{q^{-k}}{n'} \sum_{t=\omega_1}^{\omega_2} t N_t.$$

Ceci implique que : $\mathrm{Prob}\, \chi_j = 0 = \left(1 - \frac{q^{-k}}{n'} \sum_{t=\omega_1}^{\omega_2} t N_t\right)^\ell$. Donc, en posant

$$p_\ell \stackrel{\text{def}}{=} 1 - \left(1 - \frac{q^{-k}}{n'} \sum_{t=\omega_1}^{\omega_2} t N_t\right)^\ell$$

on a prouvé la proposition suivante.

Proposition 4.4.1

Le nombre $\mathsf{E}[U_\ell]$ d'éléments de J révélés par ℓ signatures est :

$$\mathsf{E}[U_\ell] \;=\; n' p_\ell.$$

On peut trouver la formule explicite pour $\mathrm{Prob}\{U_\ell = j\}$ pour tout entier j dans [11].

Il est nécessaire de connaître la distribution des poids du code $\mathscr{U}$ si l'on veut utiliser la proposition 4.4.1. Cette propriété est en général difficile à calculer pour un code linéaire arbitraire. Toutefois, un attaquant peut facilement effectuer l'attaque suivante. Comme $\mathsf{E}[U_\ell] = n' p_\ell$ positions

de J en moyenne sont connues avec ℓ signatures, l'attaquant doit chercher les $(n' - n'p_\ell)$ éléments manquants de J parmis les $(n - n'p_\ell)$ positions restantes.

À chaque étape, il résout k systèmes de r équations linéaires avec n' inconnues et arrête dès que le système admet une solution. Le coût de cette attaque est donc $\mathcal{O}(kn'^\omega \binom{n-n'p_\ell}{n'-n'p_\ell})$ où n'^ω représente le coût de résolution d'un système linéaire avec n' inconnues (naïvement $\omega = 3$). Dans le but d'appliquer la proposition 4.4.1 à n'importe quel code linéaire, nous avons besoin de donner des inégalités sur p_ℓ. Cela peut être fait en remarquant que :

$$1 - \left(1 - \frac{\omega_1}{n'}\right)^\ell \leq p_\ell \leq 1 - \left(1 - \frac{\omega_2}{n'}\right)^\ell.$$

Posons $a \stackrel{\text{def}}{=} n'(1 - \frac{\omega_1}{n'})^\ell$ et $b \stackrel{\text{def}}{=} n - n' + n'(1 - \frac{\omega_1}{n'})^\ell$. Nous avons alors $\binom{n-n'p_\ell}{n'-n'p_\ell} \leq \binom{b}{a}$. Nous notons dans la figure 4.4 le nombre d'opérations pour une attaque pour différentes valeurs de ℓ. Ces résultats numériques ont été obtenus par l'inégalité (4.5) et en posant $\omega = 3$:

$$\binom{b}{a} \leq \frac{1}{\sqrt{2\pi a(1 - \frac{a}{b})}} 2^{bh_2(\frac{a}{b})}. \tag{4.5}$$

Nous voyons par exemple que nous avons besoin de seulement $\ell = 13$ signatures pour casser KKS-2 au moyen de (2^{78}) opérations, et $\ell = 20$ signatures pour casser la version #1 de KKS-3 à l'aide de (2^{77}) opérations.

	$\ell = 15$	$\ell = 14$	$\ell = 13$	$\ell = 12$	$\ell = 11$	$\ell = 10$
KKS-2	2^{56}	2^{65}	2^{78}	2^{97}	2^{122}	2^{160}

	$\ell = 23$	$\ell = 22$	$\ell = 21$	$\ell = 20$	$\ell = 19$	$\ell = 18$
KKS-3 (version #1)	2^{58}	2^{64}	2^{70}	2^{77}	2^{86}	2^{96}

	$\ell = 6$	$\ell = 5$	$\ell = 4$	$\ell = 3$	$\ell = 2$
KKS-4	2^{46}	2^{63}	2^{96}	2^{155}	2^{261}

FIGURE 4.4 – Nombre d'opérations pour retrouver J pour différentes valeurs de ℓ pour les schémas proposés dans [60].

Ces résultats numériques sont confirmés par la proposition 4.4.2 qui donne une très bonne approximation du nombre maximum de signatures autorisées sans compromettre la sécurité du schéma KKS.

Proposition 4.4.2

Supposons n suffisamment grand et n' tels que $2n' \leq n$ et tels que le paramètre de sécurité λ défini par $\frac{80 - \omega \log_2 n' - \log_2 k}{n - n'}$ vérifie $0 < \lambda < 1$. Soit γ le plus petit réel > 0 tel que $h_2(\gamma) = \lambda$. Définissons ℓ_γ de la manière suivante :

$$\ell_\gamma \stackrel{\text{def}}{=} \frac{\ln \frac{\gamma}{1-\gamma} + \ln(\frac{n}{n'} - 1)}{\ln(1 - \frac{\omega_2}{n'})}.$$

La clé privée du système de KKS peut être retrouvée avec ℓ signatures si $\ell \geq \ell_\gamma$.

Preuve Soit $\ell \leq \ell_\gamma$ et soit $\delta_\ell \stackrel{\text{def}}{=} \frac{n'-n'p_\ell}{n-n'p_\ell}$. On a $\delta_\ell \leq \frac{n'}{n} \leq \frac{1}{2}$. On sait que $\binom{n-n'p_\ell}{n'-n'p_\ell} = 2^{(n-n'p_\ell)h_2(\delta_\ell)+o(n)}$. On peut vérifier que si $\ell \leq \ell_\gamma$ alors $\delta_\ell \geq \gamma$ et donc $h_2(\delta_\ell) \geq h_2(\gamma)$. Comme $p_\ell \leq 1-(1-\frac{\omega_2}{n'})^\ell$, on peut écrire :

$$\begin{aligned}(n-n'p_\ell)\,h_2(\delta_\ell) &\geq \left(n-n'+n'\left(1-\frac{\omega_2}{n'}\right)^{\ell_\gamma}\right) h_2(\gamma)\\ &\geq \frac{1+\gamma}{1-\gamma}\left(80-\omega\log_2 n'-\log_2 k\right).\end{aligned}$$

Nous avons donc $kn'^{\omega}2^{(n-n'p_\ell)h_2(\delta_\ell)} \geq 2^{80}$ car $\frac{1+\gamma}{1-\gamma} \geq 1$ ($\gamma \geq 0$) et ceci termine la preuve. □

La proposition 4.4.2 donne $\ell_\gamma = 46$ signatures autorisées obtenues avec $\gamma = 0,00421$ pour nos paramètres de KKS-3 version #2. Finalement, l'inégalité (4.5) nous montre que l'on peut signer au plus 40 fois.

4.5 Extension à un schéma de multi-time signature

4.5.1 D'un schéma one-time vers un schéma multi-time

Les arbres de Merkle ont été inventés en 1979 par Merkle [92]. Le but initial était de permettre de traiter de manière efficace de nombreux schémas de one-time signature introduits par Lamport [72]. Un arbre de Merkle est une façon d'engager n messages avec une seule valeur de hachage de manière à ce que tel ou tel message exige la révélation de $\log n$ valeurs de hachage.

L'idée sous-jacente est de placer les n messages sur les feuilles d'un arbre binaire (en supposant que n est une puissance de 2 pour des raisons de simplicité), puis à calculer la valeur à chaque noeud sans feuille dans l'arbre comme étant le haché de ses deux enfants. La valeur à la racine de l'arbre est l'*engagement* des n messages. Pour révéler une valeur, l'utilisateur la publie ainsi que les valeurs de la fratrie de chaque ancêtre de celui-ci (on appelle cela le *chemin de l'authentification*). On peut facilement vérifier que la valeur a été correctement révélée par simple calcul des hachés de l'arbre et de vérifier que l'ultime valeur de hachage correspond à la racine.

Les arbres de Merkle ont été proposé pour prolonger un schéma d'une à plusieurs signatures (one-time signature vers multi-time signature). L'idée est de générer une seule fois n clés publiques, et de les placer dans un arbre de Merkle. La racine de l'arbre de Merkle devient la clé publique de la signature. Pour plus de détails, nous renvoyons le lecteur au papier originel de Merkle [92].

4.5.2 D'un schéma few-time vers un schéma multi-time

En suivant l'idée de Merkle, il est possible d'étendre un schéma de *few-time* en un schéma *multi-time* avec la même sécurité. Si le problème sous-jacent est sûr contre une ℓ-attaque à messages clairs choisis, l'idée est de mettre les n messages aux feuilles d'un arbre ℓ-aire complet.

Soit $\Sigma_\ell = (\mathsf{Setup}_\ell, \mathsf{Sign}_\ell, \mathsf{Verify}_\ell)$ un schéma de signature sûr contre les ℓ-attaques à messages clairs choisis et soit n un entier (pour des facilités d'explication, nous supposons que $n = \ell^p$ est une puissance de ℓ).

Le schéma $\Sigma_{\mathsf{multi}} = (\mathsf{Setup}_{\mathsf{multi}}, \mathsf{Sign}_{\mathsf{multi}}, \mathsf{Verify}_{\mathsf{multi}})$ est défini de la manière suivante :

– $\mathsf{Setup}_{\mathsf{multi}}$: en entrée un entier λ (le paramètre de sécurité), $\mathsf{Setup}_{\mathsf{multi}}$ appelle $(\ell^{p-1}-1)/(\ell-1)$ fois $\mathsf{Setup}_\ell(\lambda)$ pour obtenir les paires de clés :

$$(\mathsf{pk}_{i,j}, \mathsf{sk}_{i,j}) \text{ pour } j \in \{0, \ldots, p-1\} \text{ et } i \in \{1, \ldots, \ell^j\}.$$

La clé publique est la racine $\mathsf{pk}_{1,0}$ et la clé privée consiste en la concaténation des paires de clés $(\mathsf{pk}_{i,j}, \mathsf{sk}_{i,j})$. L'utilisateur doit garder un compteur en mémoire qui contient le nombre de signatures précédemment créées. Au début, le compteur est mis a zéro.

– $\mathsf{Sign}_{\mathsf{multi}}$: soit i le compteur. En entrée un message m et la clé secrète, $\mathsf{Sign}_{\mathsf{multi}}$ calcule $\sigma_0 = \mathsf{Sign}_\ell(m, \mathsf{sk}_{r_0,p-1})$ où $r_0 = \lfloor i/\ell \rfloor$ et alors récursivement $\sigma_{t+1} = \mathsf{Sign}_{\mathsf{k}}(\mathsf{pk}_{r_t,p-1-t}, \mathsf{sk}_{r_{t+1},p-2-t})$, où $r_{t+1} = \lfloor r_t/\ell \rfloor$ pour $t \in \{0, \ldots, p-2\}$. La signature de m est alors :

$$\sigma = \left(\sigma_0, \mathsf{pk}_{r_0,p-1}, \sigma_1, \mathsf{pk}_{r_1,p-2}, \ldots, \sigma_{p-2}, \mathsf{pk}_{r_{p-2},1}\right).$$

– $\mathsf{Verify}_{\mathsf{multi}}$: en entrée un message m, une signature

$$\sigma = \left(\sigma_0, \mathsf{pk}_{r_0,p-1}, \sigma_1, \mathsf{pk}_{r_1,p-2}, \ldots, \sigma_{p-2}, \mathsf{pk}_{r_{p-2},1}\right),$$

et une clé publique $\mathsf{pk}_{0,1}$, $\mathsf{Verify}_{\mathsf{multi}}$ accepte la signature σ si et seulement si :

$$\mathsf{Verify}_\ell(\mathsf{pk}_{r_t,p-1-t}, \mathsf{sk}_{r_{t+1},p-2-t}, \sigma_{t+1}) = 1 \text{ pour } t \in \{1, \ldots, p-2\} \text{ et}$$
$$\mathsf{Sign}_\ell(m, \mathsf{sk}_{r_0,p-1}, \sigma_0) = 1.$$

La sécurité de Σ_{multi} contre une n-attaque à messages clairs choisis est trivialement équivalente à une des Σ_ℓ contre une ℓ-attaque à messages clairs choisis. Dans la construction de Σ_{multi}, un équilibre peut être établi entre la taille de la signature et la taille de la clé publique.

Cette construction permet de transformer les systèmes de signature en KKS classique en schéma *multi-time*, mais les signatures sont malheureusement très longues, et afin de rendre le système plus pratique, il est nécessaire de réduire la taille des paramètres publiques.

4.6 Réduction des paramètres

Dans cette section, nous étudions les tailles de clé des schémas KKS et la manière de les réduire. Nous nous limitons au cas binaire.

Tout d'abord, nous rappelons la taille des paramètres des différents schémas KKS.

La clé privée est de $nh_2(\frac{n'}{n}) + k^2$ bits dans le cas de KKS-2 et $nh_2(\frac{n'}{n}) + k(n' - k)$ bits dans le cas de KKS-3. Comme pour la clé publique, les deux schémas nécessitent de garder en mémoire $r(n - r) + rk$ bits. La figure 4.5 donnant les valeurs numériques obtenues montre à quel point ces solutions ne sont pas réalisables en pratique.

Cependant, nous pouvons améliorer la taille de la clé publique en reprenant des idées de P. Gaborit et Girault dans [50] (voir aussi section 2.2.2 page 24). En effet, H peut être partagé par tous les utilisateurs. Ainsi, chaque utilisateur n'a besoin que de fournir son propre F.

Schéma		Clé Publique			Clé Privée
		Commun (H)	Personnel (F)	Total clé publique	
KKS-2		539136	168480	707616	6378
KKS-3	version #1	257400	59400	316800	14160
	version #2	990000	176000	1166000	120385

FIGURE 4.5 – Tailles des clés en bit

L'idée repose sur l'utilisation de matrices presque quasi-cycliques. Ces matrices sont complètement déterminées si la première ligne est connue.

Définition 4.6.1 (*Matrice presque quasi-cyclique*)

Une matrice M de taille $r \times n$ avec $r \geq 2$ et $n \geq 2$ est une matrice presque quasi-cyclique si chaque vecteur ligne est la rotation vers la droite de la ligne précédente :

$$M = \begin{pmatrix} c_1 & c_2 & \dots & & & \dots & c_n \\ c_n & c_1 & c_2 & & & & c_{n-1} \\ c_{n-1} & c_n & c_1 & c_2 & & & c_{n-2} \\ \vdots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ c_{n-r+2} & \dots & c_{n-1} & c_n & c_1 & c_2 \dots & c_{n-r+1} \end{pmatrix}.$$

Notre nouveau système repose sur l'utilisation de matrices presque quasi-cycliques aléatoires au lieu de purs codes linéaires aléatoires. Nous modifions les schémas KKS en remplaçant chaque matrice par une matrice aléatoire systématique presque quasi-cyclique. En d'autres termes, la matrice de parité $H = (I_r|D)$ est choisie de telle sorte que D soit presque quasi-cyclique.

La clé publique commune a pour taille $(n - r)$ bits et la clé publique personnelle fait rk bits. Pour KKS-2 la clé privée ne change pas, mais pour KKS-3 la matrice sous forme systématique aléatoire G peut être aussi une matrice quasi-cyclique aléatoire sous forme systématique. Dans ce cas, la clé privée a $nh_2(\frac{n'}{n}) + (n' - k)$ bits. Lorsque nous appliquons à notre proposition de la version KKS-3 (numéro 2), cette méthode donne 176000 bits pour la clé publique personnelle, 900 bits pour la clé publique commune et seulement 2726 bits pour la clé privée. La longueur de la signature est d'environ $\log_2 \binom{n}{\omega_2} = \log_2 \binom{2000}{110} = nh_2(\frac{11}{200}) = 615$ bits.

Schéma	HORS $(k,t) = (16, 23657)$	HORS $(k,t) = (20, 14766)$	HORS $(k,t) = (32, 8364)$	**KKS-3**
clé publique	23833000	147836000	854000	**176900**
clé privée	3785120	2362560	1338240	**2726**
signature	2560	3200	5120	**615**

FIGURE 4.6 – Comparaison de l'efficacité de KKS et de HORS pour une sécurité heuristique de 80-bits

4.7 Conclusion

Dans [104], Perrig a proposé un schéma de one-time signature appelé *BiBa* (pour *Bins and Balls*), dont les principaux avantages sont la vérification rapide et la petite taille des signatures. En 2002, Reyzin et Reyzin [116] ont présenté un schéma plus simple de one time, un schéma appelé *HORS*, qui maintient les avantages de *BiBa* et élimine son principal inconvénient, à savoir la génération coûteuse de signatures. Comme les schémas étudiés dans ce chapitre, le schéma *HORS* ne peut pas être utilisé pour signer beaucoup de messages, sa sécurité décroît quand le nombre de signature croît. Par conséquent, il est utile de comparer son efficacité à celle des schémas KKS.

Dans la figure 4.6, nous comparons (pour la même sécurité heuristique) les performances de KKS-3 version #2 avec nos paramètres et le schéma *HORS* implémenté avec la même fonction à sens unique et autorisons le même nombre de signatures (à savoir 40).

Cette comparaison donne un avantage très favorable à KKS. Les tailles de clés de KKS sont beaucoup plus petites et permettent une utilisation sur un canal de faible largeur de bande. Toutefois, la génération de signatures de HORS est plus rapide car elle ne requiert que l'évaluation d'une fonction de hachage. En outre, la sécurité des HORS se réduit à une entité ad hoc (mais bien définie) de sécurité sur l'hypothèse sous-jacente des fonctions de hachage, alors que KKS a été proposé, sans aucune analyse de sécurité.

Dans ce chapitre, nous avons quantifié les variations de sécurité des schémas de KKS contre un attaquant passif qui peut intercepter des signatures.

Chapitre 5

Schéma de signature basé sur l'identité prouvé sûr

Ce chapitre a donné lieu aux publications suivantes [28, 27, 29] :

Identity-based identification and signature schemes using correcting codes
avec Philippe Gaborit et Marc Girault
(International Workshop on Coding and Cryptography, WCC 2007)
pages 69-78, editors : Augot, D., Sendrier, N., and Tillich, J.-P.

On Code-based Identity-based Identification and Signature Schemes
avec Philippe Gaborit, David Galindo et Marc Girault
(soumis à IEEE IT, version longue avec preuve de sécurité)

Identity-based Identification and Signature Schemes using Error Correcting Codes
avec Philippe Gaborit et Marc Girault
(chapitre du livre Identity-Based Cryptography à paraître)
Editors Marc Joye and Gregory Neven

Dans un article datant de 1984, Shamir [125] propose un schéma de cryptographie à clé publique novateur :

... we introduce a novel type of cryptographic scheme, which enables any pair of users to communicate securely and to verify each other's signatures without exchanging private or public keys, without keeping key directories, and without using the services of a third party. The scheme assumes the existence of trusted key generation centers, whose sole purpose is to give each user a personalized smartcard when he first joins the network. The information embedded in this card enables the user to sign and encrypt the messages he sends and to decrypt and verify the messages he receives in a totally independent way, regardless of the identity of the other party. Previously issued cards do not have to be updated when new users join the network, and the various centers do not have to coordinate their activities or even to keep a user list. The centers can be closed after all the cards are issued, and the network can continue to function in a completely decentralized way for an indefinite period . The scheme is ideal for closed groups of users such as the executives of a multinational company or the branches of a large bank ...

Il propose de considérer l'espace des clés publiques comme étant l'espace des chaînes de caractères quelconques, en employant une chaîne de caractères indéniablement liée à l'utilisateur (par exemple un courriel du type bob@domaine.org) comme clé publique, la clé privée étant dérivée de la clé publique par un tiers de confiance. Shamir donne à son système le nom de schéma basé sur l'identité (identity-based scheme), présente un système de signature (identity-based signature ou IBS) utilisant RSA et conjecture que des systèmes de chiffrement pourront être conçus. La figure 5 donne une représentation de ce système.

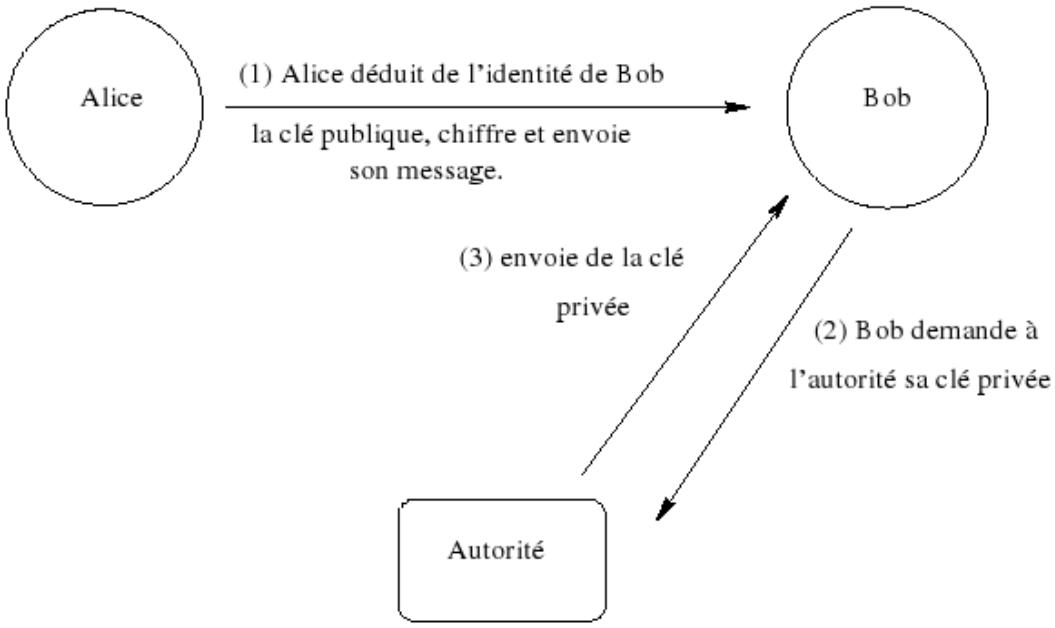


FIGURE 5.1 – Schéma basé sur l'identité

Les étapes 2 et 3 ne sont effectuées qu'une seule fois.

La recherche d'un procédé de chiffrement basé sur l'identité (identity-based encryption ou IBE) est restée ouverte jusqu'en 2001. Il semble qu'entre 1984 et 2001 aucun système de chiffrement soumis par la communauté cryptographique internationale n'ait été suffisamment sûr et réellement efficace.

En 2001, Boneh et Franklin [48] résolurent le problème en développant un IBE à la fois sûr et performant exploitant les couplages sur les courbes elliptiques. Depuis la recherche s'est orientée vers la production et l'optimisation de systèmes dérivés de ce dernier.

5.1 Notations et définitions

Introduisons tout d'abord quelques notations. Si x est un vecteur, alors $|x|$ désigne sa taille, de même, si S est un ensemble alors $|S|$ désigne son cardinal. Si $k \in \mathbb{N}$ alors 1^k représente une suite de k uns. Si S est un ensemble alors $s \xleftarrow{R} S$ représente l'opération de prendre de manière uniformément aléatoire un élément s de S.

Sans autre indication, les algorithmes sont considérés comme des algorithmes probabilistes en temps polynomial (PPT). On écrit $\mathcal{A}(x, y, \ldots)$ pour indiquer que $\mathcal{A}$ est un algorithme qui prend en entrée $x, y, \ldots$ et par $z \leftarrow \mathcal{A}(x, y, \ldots)$ on représente l'exécution de $\mathcal{A}$ avec en entrée $(x, y, \ldots)$ et en produisant z en sortie. Nous notons $\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \ldots)$ pour indiquer que $\mathcal{A}$ est un algorithme avec en entrée $x, y, \ldots$ et accède aux oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$ et par $z \leftarrow \mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \ldots)$ nous représentons l'exécution de $\mathcal{A}$ avec en entrée $(x, y, \ldots)$ et l'accès aux oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$, et en produisant z en sortie.

• PROUVEURS ET VÉRIFIEURS : Un algorithme interactif est un algorithme PPT qui prend en entrée un message M_{in} (on notera ε si on initialise le protocole) et l'état d'information St produit un message sortant M_{out} et met à jour l'état St. L'état initial contient les entrées initiales de l'algorithme. Nous disons que $\mathcal{A}$ accepte si $M_{\text{out}} = \mathsf{acc}$ et rejette si $M_{\text{out}} = \mathsf{rej}$. Une interaction entre un prouveur P et un ieur V, tous deux considérés comme des algorithmes interactifs, se termine lorsque V soit accepte soit rejette. L'expression

$$(C, d) \leftarrow \mathbf{Run}[\mathsf{P}(p_1, \ldots) \leftrightarrow \mathsf{V}(v_1, \ldots)]$$

représente le fait que P et V sont engagés dans une interaction avec en entrée $p_1, \ldots$ et $v_1, \ldots$ respectivement, prenant une conversation C et une décision booléenne d,

- 1 signifiant que V accepte,
- 0 signifiant qu'il rejette.

• SCHÉMAS D'IDENTIFICATION STANDARDS. Un schéma d'identification standard $\mathcal{S} = (\mathsf{Kg}, \mathsf{P}, \mathsf{V})$ consiste en trois algorithmes PPT :

Génération de clé : algorithme Kg prenant en entrée un paramètre de sécurité k et retournant une clé secrète SK et une clé publique correspondante PK. On utilisera la notation $(SK, PK) \leftarrow \mathsf{Kg}(1^k)$.

Dans un protocole d'identification interactif, le prouveur P possède un état initial SK, tandis que l'état initial du vérifieur est PK. Nous devons avoir pour tout $k \in \mathbb{N}$ et toute paire de clés valide (PK, SK), la sortie produite par V à chaque interaction entre V (avec en entrée PK) et P (avec en entrée SK) est acc avec probabilité 1.

La notion de sécurité d'usurpation d'identité (*impersonalisation*) sous les attaques passives peut être trouvée dans [45].

• SIGNATURES STANDARDS. Un schéma de signature standard $\mathcal{S} = (\mathsf{KG}, \mathsf{Sign}, \mathsf{Vfy})$ consiste en trois algorithmes PPT :

Génération de clé : algorithme KG prenant en entrée un paramètre de sécurité k et retournant une clé secrète SK et une clé publique correspondante PK. On utilisera la notation $(SK, PK) \leftarrow \mathsf{KG}(1^k)$.

Signature : algorithme Sign prenant en entrée une clé secrète SK et un message m. La sortie est une signature $sig_{SK}(m)$, ce que nous noterons $sig_{SK}(m) \leftarrow \mathsf{Sign}(SK, m)$.

Vérification : algorithme Vfy prenant en entrée une clé publique PK, un message m, et une signature $sig = sig_{SK}(m)$. La sortie est 1 si la signature est valide, ou 0 sinon. On utilisera la notation $\{0,1\} \leftarrow \mathsf{Vfy}(PK, m, sig)$ pour représenter une exécution de cet algorithme.

La notion de sécurité standard dans les schémas de signature est l'inforgeabilité contre des attaques à messages choisis adaptatifs, qui peut être trouvée dans [53].

• IDENTIFICATION BASÉE SUR L'IDENTITÉ. Un schéma d'identification basé sur l'identité $\mathcal{IBS} = (\mathsf{MKg}, \mathsf{UKg}, \overline{\mathsf{P}}, \overline{\mathsf{V}})$ consiste en 4 algorithmes PPT, comme suit :

Génération de la clé maîtresse : l'algorithme MKg prend en entrée un paramètre de sécurité k et retourne d'un côté, les paramètres publics du système mpk et, d'un autre côté, la valeur de la clé secrète maître msk, qui est seulement connue de l'entité maîtresse, qu'on notera :

$$(mpk, msk) \leftarrow \mathsf{MKG}(1^k).$$

Extraction de clé : l'algorithme UKg prend en entrée la clé secrète maître msk et une identité $\mathsf{id} \in \{0,1\}^*$, et renvoie une clé secrète $sk[\mathsf{id}]$. Nous utiliserons la notation :

$$sk[\mathsf{id}] \leftarrow \mathsf{UKg}(msk, \mathsf{id}).$$

Dans un protocole d'identification interactif, le prouveur avec pour identité id exécute l'algorithme interactif $\overline{\mathsf{P}}$ avec pour état initial $sk[id]$, et le vérifieur exécute $\overline{\mathsf{V}}$ avec pour état initial mpk, id.

• SÉCURITÉ DES SCHÉMAS D'IDENTIFICATION BASÉS SUR L'IDENTITÉ. Un schéma d'identification basé sur l'identité est dit sûr contre l'usurpation d'identité contre les attaques passives (imp-pa) si n'importe quel adversaire $\mathcal{A} = (\overline{\mathsf{CP}}, \overline{\mathsf{CV}})$, constituant un prouveur tricheur $\overline{\mathsf{CP}}$ et un vérifieur tricheur $\overline{\mathsf{CV}}$, a un avantage négligeable dans le jeu suivant :

Setup Le challenger choisit un paramètre de sécurité k et exécute l'algorithme de génération de clé maître MKg. Cela donne mpk à l'adversaire qui garde la clé secrète maître msk pour lui. Cela initialise une liste UK^{list}.

Phase 1 Les requêtes de l'adversaire sont les suivantes

- Requête de clé de l'utilisateur $\langle \mathcal{ID}_i \rangle$. Le challenger vérifie qu'il existe une entrée $(id_i, sk[id_i])$ dans la liste UK^{list}. Si cela est le cas, il retourne la clé secrète de l'utilisateur $sk[id_i]$. Sinon, il exécute l'algorithme UKg pour générer la clé privée $sk[id_i]$ correspondant à id_i. Il envoie $sk[id_i]$ à l'adversaire et inclut l'entrée $(id_i, sk[id_i])$ dans la liste UK^{list}.
- Requête de conversation $\langle \mathcal{ID}_i \rangle$. Le challenger vérifie qu'il existe une entrée $(id_i, sk[id_i])$ dans la liste UK^{list}. Si cela est le cas, il retourne la clé secrète de l'utilisateur $sk[id_i]$. Sinon, il exécute l'algorithme UKg pour générer la clé privée $sk[id_i]$ correspondant à $\mathcal{ID}_i$. Le challenger retourne (C, d) avec $(C, d) \leftarrow \mathbf{Run}[\overline{\mathsf{CP}}(sk[id_i]) \leftrightarrow \mathsf{V}(mpk, id_i)]$.

Ces requêtes peuvent être formulées de manière adaptative, ce qui signifie que chaque requête dépend de la réponse obtenue aux précédentes requêtes.

Challenge Le vérifieur tricheur $\overline{\mathsf{CV}}$ retourne son identité $id^\star$ et son état $St_{\overline{\mathsf{CV}}}$, de telle manière que la clé privée $id^\star$ n'ait pas été reçue en Phase 1.

Phase 2 Le prouveur tricheur $\overline{\mathsf{CP}}$, avec en entrée $St_{\overline{\mathsf{CV}}}$, interagit avec le vérifieur honnête avec en entrée $mpk, id^\star$. Le prouveur tricheur est autorisé à faire des requêtes avec les mêmes oracles qu'en Phase 1, excepté pour la requête $id^\star$. Finalement, $\mathcal{A}$ gagne si la sortie de V est acceptée, i.e. $d = 1$ dans $(C, d) \leftarrow \mathbf{Run}[\overline{\mathsf{CP}}(sk[id_i]) \leftrightarrow \mathsf{V}(mpk, id_i)]$.

Un tel adversaire est appelé un adversaire $\mathsf{imp\text{-}pa}$ $\mathcal{A}$, et son avantage est défini par

$$\mathsf{Adv}^{\mathsf{imp-pa}}_{\mathcal{IBI},\mathcal{A}}(1^\ell) = \Pr[d = 1].$$

• SIGNATURE BASÉE SUR L'IDENTITÉ. Un schéma de signature basé sur l'identité est $\mathcal{IBS} = (\mathsf{MKG}, \mathsf{UKg}, \overline{\mathsf{Sign}}, \overline{\mathsf{Vfy}})$ et consiste en 4 algorithmes PPT [32] comme suit :

Setup l'algorithme MKG prend en entrée un paramètre de sécurité k et retourne, d'un côté, les paramètres du système mpk et, d'un autre côté, la valeur de la clé secrète maître msk, qui est seulement connue par l'entité maître. On utilisera la notation :

$$(mpk, msk) \leftarrow \mathsf{MKG}(1^k).$$

Extraction de clé l'algorithme UKg prend en entrée un paramètre de sécurité k, la clé secrète maître msk et une identité $\mathsf{id} \in \{0,1\}^*$, et retourne un clé secrète $sk[\mathsf{id}]$. On utilisera la notation :

$$sk[\mathsf{id}] \leftarrow \mathsf{UKg}(msk, \mathsf{id}).$$

Signature l'algorithme $\overline{\mathsf{Sign}}$ prend en entrée une clé secrète $sk[\mathsf{id}]$, le paramètre public mpk, une identité, et un message m. La sortie est une signature $sig = sig_{msk}(\mathsf{id}, m)$. On utilisera la notation :

$$sig \leftarrow \overline{\mathsf{Sign}}(mpk, \mathsf{id}, sk[\mathsf{id}], m).$$

Vérification l'algorithme $\overline{\mathsf{Vfy}}$ prend en entrée mpk, un message m, une identité id et une signature sig et donne en sortie 1 si la signature est valide, et 0 sinon. Pour faire référence à une exécution de ce protocole, on utilisera la notation :

$$\{0,1\} \leftarrow \overline{\mathsf{Vfy}}(mpk, \mathsf{id}, m, sig).$$

La notion de sécurité standard pour un schéma de signature basé sur l'identité est l'inforgeabilité contre les attaques à messages identifiés choisis de manière adaptative, qu'on peut trouver dans [12].

5.2 Schéma basé sur l'identité

5.2.1 Protocole de Shamir

Ce schéma est sous-jacent au protocole de signature publié par Shamir en 1984. L'autorité crée une biclé RSA (les mêmes notations seront utilisées et à partir de maintenant, tous les calculs doivent être effectués modulo n, dans ce protocole et dans les suivants). L'autorité confie à Alice $r = I^d$, où I est l'identité d'Alice. Pour prouver qui elle est, Alice devra montrer qu'elle connaît r, sans dévoiler r.

Pour cela, lorsqu'elle envoie un message à Bob, elle choisit un aléa $x < n$ et lui envoie $t = x^e$. Bob prend au hasard un nombre c appartenant à l'intervalle $[1, e-1]$, et le communique à Alice qui lui retourne alors $y = rx^c$. Il ne reste plus à Bob qu'à vérifier que l'on a bien $y^e = It^c$.

5.2.2 Protocole de Cocks

Soit i l'ensemble d'informations représentant l'identité d'Alice. On suppose l'existence d'un procédé public h à base de fonctions de hachage tel que $(\frac{h(i)}{M}) = 1$ où $\frac{a}{b}$ représente le symbole de Legendre. On considèrera dans la suite l'expression $a = h(i)$ comme étant l'identité de Alice.

Supposons que Bob veuille transmettre un courrier à Alice, d'identité a. La valeur M est publique. On supposera de plus sans perte de généralité que a est un carré modulo M. Bob va traiter son message bit à bit.

Chiffrement : Soit x la représentation d'un bit à envoyer, $x = 1$ ou -1. Bob choisit un entier t aléatoirement tel que $(\frac{t}{M}) = x$. Il peut alors envoyer le message chiffré s correspondant à x :

$$s = (t + \frac{a}{t})(\mathrm{mod}\ M)$$

Déchiffrement : Conformément au principe des cryptosystèmes basés sur l'identité, Alice interagit avec l'autorité de manière à obtenir r vérifiant $r^2 = a(\mathrm{mod}M)$. Cette connaissance sera le secret de l'autorité et r sera la clé secrète d'Alice, fournit par l'autorité. Ainsi, comme $s + 2r = t(1 + r/t)^2(\mathrm{mod}\ M)$, Alice peut retrouver x avec :

$$(\frac{s+2r}{M}) = (\frac{t}{M}) = x.$$

Dans le cas où $-a$ est un carré modulo M, on utilise $s = t - a/t(\mathrm{mod}\ M)$.

5.3 Protocole de Cayrel-Gaborit-Girault

5.3.1 Description

Soit $\mathscr{C}$ un code linéaire q-aire t-correcteur de longueur n et de dimension k. Soit H une matrice de parité de $\mathscr{C}$. Soit $H' = QHP$ avec Q inversible et P matrice de permutation. Soit h une fonction de hachage à valeurs dans $\{0,1\}^{n-k}$. Soit id_A l'identité d'Alice. id_A est calculable par tous. De même, H' est publique. La décomposition de H' est en revanche un secret détenu par l'autorité. Nous allons décrire une méthode d'authentification basée sur l'identité : Alice cherchera à s'identifier auprès de Bob.

5.3.2 Travail Préliminaire : obtention de la clé privée

Alice doit s'authentifier de manière classique afin d'obtenir la clé privée s qui lui permettra ensuite de s'authentifier un grand nombre de fois auprès de tiers. On utilise pour cela une variante pour l'identité. Admettons que l'on connaissse l'identité id_B de Bob. Soit h une fonction de hachage à valeurs dans $\{0,1\}^{n-k}$. On cherche un moyen de trouver $s \in E_{q,n,t}$ tel que $h(id_B) = H's^T$.
Il s'agit donc de déchiffrer $h(id_B)$. Le principal problème est que $h(id_B)$ n'est pas *a priori* dans l'espace d'arrivée de $x \longrightarrow H'x^T$. C'est-à-dire que $h(id_B)$ n'est pas *a priori* dans l'espace des chiffrés par le système de Niederreiter.

Ce problème peut se résoudre par l'algorithme suivant appelé mCFS dans [41] et chapitre 2.

Soit κ le paramètre de sécurité. On obtient en sortie un couple $\{s, j\}$ tel que $h(id_B \oplus j) = H's^T$.

- $\mathsf{Gen}_{\mathsf{mCFS}}(1^\kappa)$: On choisit n, k et t en fonction de κ. On prend une matrice de parité aléatoire H d'un code de Goppa (n,k)-binaire décodant t erreurs. On choisit une matrice $(n-k) \times (n-k)$ aléatoire non singulière S, une $n \times n$ matrice de permutation aléatoire P et une fonction de hachage $h : \{0,1\}^* \longrightarrow \mathbb{F}_2^{n-k}$. La clé publique est $H' = SHP$ et la clé privée est (S, H, P).
  Posons $t = \frac{n-k}{\log_2 n}, i = 0$.
- $\mathsf{Sign}_{\mathsf{mCFS}}(m, H)$:
  1. $id_B \leftarrow$ élément aléatoire de $\{1; \ldots; 2^{n-k}\}$
  2. $x' = \mathcal{D}ecode_H\left(S^{-1}h(m|id_B)\right)$
  3. Si on ne trouve pas x' on retourne à 1
  4. en sortie on a $(id_B, x'P)$
- $\mathsf{Verify}_{\mathsf{mCFS}}(m, x', id'_B, H')$: on calcule $s' = H'x'^T$ et $s = h(m|id_B)$. La signature est valide si s et s' sont égaux.

FIGURE 5.2 – Le protocole mCFS

On peut remarquer qu'on a nécessairement s de poids t.

5.3.3 Authentification auprès de Bob.

On utilise un léger dérivé du protocole de Stern chapitre 2.2.2 page 22. On suppose dans ce protocole qu'Alice a obtenu un couple $\{s, j\}$ vérifiant : $h(id_A \oplus j) = H's^T$.

1. Alice choisit aléatoirement un mot y de n bits et une permutation σ des entiers $\{1, 2, \ldots, n\}$. Elle envoie alors à Bob c_1, c_2, c_3 et **j** tels que :

$$c_1 = h(\sigma | H'y^T);\ c_2 = h(\sigma(y));\ c_3 = h(\sigma(y \oplus s)),$$

2. Bob envoie à Alice un élément de $\{0, 1, 2\}$.
3. Trois possibilités :
   - si $b = 0$: P révèle y et σ.
   - si $b = 1$: P révèle $(y \oplus s)$ et σ.
   - si $b = 2$: P révèle $\sigma(y)$ et $\sigma(s)$.
4. Trois possibilités :
   - si $b = 0$: Bob vérifie que les c_1, c_2 reçus à l'étape 2 ont bien été calculés honnêtement.
   - si $b = 1$: Bob vérifie que les c_1, c_3 reçus à l'étape 2 ont bien été calculés honnêtement. Pour c_1 on peut remarquer que $H'y^T$ se déduit directement de $H'(y \oplus s)^T$ par :

$$H'y^T = H'(y \oplus s)^T \oplus h(id_A \oplus \mathbf{j})$$

   - si $b = 2$: Bob vérifie que les c_1, c_3 reçus à l'étape 2 ont bien été calculés honnêtement, et que le poids de $s.\sigma$ est bien égal à t.
5. Réitérer les étapes 1,2,3,4 tant que la sécurité exigée n'est pas atteinte.

FIGURE 5.3 – Authentification auprès de Bob

5.3.4 En pratique

Nous allons ici revenir sur la sécurité des protocoles *classiques*, leur applicabilité pour enfin finir par notre protocole.

Faisabilité et sécurité de Niederreiter version identité

Pour que notre schéma reste réaliste, cette partie doit respecter deux impératifs :

- rendre le calcul de $\{s, j\}$ tel que $h(id_A \oplus j) = H's^T$ difficile sans connaissance de la décomposition de H';
- faire en sorte que le nombre de *tâtonnements* pour déterminer le bon j ne soit pas trop important (afin de réduire le coût du calcul de s).

Le deuxième point incite à utiliser des codes de Goppa de longueur $n = 2^m$. En effet, pour un tel code la proportion de syndromes décodables est d'environ $1/t!$ (ce qui est une relativement bonne proportion). On doit donc également s'imposer un t assez faible.

Le processus de production de $\{s, j\}$ sera donc itéré en moyenne $t!$ fois avant de trouver un bon j. Or chaque itération oblige à calculer $D(h(id_A \oplus j))$. Le déchiffrement d'un chiffré par Niederreiter revient, pour le cas d'un code de Goppa, à :

- calculer un syndrome : $t^2m^2/2$ opérations binaires ;
- calculer un polynôme localisateur : $6t^2m$ opérations binaires ;
- calculer ses racines : $2t^2m^2$ opérations binaires.

On obtient donc un coût total pour le calcul de la clé privée d'Alice de l'ordre de :

$$t!t^2m^2(1/2 + 2 + 6/m).$$

Le coût de l'attaque par décodage à l'aide de l'algorithme de séparation des supports (*split syndrome decoding*) est lui estimé à :

$$2^{tm(1/2+o(1))}.$$

Le choix des paramètres devra être suffisamment pertinent pour concilier coût et sécurité. Bien que moins rédhibitoires, certaines grandeurs doivent également rester raisonnables : la longueur de $\{s, j\}$, le coût de la vérification et la taille de H'.

Comme s est de poids t faible, il sera intéressant de la représenter par la position de ses bits non nuls, pour :

$$s = (s_0, s_1, \ldots, s_{2^m-1})\ (2^0, 2^1, \ldots, 2^{2^m-1})$$

On utilise $(i_1, \ldots, i_t)$ tel que $(s_{i_1}, \ldots, s_{i_t}) = (1, \ldots, 1)$. $\forall l \in \{1, \ldots, t\}, i_l \leq 2^m - 1$, tout i_l se code donc sur m bits : on obtient une taille maximale de s d'environ tm bits. La vérification de $\{s, j\}$ consiste en un calcul matriciel : une matrice de taille $(n - k) \times n$ multipliée à un vecteur de taille n mais dont seulement t positions sont non nulles. D'où $(n - k)t$ calculs et dans le cas d'un code de Goppa : t^2m.
La taille de H' est $(n - k) \times n$, soit pour un code de Goppa $2^m tm$. La figure suivante résume les différents paramètres à concilier :

quoi	ordre de grandeur
coût de la signature	$t!t^2m^2(1/2 + 2 + 6/m)$
taille de la signature	tm
coût de la vérification	t^2m
coût de l'attaque	$2^{tm(1/2+o(1))}$
taille de H'	$2^m tm$

On peut par exemple prendre $t = 9$ et $m = 16$. Le coût de la signature reste alors assez raisonnable pour une sécurité de l'ordre de $2^{\frac{mt}{3}} \approx 2^{50}$. Les autres grandeurs restent dans ce contexte très acceptables.

Stern en pratique :

On a déjà fait remarquer que la sécurité de ce schéma repose essentiellement sur la sécurité du cryptosystème de Niederreiter (à condition d'itérer l'algorithme suffisamment de fois). Donc si l'on se place dans les conditions indiquées dans le paragraphe précédent, le protocole de Stern reste sûr. On remarquera à ce sujet que la transmission par Alice de j à l'étape 1 ne pose pas de problème de sécurité : pour tricher à ce niveau et se faire passer pour Alice, il faudrait arriver à obtenir par hasard de l'autorité un j_t (t pour *tricheur*) tel que la fonction de hachage connaisse une collision. Il faudrait avec l'aide du hasard $h(id_t \oplus j_t) = h(id_A \oplus j_t)$ ce qui est quasi-impossible.

Un problème peut en revanche venir des temps de calcul : H' est beaucoup plus grande avec nos paramètres qu'avec ceux initialement proposés par Stern (à savoir $n = 1024$ et $k = 512$). Toutes les multiplications par H' sont donc plus lourdes, or, on en compte en moyenne $5/3$ par tour. Pour obtenir une sécurité de l'ordre de 2^{80}, on effectue environ 150 tours. D'où environ 250 multiplications $H'y^T$ par authentification, chacune comptant 16 fois plus de multiplications de bits qu'avec les paramètres initiaux. Dans [39], une étude de l'efficacité montre qu'avec des paramètres comparables aux notres, le temps de calcul de $H'y^T$ est de l'ordre de 10^{-6} secondes (machine cadencée à 1Ghz). L'ensemble des calculs du type $H'y^T$ sur une utilisation de notre protocole prend donc environ un quart de seconde. Notre schéma reste donc parfaitement valable malgré ce surcoût de calcul.

5.3.5 Conclusion

Ce système parait donc fonctionner de manière correcte. Sans être un modèle de rapidité (à cause d'une production assez lente de s) il reste intéressant sous certaines conditions : le choix des paramètres est comme toujours primordial, et nous recommandons pour un code de Goppa :

$$t = 9; m = 16.$$

Compte tenu du fait qu'une attaque sur l'autorité serait fatale au système, on peut éventuellement demander une sécurité de l'ordre de 2^{90}. Ce nombre est atteint pour :

$$t = 10; m = 16.$$

Ce gain nécessite quelques sacrifices dans les autres grandeurs : la taille de la clé privée d'un individu et la taille de H' sont multipliées par 1,25. Dans le même ordre de grandeur, le coût de la vérification est multiplié par 1,57, rendant la deuxième partie de notre protocoole plus longue de 1/8 de seconde. Enfin, le coût de la signature est lui multiplié par 12,5 : c'est une contrainte acceptable, mais qui limite la portée de ce système aux infrastructures disposant de moyens de calculs importants.

Enfin, il reste un point à améliorer : comme dans tout système basé sur l'identité, la scission de l'autorité en plusieurs entités est fortement appréciée.

5.4 Étude du schéma

5.4.1 Solidité

Il est trivial de constater qu'Alice peut toujours répondre correctement. Dans le cas où $b = 0$ ou 1, elle est capable, de construire les bons hachés. Seul le cas $b = 2$ peut poser des problèmes.

Un n-bit aléatoire peut être considéré comme une suite de n réalisations indépendantes d'un processus de Bernoulli.

Soit $t = y + \omega_i$, pour i fixé, $1 \leq i \leq q$; soient $m = |t|$ et $m' = |t + s|$. En fait, $m' = |t + s| = |t| + |s| - 2T$, où T est le nombre de 1 communs aux deux vecteurs s et t. On a :

$$T = \sum_{s(i)=1} t(i).$$

On a, par conséquent, $|m - m'| = ||t| - |t| - |s| + 2T| = |2T - p|$ car $|s| = p$ par définition. Il faut donc étudier la variable aléatoire $X = |2T - p|$ où T est la somme de p processus de Bernoulli indépendants. En fait, on fait comme si les ω_i étaient indépendants, ce qui n'est pas tout à fait le cas ; on considère cependant qu'ils le sont étant donnés qu'ils sont choisis aléatoirement dans une liste fixée à l'avance. L'espérance ν de la variable aléatoire X est :

$$\nu = \sqrt{2p/\pi}\, 0,798\sqrt{p}.$$

Sa variance étant :

$$V = \sqrt{p - 2p/\pi}\, 0,603\sqrt{p}.$$

On peut prendre e, pour $|m-m'|, |n_j - m_j|$. On utilise alors le théorème central limite pour estimer la probabilité que la moyenne expérimentale μ ne diffère pas trop de la moyenne théorique ν. On obtient :

$$P\{|\mu - \nu| \geq \tau\sqrt{p}\} \leq \frac{2\sigma}{\tau}\sqrt{\frac{2}{\pi pq}}e^{-\tau^2 pq/2\sigma^2} \approx \frac{0,962}{\tau\sqrt{q}}(0,253)^{\tau^2 q}.$$

En prenant $\tau = 0,272$, on obtient que la probabilité d'avoir μ supérieure à $1,07\sqrt{p}$ est au plus $6,96.10^{-7}$ pour $q = 128$. Même lorsque r, le nombre de fois que l'on répète l'algorithme, est 60, soit en moyenne 20 fois le cas $b = 2$, la probabilité de rejeter une personne de bonne foi est inférieure à $1,4.10^{-5}$. Pour éviter ce risque, il suffit, lorsque quelqu'un est rejeté à cette étape, de donner une nouvelle chance au prouveur.

Par conséquent, Alice est sûre d'être identifiée. Encore faut-il que personne ne puisse se faire passer pour elle.

5.4.2 Remarques

On peut faire quelques remarques sur le schéma évoqué précédemment. Tout d'abord, on pourrait penser qu'en abaissant le poids du secret p, la probabilité de réussite du tricheur diminuerait a priori. Seulement, si p est trop faible, il est possible de retrouver sans grandes difficultés le secret à partir de la clé publique (voir Algorithme 2.3.1 page 29).

Le fait que l'on partitionne les n-bits y et $y \oplus s$ pourrait s'avérer dangereux pour des raisons identiques. En effet, les vecteurs formant les partitions de y et de $y \oplus s$ sont de poids assez faible et l'on pourrait retrouver des parties de ces vecteurs grâce à l'algorithme décrit dans le chapitre

2 voir algorithme 2.3.1 page 29, et en déduire des renseignements sur s. En moyenne y et $y \oplus s$ sont de poids voisins de $n/2$, puisqu'ils sont aléatoires. On en déduit, puisque les partitions sont aussi aléatoires, que les vecteurs de $h(y)$ et de $h(y \oplus s)$ sont en moyenne de poids $n/2l$, soit 64 avec les exemples de paramètres déjà cités. On peut supposer que ces vecteurs partitions de taille n sont représentés par n processus de Bernoulli indépendants (un pour chaque bit) avec pour chacun une probabilité $1/2l$ d'être 1 et $1 - 1/2l$ d'être 0. La probabilité d'avoir exactement k uns suit une loi binomiale de paramètres k, n et $1/2l$. Soit S cette probabilité. On peut approximer cette loi binomiale par une loi gaussienne et on obtient que la probabilité d'avoir un vecteur partition de poids inférieur à 30 est négligeable. Or on verra qu'au delà de $p = 30$, il est difficile de trouver des mots de poids p.

5.4.3 Authenticité des paramètres du système

Supposons qu'un attaquant forge sa propre clé maître et ses propres paramètres ID-PKC et réussisse à induire en erreur les utilisateurs en remplaçant le système valide par le sien. Il peut alors déchiffrer tout message chiffré avec ses paramètres. Il peut aussi signer les messages de son choix sous n'importe quelle identité et fournir des clés privées aux utilisateurs. De même dans un système à pki, si un attaquant se substitue à la CA (Certificate Authority), il peut ensuite forger un faux certificat contenant une clé publique dont il possède la clé privée. Il peut ainsi déchiffrer et signer à volonté les messages utilisant ce certificat, par contre il ne peut pas biaiser les clés des utilisateurs qui lui seraient soumises, ces derniers se rendraient compte de la modification de leur clé publique. Ceci est dû au fait que dans un système asymétrique classique l'utilisateur génère lui-même son bi-clé.

Inscription auprès de l'autorité

Dans les deux types de systèmes (PKC et ID-PKC) l'utilisateur doit s'inscrire auprès d'une Registration Autority ou RA que l'on peut considérer respectivement comme une composante du KGC ou de la CA. C'est la RA qui délivre un identifiant à l'utilisateur (respectivement une adresse mél et un certificat) après une enquête plus ou moins poussée pour vérifier son identité. Le point faible de l'ID-PKC réside ici dans le fait que la clé privée issue du KGC doit transiter sur un canal sûr afin de parvenir à l'utilisateur. Shamir [125] suggère de distribuer les clés privées sur des cartes à puce.

Séquestre des clés

Les ID-PKC de par leur design assurent naturellement un séquestre des clés privées, le KGC pouvant à tout moment générer une clé privée à partir d'une identité. Ce séquestre peut être intéressant pour récupérer les clés perdues et pour répondre à des impératifs juridiques (par exemple la capacité d'une DRH à déchiffrer les communications des employés d'une entreprise). Par contre la possibilité pour le KGC de contrefaire une signature remet en cause le principe de non-répudiation (ou imputabilité).

Une solution à ce problème est de distribuer la génération des clés privées auprès de plusieurs KGCs possédant chacun leur propre clé maître. L'utilisateur pourra récupérer sa clé privée à partir d'une combinaison des clés partielles issues des KGCs. Tous les KGCs devront alors être de connivence s'il veulent la reconstituer.

Révocation des clés

Lorsqu'une clé privée est compromise ou qu'un utilisateur en change dans un schéma à pki la révocation d'un certificat est notifiée par une CRL (Certificate revocation list). Dans un schéma ID-PKC la révocation des clés publiques est réalisable très simplement en concaténant l'user-id à une chaîne de caractères d'horodatage (par exemple user-id||2006 ou user-id||10||2006), ce qui offre une forte granularité dans la gestion du temps de vie des clés publiques. Le KGC fournit à chaque échéance une nouvelle clé privée aux utilisateurs qui ne pourront plus utiliser la précédente pour déchiffrer. Il faut bien entendu qu'une politique de sécurité idoine soit dans ce cas parfaitement définie.

Sécurité de la clé maître

Si un attaquant réussit à connaître la clé maître du KGC, il pourra dériver n'importe quelle clé privée à partir d'un user-id et ainsi déchiffrer et signer tout message de son choix, il s'agit du gros point faible de l'ID-PKC. C'est pourquoi il est indispensable dans un tel schéma de s'assurer de la sécurisation de la clé maître. Une bonne solution est de la distribuer entre plusieurs KGCs.

5.4.4 D'un schéma d'identification à un schéma de signature

En employant la méthode décrite par Fiat et Shamir section 1.1.7 page 8, on peut transformer le schéma précédent en schéma de signature. Pour cela il suffit de faire en sorte qu'il n'y ait plus d'interaction entre Alice et Bob : Alice créera elle-même les n-bits z et les valeurs de b, sans toutefois en avoir le contrôle. Pour cela, elle utilisera une fonction de hachage, c'est-à-dire une fonction qui, à partir du message M à signer et des aléas, y générés par Alice donne un résultat distinct pour chaque M (si tel n'était pas le cas, Alice pourrait trouver deux messages M et M' ayant même résultat par la fonction de hachage, elle pourrait alors envoyer M à Bob et prétendre ultérieurement lui avoir transmis M', ce que justement l'on veut éviter).

On obtient l'algorithme suivant :

1. Alice choisit aléatoirement r, par exemple 64, n-bits y_j.
2. Elle calcule alors la valeur d'une fonction de hachage $f(M, \sigma(y_1), \ldots, \sigma(y_r)$ où M est le message à signer, pour obtenir r n-bits aléatoires z_j.
3. Pour chaque j, elle calcule :
$$n_{ij} = |y_j + z_{ij}|$$
$$m_{ij} = |y_j + s + z_{ij}|$$
pour $1 \leq i \leq q, 1 \leq j \leq r$; avec $z_{ij} = z_j + \omega_{(ai+c) \bmod q}$ où a et c sont deux entiers aléatoires avec a premier avec q.
À l'aide de la fonction de hachage appliquée aux permutés des (n_{ij}) et des (m_{ij}), Alice fabrique r valeurs b_j, telles que pour tout $j, 0 \leq b_j \leq 2$.
Il ne lui restera qu'à envoyer à Bob pour chaque $j, \sigma(y_j), \sigma(y_j + s), \sigma((n_{ij})), \sigma((m_{ij}))$ qui lui permettront entre autre de reconstituer les vecteurs (z_j) ainsi que les (b_j), Bob connaissant la fonction de hachage, et, si $b = 0, h(y_j)$ et $h((n_{ij}))$, si $b = 1, \sigma(y_j + s)$ et $\sigma((m_{ij}))$, si $b = 2, \sigma((n_{ij}))$ et $\sigma((n_{ij}))$.
4. Pour chaque j, Bob fera les mêmes vérifications que lors des étapes 2 et 4 de l'algorithme d'authentification.

5.4.5 Construction générique

Pour construire un schéma de signature basé sur l'identité il existe une construction *standard* due à Bellare, Namprempre et Neven [12].

Leur contruction nécessite deux schémas de signature, pour une comparaison nous utilisons :

1. mCFS - mCFS
2. Stern - Stern

Ce tableau donne les résultats (pour un schéma de signature) :

	mCFS-Stern	(1)	(2)
taille	1,5 Mo	$\approx$ 1 Mo	48 Ko
coût	1.4 $\times 10^9$ op.	2.2 $\times 10^{10}$ op.	2.2 $\times 10^7$ op.

Valeurs pratiques

La grosse différence comparée à une utilisation classique du schéma de CFS est que le code utilisé dans le schéma de Stern est très long : 2^{16} au lieu 2^9 pour le schéma de Stern classique, ce qui accroît grandement le coût de communication.

Dans la table suivante, nous résumons pour des paramètres de $m = 16$, $t = 9$, les paramètres généraux pour les schémas d'authentification et signature décrit précédemment.

clé publique	clé privée	taille de la matrice
tm	tm	$2^m tm$
144	144	1 Mo

coût de la communication	génération de la clé
$\approx 2^m \times \#tours$	
500 Ko (58 tours)	1 s

Paramètres pour le schéma d'authentification : $m = 16, t = 9$

clé publique	clé privée	taille de la matrice
tm	tm	$2^m tm$
144	144	1 Mo

taille de la signature	génération de la clé
$\approx 2^m \times 150$	
1.5 Mo	1 s

Paramètres pour le schéma de signature : $m = 16, t = 9$

5.5 Preuve de sécurité du schéma

Théorème 5.5.1

Le schéma d'authentification basé sur l'identité présenté dans la section 5.3 page 93 est sûr au sens imp-pa *si les problèmes BD et GD sont difficiles à résoudre.*

Preuve La preuve est obtenue en adaptant les preuves de Dallot [41] et Stern [129] à notre schéma. La suite de cette section détaille cette preuve. Nous structurons la preuve en suivant la suite de jeux **Jeu 0, Jeu 1, Jeu 2, Jeu 3, Jeu 4, Jeu 5**. Le **Jeu 0** étant le jeu d'attaque originel, c'est-à-dire le jeu standard imp-pa. □

Les jeux successifs sont obtenus par de petites modifications des jeux précédents, de telle manière que les différences des avantages des adversaires dans des jeux consécutifs est facilement quantifiable. Pour calculer cette différence, nous aurons besoin du lemme suivant :

Lemme 5.5.1

Soit X_i,X_{i+1}, B des événements définis avec une certaine distribution de probabilités et supposons que :

$$X_i \wedge \neg B \Leftrightarrow X_{i+1} \wedge \neg B.$$

Alors :

$$|\operatorname{Prob} X_i - \operatorname{Prob} X_{i+1}| \leq \operatorname{Prob} B.$$

Soit q_h, q_E, q_C le nombre de requêtes qu'effectue un adversaire $\mathcal{A}$ à des fonctions de hachage, des clés de l'utilisateurs et l'utilisation d'oracles.

Nous voulons montrer qu'il existe des adversaires $\mathcal{C}, \mathcal{D}$ qui sont capables de casser les problèmes BD et GD respectivement. Pour donner la valeur du haché, de la clé de l'utilisateur et de l'oracle, trois listes $h^{\text{list}}, UK^{\text{list}}$ et Λ sont mises à jour. S'il n'y a aucune valeur associée à une entrée dans une liste, nous notons sa sortie $\perp$.

- La liste h^{list} est constituée des couples de la forme (x, s) indexés par (id, i), où i est un indice dans $\{1, \ldots, 2^{n-k}\}$, id est une identité, et $H's^T = x = h(id, i)$ si $x \neq \perp \neq s$.
- La liste UK^{list}, est constituée d'entrées de la forme $(id, sk[id])$.
- La liste Λ contient les indices $\Lambda(m)$ associés au message m, pour lequel le simulateur est capable de produire une signature sur $h(m, \Lambda(m))$.

Jeu 0. Il s'agit du jeu standard imp-pa. Les clés maîtres publiques et secrètes sont obtenues en utilisant l'algorithme $\mathsf{Gen}_{\mathsf{mCFS}}(1^\kappa)$.

En particulier, la clé publique maîtresse est constituée de $H' = SHP$ et d'une fonction de hachage $h : \{0,1\}^* \to \mathbb{F}_2^{n-k}$. La clé secrète maître est constituée de (S, H, P), où $H \xleftarrow{R} \mathsf{Goppa}(n, k)$, S est une matrice non singulière $(n-k) \times (n-k)$ et P est une matrice de permutation $n \times n$. Par conséquent :

$$\operatorname{Prob} X_0 = \mathsf{Adv}^{\mathsf{imp-pa}}_{\mathcal{IBI},\mathcal{A}}.$$

Jeu 1.(Simulation des requêtes de hachage et de clé de l'utilisateur) Nous changeons la façon dont le hachage et l'extraction de clé d'utilisateur sont donnés. Pour les requêtes de hachage de la forme (id, i), il y a deux situations, dépendant de $i = \Lambda(id)$. Si cela est le cas, un syndrome décodable $x = H's^T$ est donné en sortie, et le mot de code correspondant s est stocké, c'est-à-dire que h^{list} est mis à jour avec (x, s) à la place de l'entrée indexée par (id, i). Si $i \neq \Lambda(id)$ les requêtes de hachage

sont simulées en prenant un élément aléatoire dans $\mathbb{F}_2^{n-k}$, et alors ces requêtes sont distribuées comme avec un oracle aléatoire. La Figure 5.4 donne les détails de ce jeu.

Input : Une paire (id, i)
Output : Un syndrome s

$(s, x) \leftarrow h^{\text{list}}(id, i)$;
if $i \neq \Lambda(id)$ **then**
  if $s = \perp$ **then**
    $x \xleftarrow{R} \mathbb{F}_2^{n-k}$;
    $h^{\text{list}}(id, i) \leftarrow (x, \perp)$;
  end
  return $h(id, i) = x$;
else
  if $x = \perp$ **then**
    $s \xleftarrow{R} \{w \in \mathbb{F}_2^n | \mathsf{wt}(w) \leq t\}$;
    $x \leftarrow H' s^T$;
    $h^{\text{list}}(id, i) \leftarrow (x, s)$;
  end
  return $h(id, i) = x$;
end

FIGURE 5.4 – Simulation des requêtes de hachage

D'un autre côté, les requêtes de clés d'utilisateur pour id sont données en choisissant l'indice spécial $\Lambda(id)$ aléatoirement, en faisant appel à l'oracle de hachage avec $(id, \Lambda(id))$ et en donnant en sortie (s, i) comme clé secrète de l'utilisateur associée. Les détails sont données dans la figure 5.5.

À la fin de la simulation, l'oracle aléatoire h a sorti $q_h + q_E + 1$ syndromes. Certains d'entre eux sont produits avec l'indice spécial $i = \Lambda(id)$; ces syndromes ne sont pas distribués de manière uniformément aléatoire dans $\mathbb{F}_2^{n-k}$, au lieu de cela ils ont été modifiés pour permettre de répondre aux requêtes de clé secrète de l'utilisateur. Ce peut alors être le cas avec un adversaire $\mathcal{A}$ faisant une requête à h sur une paire (id, j) telle que j peut être remplacé par $\Lambda(id)$. Ceci causerait une incohérence, puisqu'alors la sortie $h(id, j)$ serait un syndrome aléatoire, au lieu d'un syndrome décodable.

Celle-ci se produit avec une probabilité d'au plus $\frac{q_E}{2^{n-K}}$ (les indices $\Lambda(id)$ sont définies seulement en répondant à la requête d'extraction de clé). Par conséquent,

$$|\Pr[X_0] - \Pr[X_1]| \leq \frac{q_E}{2^{n-k}}$$

Input : Une identité id
Output : Une clé secrète d'utilisateur (s, id)
if $\Lambda(id) = \perp$ **then**
  $\Lambda(id) \xleftarrow{R} \{1, \ldots, 2^{n-k}\}$;
end
$(x, s) \leftarrow h(id, \Lambda(id))$;
$i \leftarrow \Lambda(id)$;
$\Lambda(id) \leftarrow \perp$;
return $sk[id] = (s, i)$;

FIGURE 5.5 – Simulation des requêtes de clé d'utilisateur

Jeu 2.(Changement de l'algorithme de génération de la clé maître) L'algorithme de génération de clés est changé de telle manière que $H \leftarrow \mathsf{Binary}(n, k)$. Alors,

$$|\Pr[X_2] - \Pr[X_1]| \leq \mathsf{Adv}^{\mathsf{GD}}_{\mathcal{D}}(n)$$

où $\mathcal{D}$ est un algorithme qui simule l'environnement du Jeu 2 pour $\mathcal{A}$ si $H \leftarrow \mathsf{Goppa}(n, k)$ et produit en sortie $d = 1$ si $\mathcal{A}$ usurpe avec succès l'identité $id^\star$, et $d = 0$ sinon ; et $\mathcal{D}$ simule l'environnement du Jeu 3 pour $\mathcal{A}$ si $H \leftarrow \mathsf{Binary}(n, k)$ et produit en sortie $d = 1$ si $\mathcal{A}$ usurpe avec succès l'identité $id^\star$, et $d = 0$ sinon. Il est facile de voir que :

$$\Pr[H \xleftarrow{R} \mathsf{Goppa}(n, k) : \mathcal{D}(H) = 1] = \Pr[X_2].$$

et

$$\Pr[H \xleftarrow{R} \mathsf{Binary}(n, k) : \mathcal{D}(H) = 1] = \Pr[X_1].$$

Jeu 3.(Deviner l'identité) Un indice aléatoire $j^+ \xleftarrow{R} \{1, \ldots, q_h + q_E + q_C\}$ est choisi. La j^+-ème requête de hachage (id^+, i^+) est fixée à $U(x^+)^T$, où $x^+ \xleftarrow{R} \mathbb{F}_2^{n-k}$, c'est-à-dire $h(id^+, i^+) = U(x^+)^T$. L'espace des probabilités n'est pas modifié puisque $x^+ \xleftarrow{R} \mathbb{F}_2^{n-k}$ et U non singulière et par conséquent $\Pr[X_3] = \Pr[X_2]$.

Jeu 4.(Abandonner le jeu) Soit $(id^\star, i^\star)$ l'identité et l'indice que $\mathcal{A}$ désire usurper. Si $id^\star \neq id^+$ ou $i^\star \neq i^+$ alors le challenger abandonne le jeu. Puisque le Jeu 4 est réalisé en conditionnant le Jeu 3 à un événement indépendant de probabilité $\frac{1}{q_H + q_E + 1}$ on obtient :

$$\Pr[X_4] = \frac{\Pr[X_3]}{q_H + q_E + 1}$$

Jeu 5. (Répondre à des requêtes concernant l'identité $id^\star$) Nous devons répondre aux requêtes concernant $id^\star$ sans connaître le mot de code $s^\star$ correspondant à $h(id^\star, i^\star) = x^\star$, c'est-à-dire $s^\star$ tel que $x^\star = H' s^+$ et $x^\star = U(x^+)^T$. Nous pouvons répondre à ces requêtes en temps polynomial en utilisant l'algorithme 3 de [130]. L'idée sous-jacente est que l'algorithme utilise une simulation à zéro [52].

Au début de chaque itération du protocole d'identification de base, l'algorithme choisit au hasard une des trois stratégies de tricherie, où chaque stratégie permet d'interagir avec succès avec un vérificateur tricheur $\overline{\mathsf{CV}}$ avec une probabilité $2/3$. Dans le cas où l'algorithme ne peut interagir

avec succès avec $\overline{\mathsf{CV}}$, il rétablit l'adversaire $\mathcal{A}$ pour le cycle actuel (voir [130] pour plus de détails). Dans l'ensemble, l'espace des probabilités n'est pas modifié, puis on obtient $\Pr[X_5] = \Pr[X_4]$.

Le théorème 1 de [130] implique qu'un adversaire $\mathcal{A}$ usurpant l'identité d'un utilisateur $id^\star$ après k tours du protocole et avec un avantage de $(2/3)^k + \epsilon_1$ pour un $\epsilon_1 > 0$ non négligeable, peut être converti en un algorithme PPT calculant $s^\star$ tel que $H'(s^\star)^T = x^\star$ avec une probabilité de $\epsilon_1^3/10$. Un calcul simple montre que $(s^+)^T = P(s^\star)^T$ est une solution au problème BD avec en entrée $H \xleftarrow{R} \mathsf{Binary}(n,k)$ et $x^+ \xleftarrow{R} F_2^{n-k}$. Soit $\mathcal{C}$ un algorithme qui simule le Jeu 5 pour un adversaire $\mathcal{A}$ en utilisant l'entrée du problème BD. Alors,

$$\mathsf{Adv}_{\mathcal{C}}^{\mathsf{BD}} \geq \frac{\left(\Pr[X_5] - (2/3)^k\right)^3}{10}$$

En collectant toutes les probabilités :

$$\begin{aligned}
(2/3)^k + \epsilon &\leq \mathsf{Adv}_{I\mathcal{BI},\mathcal{A}}^{\mathsf{imp-pa}} \\
&\leq \frac{q_E}{2^{n-k}} + \mathsf{Adv}_{\mathcal{D}}^{\mathsf{GD}}(n) + \Pr[X_5](q_h + q_E + 1) \\
&\leq \frac{q_E}{2^{n-k}} + \mathsf{Adv}_{\mathcal{D}}^{\mathsf{GD}}(n) + \left((\mathsf{Adv}_{\mathcal{C}}^{\mathsf{BD}})^{1/3} + (2/3)^k\right) 10^{1/3}(q_h + q_E + 1)
\end{aligned}$$

et alors

$$\epsilon \leq \frac{q_E}{2^{n-k}} + \mathsf{Adv}_{\mathcal{D}}^{\mathsf{GD}}(n) + \left((\mathsf{Adv}_{\mathcal{C}}^{\mathsf{BD}})^{\frac{1}{3}} + (1 - \frac{1}{\sqrt[3]{10}})(\frac{2}{3})^k\right) 10^{\frac{1}{3}}(q_h + q_E + 1)$$

Cette dernière équation peut se lire comme suit : un succès d'un adversaire usurpant l'identité d'adversaire avec un avantage de $(\frac{2}{3})^k + \epsilon$ implique un adversaire réussissant à résoudre un de deux problèmes BD et GD.

5.6 Conclusion

Dans ce papier, nous avons présenté un schéma d'authentification et un schéma de signature basé sur les codes correcteurs d'erreurs. Ce schéma est le premier schéma basé sur l'identité non fondé sur la théorie des nombres. Le schéma combine deux schémas bien connus et hérite des mauvaises propriétés de ces systèmes : la taille des données publiques est grande, le coût de communication pour le schéma d'authentification est élevé et la taille de la signature est également très grande. Mais malgré ces faiblesses notre système présente la première alternative (non générique) à la théorie des nombres pour la cryptographie basée sur l'identité et permet d'ouvrir de nouveaux domaines de recherche.

Chapitre 6

Schéma de signature de cercle à seuil

Ce chapitre a donné lieu à l'article suivant [2] :

A new efficient threshold ring signature scheme based on coding theory
avec Carlos Aguilar Melchor et Philippe Gaborit
PQ CRYPTO 2008, LNCS, à paraître.

La notion de signature de cercle a été introduite par Rivest, Shamir et Tauman en 2001. Bresson, Stern et Szydlo ont étendu cette notion au concept de signature de cercle à seuil (t-out-of-N threshold ring signature) en 2002. La taille de la signature pour le meilleur schéma de signature de cercle à seuil anonyme connu est exponentiel en t et linéaire en N. En 1993, Stern a proposé un schéma de signature et d'identification basé sur le problème du décodage par syndrome ([129] et chapitre 2 page 22 de cette thèse) Nous présentons dans ce chapitre, une *généralisation* du schéma de Stern dans le cas d'un schéma de signature de cercle à seuil. La taille de notre signature est en $\mathcal{O}(N)$ et ne dépend pas de t. Notre protocole est anonyme, sûr dans le modèle de l'oracle aléatoire, possède une très petite taille de clé publique et a une complexité en $\mathcal{O}(N)$ ce qui permet de traiter des t de n'importe quelle taille. Ce protocole est le premier protocole de signature de cercle basé sur les codes correcteurs d'erreurs effectif ainsi que le premier protocole de signature de cercle à seuil basé sur les codes correcteurs d'erreurs. De plus, il a une meilleure complexité que les schémas basés sur la théorie des nombres car sa complexité est en $\mathcal{O}(Nt)$.

Cette notion, bien que peu étudiée jusqu'à présent dans la littérature, est porteuse d'applications réelles et pratiques. Dans ce chapitre, nous présentons un schéma de signature de cercle (pouvant être utilisé à seuil) basé sur le schéma de Stern [130]. Nous montrons comment généraliser la construction de Stern dans un contexte de signature de cercle.

6.1 Introduction

La notion de *signature de cercle* a été introduite par Rivest, Shamir et Tauman en 2001 [117] (notée RST dans la suite). Les signatures de cercle sont reliées mais pas comparables à la notion de signatures de groupe [37]. D'un côté, les signatures de groupe ont la propriété particulière que l'anonymat d'un signataire peut être révoqué (ce qui veut dire que le signataire peut être tracé) par un manager du groupe qui est désigné, d'un autre côté, les signatures de cercle autorisent une plus grande flexibilité et il n'y a pas de manager de groupe centralisé ni de coordination entre les différents utilisateurs (en fait, les utilisateurs peuvent ne pas savoir qui sont les autres signataires

potentiels au moment de la création de leur clé publique). La motivation originelle est d'autoriser le partage du secret de manière anonyme. Par exemple, un officiel haut placé d'un gouvernement peut signer des informations au nom du cercle de tous les autres officiels haut placés, l'information peut être vérifiée comme venant de quelqu'un de réputé sans dévoiler le véritable signataire.

Bresson et al. [20] étendent la notion de signature de cercle en un schéma de *signature de cercle à seuil* (*threshold ring signature* scheme) en utilisant le concept de partitionnement et de fonctions de combinaisons. Supposons que t utilisateurs veulent partager des informations secrètes, de telle manière que n'importe quel vérifieur sera convaincu que t utilisateurs *parmi le groupe sélectionné* ont certifié la validité de l'information. Construire t signatures de cercle ne prouve clairement pas que le message ait été signé par différents signataires. Un schéma de *signature de cercle à seuil* prouve effectivement qu'un minimum d'utilisateurs d'un certain groupe ont collaboré pour produire la signature, en cachant les membres précis formant ce sous-groupe (par exemple le cercle des clés publiques de tous les membres du cabinet du Président).

Contribution Dans cette section, nous présentons une *généralisation* du schéma d'authentification et de signature de Stern [129] dans le cadre de schémas de signature de cercle et de signature de cercle à seuil. Les performances de notre schéma ne dépendent pas du nombre t de signataires dans le cercle, la complexité totale et la longueur des signatures dépendent seulement linéairement en le nombre maximum de signataires N. Notre protocole garantit également l'anonymat. Au delà de ces considérations et de son efficacité, notre protocole est aussi le premier protocole non générique de signature de cercle à seuil basé sur la théorie des codes correcteurs et constitue une alternative intéressante aux protocoles basés sur la théorie des nombres. De plus, notre protocole possède la plus petite taille de clé publique, de signature et de complexité pour un schéma de signature de cercle à seuil.

Organisation du chapitre Le reste de ce chapitre s'organise de la manière suivante. Dans la section 6.2, nous donnons un état de l'art des signatures de cercle et des signatures de cercle à seuil. Dans la section 6.3, nous présentons notre nouvelle généralisation du schéma de Stern dans le cas des signatures de cercle à seuil. Dans la section 6.4, nous étudions la sécurité du schéma que nous proposons. Dans la section 6.5 nous considérons une variation du protocole en utilisant des matrices doublement circulantes. Dans la section 6.6 nous discutons du coût et de la longueur de la signature. Finalement, nous concluons dans la section 6.7.

6.2 Survol des signatures de cercle

6.2.1 Signature de cercle

Cette section reprend en partie le chapitre 7 de la thèse d'Emmanuel Bresson [19].

Dans un schéma de signature de groupe, l'anonymat peut être levé en cas de conflit. Ceci offre donc aux membres non-signataires une certaine sécurité contre les abus. En fait, les signatures de groupe constituent un outil approprié dans le cas où les membres sont d'accord pour coopérer (i.e., pour s'enregistrer, être inscrits dans la liste, etc.). Le concept de ring signature (littéralement signature de cercle, certain préfère signature d'anneau) est quelque peu différent. Il n'a été formalisé que fin 2001 par Rivest, Shamir et Tauman [117]. Il est particulièrement bien adapté aux situations

où les membres du cercle ne veulent pas coopérer, puisque ce genre de schéma ne requiert ni procédure d'initialisation, ni enregistrement, ni manager du groupe. Une signature de cercle spécifie un ensemble de signataires potentiels et produit une preuve tangible démontrant que l'auteur de la signature appartient à cet ensemble. Le schéma est anonyme, c'est-à-dire, que le vérifieur n'apprend rien sur l'identité du signataire en effectuant sa vérification. Mais cette fois, il n'y a pas de juge qui puisse lever cet anonymat.

En suivant le formalisme des signatures de cercle proposées dans [117], nous décrivons dans cette section les définitions de base et les propriétés éligibles aux schémas de signature de cercle. Supposons que chaque utilisateur ait reçu (à l'aide d'une PKI ou d'un certificat) une clé publique p_{k_i}, pour laquelle correspond une clé secrète notée s_{k_i}. Un schéma de signature de cercle régulier consiste en le triplet suivant : (Key-Gen, Sign et Verify) :

- **Key-Gen** est un algorithme polynomial qui prend en entrée un paramètre de sécurité et retourne les paramètres privés et publics.
- **Sign** est un algorithme polynomial probabiliste qui prend en entrée les paramètres du système, un paramètre privé, une liste de clés publiques $p_{k_1}, \ldots, p_{k_N}$ du cercle et un message M. La sortie de cet algorithme est une signature de cercle σ pour le message M.
- **Verify** est un algorithme polynomial probabiliste qui prend en entrée un message M, une signature de cercle σ, et les clés publiques de tous les membres du cercle correspondant, et retourne *Vrai* si la signature de cercle est valide et *Faux* dans le cas contraire.

La plupart des schémas de signature de cercle ont une taille de signature linéaire en N, la taille du cercle. Beaucoup de schémas ont été proposés, on pourra citer le travail de Bendery, Katzyz et Morselli [13] dans lequel ils présentent trois schémas de signature de cercle qui sont prouvés sûrs dans le modèle standard. Récemment, Shacham et Waters [124] ont proposé une signature de cercle où pour N membres la signature consiste en $2N + 2$ éléments du groupe et nécessite $2N + 3$ couplages à vérifier.

Une avancée importante en ce qui concerne la taille des signatures de cercle a été obtenue dans [42], les auteurs proposent le premier (et unique pour l'instant) schéma à taille constante basé sur les fonctions d'accumulations et le schéma d'identification zero-knowledge de Fiat-Shamir. La signature dérivée du schéma de Fiat-Shamir a une taille d'au moins 160 kbits. Une autre construction proposée par Chandran, Groth et Sahai ([34]) a une taille en $\mathcal{O}(\sqrt{N})$.

Récemment dans [140], Zheng, Li et Chen présentent un schéma de signature basé sur les codes correcteurs d'erreurs pour lesquels la longueur de la signature est de $144 + 126N$ bits, mais ce schéma est basé sur la signature de Courtois, Finiasz et Sendrier (voir [39] et section 2.2.1 page 21 de cette thèse) qui reste très lent comparé aux autres schémas. Une généralisation des signatures de cercle en *mesh signature* a été proposée par Boyen dans [18].

6.2.2 Signature de groupe *vs* signature de cercle

En examinant plus précisément les scénarios, on peut mettre en lumière trois différences entre ces schémas et les signatures de groupe classiques.

Tout d'abord, il n'y a plus de groupe prédéfini, de clé publique de groupe ou de liste de membres. Aucune autorité (juge ou manager) n'est nécessaire, et les procédures de génération de clés, d'enregistrement, ou d'ouverture sont absentes. Le seul pré-requis est l'existence d'une infrastructure à clé publique attribuant à chaque joueur (signataire potentiel) une paire clé privée/clé publique. Tout joueur a alors la possibilité de spécifier un ensemble de joueurs auquel il appartient, et de produire une signature prouvant que l'auteur (i.e. lui-même), appartient à ce cercle. Aucune

action concertative n'est nécessaire, en particulier, les joueurs non-signataires ne sont pas forcément au courant de leur implication dans cette signature. Ceci n'est évidemment pas réalisable dans une signature de groupe classique, où les membres, par définition, sont des joueurs enregistrés d'un groupe publiquement connu (et, donc, devant avoir coopérés à un moment ou à un autre).

Deuxièmement, l'absence d'autorité dans une signature de cercle permet d'obtenir un anonymat inconditionnel ; autrement dit, même un juge avec une puissance de calcul infinie ne peut déterminer l'identité de l'auteur. Ceci, à nouveau, n'est pas possible avec une signature de groupe, pour laquelle doit exister une trappe, destinée au juge : de façon inhérente au concept, une signature de groupe ne peut assurer qu'un anonymat calculatoire. Cette différence peut paraître mineure, mais prend cependant tout son sens lorsque l'on veut protéger des données extrêmement sensibles ; par exemple, même si RSA est cassé, l'anonymat demeure.

Troisièmement, on notera que les signatures de cercle peuvent être rendues extrêmement efficaces en termes de coût calculatoire. Alors que les schémas de signature de groupe utilisent intensément des calculs asymétriques coûteux, tels que des preuves sans information de connaissance ou d'appartenance à un langage, le schéma proposé dans [117] est très rapide ; il ne nécessite qu'une exponentiation modulaire pour signer, plus un nombre linéaire de multiplications modulaires pour la signature et la vérification. Une telle efficacité mérite d'être soulignée. La contrepartie est que la taille de la signature croît linéairement avec la taille du cercle spécifié, mais cet inconvénient est indissociable du concept : le cercle n'étant pas fixé mais choisi au vol, sa composition doit être indiquée dans la signature elle-même.

6.2.3 Signature de cercle à seuil

Dans [20], Bresson, Stern et Szydlo introduisent la notion de signature de cercle à seuil. Nous décrivons dans cette section les définitions de base et les propriétés des schémas de signature de cercle à seuil.

Supposons que chaque utilisateur ait créé ou reçu une clé secrète s_{k_i} et qu'une clé publique correspondante p_{k_i} est publiée pour tout le monde.

Soit $A_1, \ldots, A_N$, N signataires potentiels du cercle avec leurs $p_{k_1}, \ldots, p_{k_N}$ clés publiques. Alors t parmi les N membres forment le groupe des signataires, l'un d'eux, L, est le leader du t-sous-groupe.

L'algorithme se déroule de la manière suivante :

- Initialisation (Set-up) : initialise l'état du système, prend en entrée un paramètre de sécurité 1^l, crée une base de données publiques $p_{k_1}, \ldots, p_{k_N}$, choisit un leader L du groupe et génère les paramètres du système ;
- Création de la clé publique du groupe (Make-GPK) : algorithme de construction de la clé publique du groupe (GPK) ;
- Engagement-Challenge-Réponse (Commitment-Challenge-Response) : un moyen permettant de cacher temporairement une suite de bits qui ne peuvent être changés ;
- Vérification : prend en entrée les réponses aux challenges et vérifie que le calcul ait été fait honnêtement, et retourne un booléen.

Dans [20], la taille de la signature grandit avec le nombre d'utilisateurs N et le nombre de signataires t. De manière plus précise, le coût de la t-out-of-N signature est de $2^{\mathcal{O}(t)}\lceil \log_2 N \rceil \times (tl + Nl)$ opérations binaires avec l le paramètre de sécurité. En 2003, Liu et al. [76] ont proposé un autre schéma de signature de cercle basée sur le schéma de partage de secret de Shamir. Leur schéma est séparable, avec une longueur de signature linéaire en N mais avec une complexité

de $\mathcal{O}(N^2)$ pour $t \approx \frac{N}{2}$ (ce qui est le coût du partage du secret). Les *mesh signatures* de [18] peuvent aussi être utilisées dans ce cas : la longueur de la signature est aussi linéaire en N mais la vérification coûte Nt pairings bilinéaires.

La même année, Wong et al. [139] proposent une autre signature de cercle à seuil en utilisant la méthode de construction *tandem.* Ils proposent une construction basée sur des codes de Reed-Solomon (section 1.2.1) d'un schéma de signature de cercle (que l'on peut noter 1-out-N par analogie aux schémas de signature de cercle à seuil). Le schéma nécessite $[\binom{N}{t} - 1]t$ permutations à sens unique et t inversions de ces permutations. La complexité est proportionnelle à $t\binom{N}{t}$.

Une variation pour des schémas de signature de cercle a été introduite dans [132], où les auteurs introduisent la notion de *signature de cercle liée (linkable ring signature)* pour laquelle un signataire peut signer une seule fois de manière anonyme, si bien qu'un vérifieur pourra se rendre compte d'une seconde signature du même signataire. Bien que cette propriété peut avoir des applications intéressantes (par exemple pour le vote électronique) elle ne garantit pas un anonymat total (dans le sens qu'on ne peut pas répéter la signature). Plus tard, leur schéma a été étendu en un schéma de signature de cercle à seuil avec une complexité en $\mathcal{O}(N)$, mais une fois encore, on obtient seulement une signature de cercle liée ce qui ne correspond pas au but originel de [117] et [20], c'est-à-dire un schéma complètement anonyme.

6.3 Notre schéma de signature de cercle à seuil

Dans cette section, nous décrivons un schéma effectif d'identification de cercle à seuil basé sur la théorie des codes correcteurs d'erreurs. Ce schéma est une *généralisation* du schéma de Stern (section 2.2.2). De plus, en appliquant l'heuristique de Fiat-Shamir [46] et section 1.1.7 page 8 à notre schéma d'identification, nous obtenons de manière immédiate un schéma de signature de cercle à seuil (t-out-of-N) dont la taille est en $\mathcal{O}(N)$.

Introduisons tout d'abord quelques notations :

6.3.1 Notation de la permutation

Nous introduisons tout d'abord deux notions de *permutation de blocs* que nous utiliserons dans notre protocole. Considérons n et N deux entiers.

Définition 6.3.1

Une permutation de n-blocs de taille constante Σ sur N blocs est une permutation par blocs qui permute N blocs de taille n bloc par bloc. Chaque bloc peut être traité comme une position unique pour les permutations usuelles.

Un type plus général de permutation est la permutation de n-blocs Σ sur N blocs.

Définition 6.3.2

Une permutation de n-blocs Σ sur N blocs est une permutation qui vérifie que la permutation d'un bloc de longueur n parmi N blocs est exactement incluse dans un bloc de longueur n.

Une permutation de n-blocs de taille constante est une permutation de n-blocs particulière dans laquelle les blocs sont permutés tels quels.

Par exemple la permutation $(6,5,4,3,2,1)$ est une permutation de 2-blocs sur 3 blocs et la permutation $(3,4,5,6,1,2)$ est une permutation de 2-blocs de taille constante sur 3 blocs telle que l'ordre dans chaque bloc ($(1,2)$, $(3,4)$ et $(5,6)$) est préservé après permutation des blocs.

La notion de produit de permutation est alors la suivante. Posons σ, une famille de N permutations $(\sigma_1,\ldots,\sigma_N)$ de $\{1,\ldots,n\}$ sur n positions et Σ une permutation constante de n-blocs sur N blocs définie sur $\{1,\ldots,N\}$. Considérons un vecteur v de taille nN de la forme :

$$v = (v_1, v_2, \ldots, v_n, v_{n+1}, \ldots, v_{n+n}, v_{2n+1}, \ldots, v_{nN}),$$

notons V_1 les n premières coordonnées de v et V_2 les n coordonnées suivantes et ainsi de suite, pour obtenir : $v = (V_1, V_2, ..., V_N)$. Nous pouvons définir une permutation de n-blocs sur N blocs, $\Pi = \Sigma \circ \sigma$ telle que :

$$\Pi(v) = \Sigma \circ \sigma(v) = (\sigma_1(V_{\Sigma(1)}), \ldots, \sigma_N(V_{\Sigma(N)})) = \Sigma(\sigma_1(V_1), \cdots, \sigma_N(V_N)).$$

6.3.2 Aperçu

Considérons un cercle de N membres $(P_1, \cdots, P_N)$ et parmi eux t utilisateurs qui veulent prouver qu'ils ont coopéré pour produire une signature de cercle. Chaque utilisateur P_i calcule une matrice publique H_i de $(n-k) \times n$ bits. Une clé publique des utilisateurs consiste en les clés publiques des utilisateurs H_i et en un entier ω (commun à toutes les clés publiques). La clé secrète associée est un mot s_i de poids ω du code $\mathcal{C}_i$ associé au dual de H_i.

L'idée générale de notre protocole est que chacun des t signataires calcule une instance du schéma de Stern (voir section 3.2 page 37) en utilisant la matrice H_i et un syndrome nul comme paramètre (comme dans la variation proposée dans [49]). Les résultats sont collectés par un leader L parmi l'ensemble des signataires dans le but de former, en plus de la simulation des $N-t$ non signataires, un nouveau protocole interactif de Stern avec le vérifieur V. La clé publique maître H est créée en faisant la somme directe des matrices publiques des membres du cercle. Le prouveur P formé par l'ensemble de t signataires parmi N (voir figure 6.1 page 111), prouve (par une version légèrement modifiée du schéma de Stern, en ajoutant une condition sur la forme de la permutation) au vérifieur V qu'il connaît un mot de code s de poids $t\omega$ avec une structure particulière :

- s a un syndrome nul pour H
- une forme spéciale sur ces N blocs de taille n : chaque bloc de longueur n est de poids 0 ou ω.

En fait, ce type particulier de mot peut être seulement obtenu par la coopération entre t membres du cercle. La complexité est le coût de N fois le coût d'un schéma de Stern classique pour un seul prouveur et cela *pour toute valeur de* t.

Au delà de la combinaison de deux protocoles de Stern (un fait individuellement par chaque signataire P_i avec le leader, et un légèrement modifié fait entre le leader et le vérifieur), notre schéma repose sur les trois idées suivantes :

1. La clé publique maîtresse H est obtenue comme concaténation de toutes les matrices publiques H_i de chacun des N utilisateurs.
2. L'indistinguabilité parmi les membres du cercle, dans un premier temps, est obtenue en utilisant un syndrome commun pour tous les membres du cercle (un syndrome nul) et dans un deuxième temps en prenant des clés secrètes s_i avec le même poids ω qui est une valeur publique associée aux matrices publiques H_i.

3. Une contrainte sur le type de permutation utilisée est ajoutée au schéma de Stern classique. Au lieu d'utiliser une permutation de taille Nn on utilise une permutation de n-blocs sur N blocs (voir Définition 6.3.1 page 109), qui garantit que le prouveur connaît un mot avec une structure particulière, qui peut être uniquement obtenue grâce à l'interaction de t signataires.

6.3.3 Initialisation

L'algorithme d'initialisation est exécuté pour obtenir les valeurs des paramètres l, n, k, t, w. L'entier l est le paramètre de sécurité, n et $n-k$ sont les paramètres de la matrice, ω désigne le poids de la clé secrète s_i et enfin t représente le nombre de signataires.

Cet algorithme crée aussi une base de données publiques $p_{k_1}, \ldots, p_{k_N}$, (ici les matrices H_i). On peut remarquer que les paramètres n, k et ω sont fixés une fois pour toute, et que chaque nouvel utilisateur connaissant ces paramètres publics peut rejoindre le cercle. Le paramètre t doit seulement être précisé au début du protocole.

Les matrices H_i sont construites de la manière suivante :
- choisir un vecteur aléatoire s_i de poids ω,
- générer $k-1$ vecteurs aléatoires,
- considérer le code $\mathcal{C}_i$ obtenu par ces k mots (l'opération peut être réitérée jusqu'à ce que la dimension soit exactement k).
- La matrice H_i est alors une matrice génératrice de taille $(n-k) \times n$ du code dual de $\mathcal{C}_i$.

On peut remarquer que cette construction mène à une matrice publique H_i de grande taille, nous montrerons dans la section 6.5 page 116, une variation de cette construction dans le but de réduire cette taille.

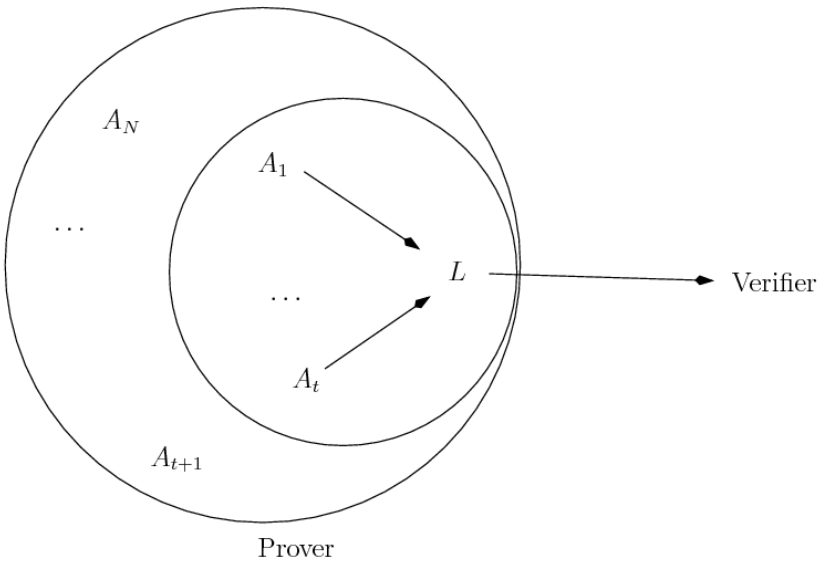


FIGURE 6.1 – Schéma de signature de cercle dans le cas où t signataires sont $A_1, \cdots, A_t$ et le leader est $L = A_1$, pour un groupe de N membres.

6.3.4 Création de la clé publique du groupe (Make-GPK)

Chaque utilisateur possède une matrice de taille $(n-k) \times n$ notée H_i (publique) et un vecteur de n bits noté s_i (secret) de petit poids ω (public) tels que

$$H_i s_i^t = 0.$$

Le problème de trouver s de poids ω est un problème difficile défini plus tôt (section 1.2.2 page 16). Les t signataires choisissent un leader L parmi eux, celui ci envoie un ensemble de matrices publiques $H_1, \cdots, H_N$.

Remarque

Dans le but de simplifier la description du protocole (et pour éviter une double indexation), nous considérons dans la suite que les t signataires correspondent aux t premières matrices H_i ($1 \leq i \leq t$) (mais, plus généralement, leur ordre peut être considéré comme aléatoire dans $\{1, .., N\}$ comme l'ordre dépend de l'ordre des N matrices envoyées par le leader.)

Construction de la clé publique pour le cercle

La clé publique du cercle (RPK (Ring Public Key)) est construite en considérant la matrice H décrite de la manière suivante :

$$H = \begin{pmatrix} H_1 & 0 & 0 & \cdots & 0 \\ 0 & H_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & H_N \end{pmatrix}.$$

H, ω et $H_i, \forall i \in \{1; \ldots; N\}$ sont publiques. Les $s_i, \forall i \in \{1; \ldots; N\}$ sont privés.

6.3.5 Engagement-Challenge-Réponse

Nous décrivons maintenant formellement notre schéma.

Le leader L collecte les engagements donnés à partir des $t-1$ autres signataires, simule les $N-t$ non-signataires et choisit une permutation de n-blocs de taille constante Σ sur N blocs. Pour tous ces engagements le leader L crée les engagements maîtres (master commitments) $\mathcal{C}_1, \mathcal{C}_2$ et $\mathcal{C}_3$ qui sont envoyés au vérifieur V, qui répond en donnant un challenge $b \in \{0, 1, 2\}$. L envoie alors ce challenge aux $t-1$ autres signataires et collecte leurs réponses pour créer la réponse globale pour V. À la réception de la réponse globale, V vérifie qu'elle est correcte en vérifiant les engagements comme dans un schéma de Stern classique.

Rappelons que dans la description du protocole, pour simplifier l'écriture et éviter le double indexage, nous considérons que les t signataires correspondent aux t premières matrices H_i.

Tous les détails du protocole sont donnés dans la figure 6.2.

1. [Engagement]
   - Chaque signataire choisit aléatoirement $y_i \in \mathbb{F}_2^n$ et une permutation aléatoire σ_i de $\{1, 2, \ldots, n\}$ puis envoie à L les engagements $c_{1,i}, c_{2,i}$ et $c_{3,i}$ tels que :

     $$c_{1,i} = h(\sigma_i | H_i y_i^t);\ c_{2,i} = h(\sigma_i(y_i));\ c_{3,i} = h(\sigma_i(y_i \oplus s_i))$$

     où $h(a_1|\cdots|a_j)$ représente le haché de la concaténation de la suite formée par $a_1, \cdots, a_j$.
   - L choisit aléatoirement $N - t$ vecteurs $y_i \in \mathbb{F}^n$ et $N - t$ permutations aléatoires σ_i de $\{1, 2, \ldots, n\}$
   - L fixe le secret s_i des $N-t$ utilisateurs manquants à 0 et calcule les $N-t$ engagements correspondants en choisissant des y_i aléatoires et σ_i ($t + 1 \leq i \leq N$).
   - L choisit une permutation constante aléatoire de n-blocs Σ sur N blocs $\{1, \cdots, N\}$ dans le but d'obtenir les *engagements maîtres* :

     $$C_1 = h(\Sigma | c_{1,1} | \ldots | c_{1,N}), C_2 = h(\Sigma(c_{2,1}, \ldots, c_{2,N})), C_3 = h(\Sigma(c_{3,1}, \ldots, c_{3,N})).$$

   - L envoie C_1, C_2 et C_3 à V.
2. [Challenge] V envoie un challenge $b \in \{0, 1, 2\}$ à L qui envoie b aux t signataires.
3. [Réponse] Soit P_i un des t signataires. La première partie de l'étape se déroule entre chaque signataire et L.
   - Trois possibilités :
     - si $b = 0$: P_i révèle y_i et σ_i.
     - si $b = 1$: P_i révèle $(y_i \oplus s_i)$ (noté $(y \oplus s)_i$) et σ_i.
     - si $b = 2$: P_i révèle $\sigma_i(y_i)$ (noté $(\sigma(y))_i$) et $\sigma_i(s_i)$ (noté $(\sigma(s))_i$).
   - L simule les $N - t$ autres protocoles de Stern avec $s_i = 0$ et $t + 1 \leq i \leq N$.
   - L calcule la réponse pour V (et l'envoie) :
     - si $b = 0$: L construit $y = (y_1, \cdots, y_N)$ et $\Pi = \Sigma \circ \sigma$ (avec $\sigma = (\sigma_1, \cdots, \sigma_N)$) et révèle y et Π.
     - si $b = 1$: L construit $y \oplus s = ((y \oplus s)_1, \cdots, (y \oplus s)_N)$ et révèle $y \oplus s$ et Π.
     - si $b = 2$: L construit $\Pi(y)$ et $\Pi(s)$ et les révèle.
4. [Vérification] Trois possibilités :
   - si $b = 0$: V vérifie que Π est une permutation de n-blocs et que C_1, C_2 ont été obtenus de manière honnête.
   - si $b = 1$: V vérifie que Π est une permutation de n-blocs et que C_1, C_3 ont été obtenus de manière honnête.
   - si $b = 2$: V vérifie que C_2, C_3 ont été obtenus de manière honnête, et que le poids de $\Pi(s)$ est $t\omega$ et que $\Pi(s)$ est formé de N blocs de longueur n et de poids ω ou 0.
5. Réitérer les étapes 1,2,3,4 jusqu'à ce que le niveau de sécurité recherché soit atteint.

FIGURE 6.2 – Le protocole de Stern généralisé

6.4 Sécurité

6.4.1 Notre modèle de sécurité

La sécurité de notre protocole consiste en deux notions d'inforgeabilité et d'anonymat sûres sous l'hypothèse de la difficulté du problème de distance minimale section 1.2.2 page 16 dans le modèle de l'oracle aléatoire.

Pour prouver la première notion, nous prouvons que notre protocole est une preuve de connaissance Honest-Verifier Zero-Knowledge (HVZK). Il a été prouvé dans [46] que tout protocole HVZK peut être transformé en un schéma de signature (voir section 1.1.7 page 8) en remplaçant le challenge par les valeurs des hachés des engagements mis ensemble avec le message à signer. Un tel schéma a été prouvé sûr contre la forge existentielle et contre les attaques à messages choisis adaptatifs dans le modèle de l'oracle aléatoire (voir [109]).

La seconde notion d'anonymat pour notre schéma est dans un contexte à seuil définie de la manière suivante :

Définition 6.4.1 (*Anonymat des signatures de cercle à seuil*)

Soit $R = \{R_k(\cdot,\cdot)\}$ une famille de schémas de signature de cercle à seuil.

Notons $SIG \leftarrow S(G, M, R_k)$ un choix aléatoire parmi les signatures de t utilisateurs d'un groupe G concernant un message M utilisant le schéma de signature de cercle R_k.

R est dit anonyme si pour tout $c > 0$, il existe un K tel que pour tout $k > K$, chaque sous-groupe distinct deux à deux G_1, G_2 de t utilisateurs, chaque message M et chaque famille de circuit probabiliste de taille polynomiale $C = \{C_k(\cdot,\cdot)\}$,

$$Pr((C_k(SIG, G_1, G_2, \mathcal{P}(k)) = G | SIG \leftarrow S(G, M, R_k))) < 1/2 + k^{-c}$$

G étant choisi aléatoirement parmi $\{G_1, G_2\}$, et $\mathcal{P}(k)$ étant l'ensemble de toutes les informations publiques concernant le schéma de signature de cercle.

6.4.2 Sécurité de notre schéma

Nous prouvons d'abord, que notre schéma est HVZK avec une probabilité de triche de $2/3$. Nous commençons d'abord par un lemme.

Lemme 6.4.1

Trouver un vecteur v de longueur nN tel que le poids global de v soit $t\omega$, le poids de v pour chacun des N blocs de longueur n soit 0 ou ω et tel que v ait un syndrome nul pour H, est une instance difficile du problème de distance minimale (Définition 1.2.11 page 18).

Preuve La structure particulière de H (somme directe des H_i de même longueur n) implique que déterminer un tel vecteur de n-blocs de taille nN est exactement équivalent à trouver une solution au problème difficile local qui est de trouver s_i de poids ω tel que $H_i s_i^t = 0$, ce qui n'est pas possible sous notre supposition. □

Théorème 6.4.1

Notre schéma est une preuve de connaissance, avec une probabilité de triche de $2/3$, telle que le groupe de signataires P connaît un vecteur v de longueur nN tel que le poids global de v

est $t\omega$, le poids de v pour chacun des N blocs de longueur n est 0 ou ω et tel que v possède un syndrome nul pour H, sous la supposition MD.

Preuve (idée) Nous devons prouver les trois propriétés usuelles : consistance (completeness), significativité (soundness) et divulgation nulle de connaissance (zero-knowledge). La propriété de consistance est évidente car par exemple pour $b = 0$, la connaissance de y et Π permet de retrouver Σ, σ_i et y_i il est donc possible pour le vérifieur de retrouver tous les c_i et alors l'engagement maître C_1, de même pour C_2. Les cas $b = 1$ et $b = 2$ se traitent de la même manière. La preuve pour la significativité et la divulgation nulle de connaissance suit la preuve originelle du schéma de Stern [130] pour le problème défini dans le lemme précédent, en remarquant que la structure de notre protocole généralisé est la copie du protocole originel avec Σ (figure 6.2 page 113) au lieu de σ (figure 2.2.2 page 23), et avec le fait que l'on vérifie si les réponses $b = 0$ et $b = 1$ dans le protocole la permutation Π est une permutation de n-blocs sur N blocs. □

Remarque

Il est aussi impossible d'avoir de l'information divulguée entre des signataires durant le protocole puisque chaque signataire donne seulement l'information à L et ceci via le schéma classique de Stern qui est à divulgation nulle de connaissance.

Traitons maintenant l'anonymat de notre protocole. L'idée de la preuve est que si un adversaire a une probabilité de posséder plus d'information sur qui est le signataire parmi les N signataires potentiels ou qui ne l'est pas, cela signifie que l'adversaire est capable de connaître avec une meilleure probabilité que $2/3$ qu'un bloc s_i de $s = (s_1, \cdots, s_N)$ de taille n parmi les N tels blocs associés au secret commun s est nul ou non. Mais comme nous avons vu que notre protocole est à divulgation nulle de connaissance (basé sur une légère modification du protocole de Stern), cela signifie qu'un adversaire, qui serait capable d'obtenir de l'information sur le secret s durant l'interaction entre L et V, aurait brisé le caractère zero-knowlege du protocole. De manière formelle nous obtenons :

Théorème 6.4.2

Notre protocole vérifie les propriétés d'anonymat d'un schéma de signature de cercle à seuil.

Preuve Supposons que pour un message donné M, un entier $c > 0$ et deux sous-groupes G_1, G_2 de t utilisateurs il existe une famille de circuit $C = \{C_k(\cdot, \cdot)\}$ telle que pour n'importe quel K il existe $k > K$ tel que :

$$Pr(C_k(SIG, G_1, G_2, \mathcal{P}(k)) = G | SIG \leftarrow S(G, M, R_k)) > 1/2 + k^{-c}.$$

Considérons un utilisateur $P_i \in G_1$ tel que $U \notin G_2$ (un tel utilisateur existe car les deux sous-groupes sont différents), et le circuit suivant :

- Qu'importe le circuit C_k produisant G_1 : la sortie du i-ème (parmi N) bloc de taille n du s associée à la matrice H est nulle.
- Qu'importe le circuit C_k produisant G_2 : la sortie du i-ème (parmi N) bloc de taille n du s associée à la matrice H est nulle.

Un tel circuit devine avec un avantage non négligeable qu'une part d'un secret s associé à la matrice du cercle H est nulle ou non, et alors casse la propriété de zero-knowledge du protocole.

□

6.4.3 Sécurité pratique du schéma de Stern d'après [129]

La sécurité du schéma de Stern repose sur les trois propriétés des codes linéaires aléatoires suivantes :

1. les codes linéaires aléatoires vérifient la borne de Gilbert-Varshamov (voir [85] et section 1.2.2 page 16),
2. Pour n grand, quasiment tous les codes linéaires satisfont la borne de Gilbert-Varshamov [108],
3. Résoudre le problème de décodage par syndrome pour des codes aléatoires est NP-complet [15].

En pratique Stern propose dans [129] d'utiliser des codes de taux $1/2$ et ω juste au dessus de la borne de Gilbert-Varshamov associée au code. Pour de tels codes, le coût exponentiel des meilleurs attaques connues (voir [22] et section 2.3 page 26) est en $\approx \mathcal{O}(n)\frac{\binom{n}{\omega}}{\binom{n-k}{\omega}}$, ce qui donne avec la sécurité d'aujourd'hui (2^{80}) un code avec $n = 634$ et un taux de $1/2$ et $\omega = 69$.

6.5 Variation avec des matrices doublement circulantes

Cette section reprend des idées déjà présentées dans la section 2.2.2 page 24.

Dans la section 6.3, nous décrivons un moyen de créer la matrice publique H_i. Cette méthode, comme dans l'article originel de Stern, donne de grandes tailles de clé publique H_i de $n^2/2$ bits. Il a été proposé récemment dans [49], d'utiliser des matrices doublement circulantes aléatoires au lieu de matrices purement aléatoires.

Une matrice doublement circulante est une matrice de la forme $H_i = (I|C)$ avec C une matrice $n/2 \times n/2$ cyclique et I la matrice identité. En suivant cette idée nous pouvons construire les matrices H_i de la manière suivante :

- considérer $s_i = (a|b)$ avec a et b des vecteurs aléatoires de taille $n/2$ et de poids $\approx \omega/2$,
- considérer la matrice $(A|B)$ obtenue pour A et B des matrices $(n/2 \times n/2)$ carrées obtenues en faisant les $n/2$ décalages cycliques de a et b (chaque ligne de A est un décalage de la ligne précédente, en commençant par la première ligne de a ou de b).

Considérons maintenant le code $\mathscr{C}$ de matrice génératrice $G_i = (A|B)$, la matrice H_i (duale de G_i) peut être vue de la manière suivante $H_i = (I|C)$ telle que H_i est la matrice duale de G_i et C est cyclique comme A et B sont cycliques, et alors H peut être décrite avec seulement sa première ligne. Il est expliqué dans [49] que cette construction ne réduit pas la difficulté du décodage mais décroît clairement la taille de la description de H_i : $n/2$ bits au lieu de $n^2/2$. L'auteur propose $n = 347$.

Nous verrons dans le chapitre 7 que le problème de décodage par syndrome quasi-cyclique est un problème décisionnel NP-complet.

Maintenant, la preuve de sécurité que nous considérons dans ce chapitre peut être adaptée au problème MD-DC. Pour la version généralisée du protocole de Stern que nous avons présentée, nous pouvons prendre n'importe quel type de matrice H_i avec le même type de problème : connaître un vecteur de petit poids associé à H_i (en fait seulement les instances du problème changent).

6.6 Coût et taille

Dans cette section, nous examinons la complexité de notre protocole et la comparons à d'autres protocoles.

6.6.1 Le cas $t = 1$

Ce cas correspond au cas d'un schéma de signature de cercle classique, notre schéma n'est alors pas très attractif en terme de taille de signature puisque nous sommes en $\mathcal{O}(N)$ mais plus précisement en $20 \times N$ (avec 20 ko le coût d'une signature de Stern), malgré cela le protocole de Stern est rapide et notre protocole est plus rapide que tous les autres protocoles pour $N = 2$ ou 3 pour lesquels il existe des applications.

6.6.2 Le cas général

Taille de signature

Il est facile de remarquer que la taille de la signature de notre protocole est en $\mathcal{O}(N)$ (plus précisement en $20 \times N$ - avec 20 ko la taille d'une signature à l'aide du paradigme de Fiat-Shamir (voir section 1.1.7 page 8) appliqué au schéma de Stern). En considérant $N = 100$ et $t = 50$, nous obtenons une signature de 2 Mo, ce qui est assez grand mais utilisable en pratique. Tandis que certains autres schémas comme [20] ne peuvent pas du tout être utilisés, car ils ont des signatures de plus de 2^{60} bits ! Par contre, il existe d'autres protocoles comme [18, 76] qui ont des signatures plus courtes (8Ko ou 25Ko) mais sont plus lents.

Nous pouvons de plus noter qu'avec une telle taille de signature nous pouvons traiter facilement $t \geq 10$ quand les autres protocoles y arrivent difficilement.

Taille de la clé publique

Si nous utilisons la construction doublement circulante décrite dans la section 6.5 page 116, nous obtenons, une clé publique de $347N$ bits qui est plus petite que pour les autres protocoles (basés sur des problèmes de théorie des nombres) d'un facteur 2 ou 3 pour [76] et du même ordre que [18].

Complexité du protocole

Le coût du protocole est N fois le coût d'une signature avec le protocole de Stern classique, c'est-à-dire en $\mathcal{O}(N)$, (plus précisement en $140n^2N$ opérations binaires) et ceci *pour tout* t, quand tous les autres protocoles complètement anonymes de signature de cercle à seuil ont une complexité en $\mathcal{O}(2^tN)$ opérations binaires au moins.

En effet, la principale particularité de notre protocole est que la complexité ne dépend pas de t, ce qui en fait le protocole le plus rapide existant.

6.7 Conclusion

Dans ce chapitre, nous avons présenté un nouveau schéma de signature de cercle à seuil (t-out-of-N threshold ring signature scheme) complètement anonyme basé sur la théorie des codes correcteurs d'erreurs. Notre protocole est une généralisation naturelle du schéma d'authentification de Stern et notre preuve est basée sur la preuve originelle de Stern. Nous avons montré que le choix du poids d'un vecteur particulier rend appliquable ce schéma dans le cas des signatures de cercle et que la notion de groupe ad hoc correspond bien à la notion de somme directe de matrices génératrices. Enfin il est compatible avec la notion de somme de vecteurs de petits poids.

Nous obtenons un protocole complètement anonyme basé sur une preuve de connaissance. Alors que les schémas pré-existants ne pouvaient traiter que des cas où $t \approx \mathcal{O}(\log(N))$, notre schéma permet de traiter n'importe quel t et en particulier $t \approx N/2$.

De plus, le protocole que nous décrivons peut aussi être généralisé. Le fait que notre construction ne soit pas basée sur des problèmes de théorie des nombres mais sur des problèmes de codes représente une alternative intéressante. Nous espérons que cela exhibera le potentiel de la théorie des codes correcteurs d'erreurs en cryptographie à clé publique.

Troisième partie

Chiffrement avec des codes

Chapitre 7

Réduction de la taille des clés

Ce chapitre a donné lieu à l'article suivant [14] :

Reducing Key Lengths with Quasi-cyclic Alternant Codes for Code-Based Cryptography
avec Thierry Berger, Ayoub Otmani et Philippe Gaborit

Ce chapitre ne traitant pas de signatures avec des codes correcteurs d'erreurs se veut auto-suffisant. Les nouvelles notations introduites ici nécessitent que l'on redéfinisse les notions de base.

Dans ce chapitre, nous proposons une construction basée sur l'utilisation de codes quasi-cycliques qui possède deux améliorations par rapport au cryptosystème de Niederreiter originel. Ces améliorations permettent d'éviter la faiblesse induite par l'utilisation de petites permutations reliées aux codes quasi-cycliques. La première amélioration consiste à utiliser pour code de base, une plus grande et plus naturelle famille de codes quasi-cycliques : les codes alternants quasi-cycliques. Cette famille a l'avantage d'avoir un grand nombre de codes avec des paramètres donnés.

La seconde amélioration consiste à considérer pour le code à utiliser, non pas directement la matrice quasi-cyclique d'un code alternant, mais un code grandement raccourci, provenant d'un très grand code alternant quasi-cyclique : on construit des codes de longueur de l'ordre de 1000 à partir de codes de longueur 2^{16} ou 2^{20}. Partir de codes de Reed-Solomon MDS, permet de préserver le caractère MDS et la structure des codes de Reed-Solomon et donc la capacité de correction du code. Cette nouvelle façon de cacher la structure introduit une partie aléatoire dans le choix du code, ce qui rend très difficile la récupération de la structure du code. On peut remarquer que cette approche fonctionne essentiellement et uniquement pour des codes MDS comme les codes de Reed-Solomon utilisés ici après un raccourcissement de codes MDS et après avoir pris un sous-code sur un sous-corps, ce qui ne modifie pas sa capacité de décodage.

Le prix à payer est de partir d'un code sur un plus grand alphabet, mais finalement le coût est d'un ordre deux en terme de vitesse globale de communication. Le fait de considérer un plus petit code induit le problème que l'attaquant ne sait pas ce qui a réduit le code utilisé, et peut donc difficilement utiliser une approche structurelle pour récupérer la permutation cachée. En effet, cette dernière idée permet de cacher la structure de deux façons complémentaires :

1. D'abord l'utilisation d'un code alternant (un sous-code sur un sous-corps) permet de briser la structure classique des codes de Reed-Solomon et permet d'éviter des attaques classiques comme celle de Sidelnikov-Shestakov,

2. D'autre part le fait d'utiliser un code raccourci (équivalent à considérer un code poinçonné en partant de la matrice duale) permet d'éviter les attaques par le biais de la petite matrice de permutation, car ces attaques, pour être efficaces, ont besoin à la fois d'un code (ou un sous-code) et de son code permuté.

La principale contribution de ce chapitre est une construction qui réduit considérablement la taille de la clé publique des cryptosystèmes de McEliece ou Niederreiter (environ 6000 bits) ; mais aussi, sur un point de vue sécurité, la preuve que le décodage de codes quasi-cycliques est un problème NP-complet, problème qui est lié au type de codes que nous utilisons pour notre système.

Le chapitre est organisé comme suit : dans la section 2, nous rappelons les bases de la théorie des codes, dans la section 3 nous considérons les codes particuliers que nous utilisons : les codes alternants quasi-cycliques. La section 4 donne une description du protocole. Enfin, dans la dernière section, nous considérons la sécurité du système.

7.1 Notation et Définitions

Soit $\mathbb{F}_q$ le corps fini à $q = p^s$ éléments avec p un nombre premier et s un entier positif. Pour tout élément β de $\mathbb{F}_q$ et pour tout entier positif n nous notons par $\boldsymbol{\beta}^n$ le vecteur $(1, \beta, \ldots, \beta^{n-1})$. L'ensemble $\{0, \ldots, n-1\}$ est noté J_n. On note $\mathcal{S}_n$ le groupe symétrique d'ordre n. Le poids de Hamming (*weight*) $\mathsf{wt}(\boldsymbol{v})$ d'un vecteur $\boldsymbol{v} = (v_0, \ldots, v_{n-1}) \in \mathbb{F}_q^n$ est le nombre de coordonnées non nulles $0 \leq i \leq n-1$. Le vecteur tout-à-un $(1, \ldots, 1)$ de longueur n est noté $\mathbf{1}_n$. Pour tout $J \subset J_n$ nous définissons $\boldsymbol{v}(J)$ le t-uplet $(v_i)_{i \in J}$ (indice pris par ordre croissant). On peut remarquer que $\boldsymbol{\beta}^n(J)$ représente le t-uplet $(\beta^j)_{j \in J}$.

On définit l'application g_i de $\mathbb{F}_q^n$ dans $\mathbb{F}_q^n$ qui envoie tout $\boldsymbol{v}$ vers $f_i(\boldsymbol{v}) = (v_0^i, \ldots, v_{n-1}^i)$. Le produit de Kronecker d'un vecteur couple de vecteurs $\boldsymbol{u}$ et $\boldsymbol{v} \in \mathbb{F}_q^n$ est :

$$\boldsymbol{u} \otimes \boldsymbol{v} = (u_1 \boldsymbol{v}, \ldots, u_n \boldsymbol{v}).$$

La matrice *identité* d'ordre n est notée I_n. La matrice *transposée* de n'importe quelle matrice $A = [a_{i,j}]$ est notée A^T. $A \otimes B$ est la matrice $[a_{i,j}B]$ pour toutes matrices A et B. Nous noterons $\mathsf{diag}(\boldsymbol{v})$ la matrice carrée dont la diagonale est $\boldsymbol{v}$ et toutes les autres entrées sont nulles. La matrice de Vandermonde $V_k(\boldsymbol{v})$ où $\boldsymbol{v} \in \mathbb{F}_q^n$ (k étant un entier positif) est définie par :

$$V_k(\boldsymbol{v}) = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ v_0 & v_1 & \cdots & v_{n-1} \\ \vdots & \vdots & & \vdots \\ v_0^{k-1} & v_1^{k-1} & \cdots & v_{n-1}^{k-1} \end{bmatrix}.$$

Pour tout n-uplet $\boldsymbol{\lambda}$ d'éléments non nuls de $\mathbb{F}_q$, la matrice de Vandermonde généralisée

$$M_k(\boldsymbol{\lambda}, \boldsymbol{v}) = V_k(\boldsymbol{v}) \times \mathsf{diag}(\boldsymbol{\lambda})$$

est définie comme suit :

$$M_k(\boldsymbol{\lambda}, \boldsymbol{v}) = \begin{bmatrix} \lambda_0 & \lambda_1 & \cdots & \lambda_{n-1} \\ \lambda_0 v_0 & \lambda_1 v_1 & \cdots & \lambda_{n-1} v_{n-1} \\ \vdots & \vdots & & \vdots \\ \lambda_0 v_0^{k-1} & \lambda_1 v_1^{k-1} & \cdots & \lambda_{n-1} v_{n-1}^{k-1} \end{bmatrix}.$$

Pour tout $\sigma \in \mathcal{S}_n$, on notera par $P_\sigma = [p_{i,j}]$ la matrice $n \times n$ telle que $p_{i,j} = 1$ si $\sigma(i) = j$ et $p_{i,j} = 0$ sinon.

Définition 7.1.1

Une transformation monomiale $\Phi_{\sigma,\boldsymbol{\lambda}}$, *où* $\sigma \in \mathcal{S}_n$ *et* $\boldsymbol{\lambda}$ *est un* n*-uplet d'éléments non nuls de* $\mathbb{F}_q$, *est une application de* $\mathbb{F}_q^n$ *dans lui-même définie pour un vecteur* $\boldsymbol{v}$ *de la manière suivante :*

$$\Phi_{\sigma,\boldsymbol{\lambda}}(\boldsymbol{v}) = (\lambda_0 v_{\sigma^{-1}(0)}, \ldots, \lambda_{n-1} v_{\sigma^{-1}(n-1)}).$$

Évidemment nous avons :

$$\Phi_{\sigma,\boldsymbol{\lambda}}(\boldsymbol{v}) = v \times P_\sigma \times \mathsf{diag}(\lambda_0, \ldots, \lambda_{n-1}).$$

On étend trivialement l'action de $\Phi_{\sigma,\boldsymbol{\lambda}}$ aux matrices. Quand la transformation monomiale est seulement une permutation σ nous notons son action par $\sigma(\boldsymbol{v})$. Une transformation monomiale particulière que nous pourrons considérer est le décalage circulaire T_n défini par :

$$T_n(\boldsymbol{v}) = (v_{n-1}, v_0, \ldots, v_{n-2}).$$

La composition de T_n par lui-même j fois est notée T_n^j. Le groupe cyclique généré par T_n est noté $\mathbb{T}_n$. Pour $n = \ell n_0$ avec ℓ et n_0 deux entiers positifs, chaque vecteur $\boldsymbol{v} \in \mathbb{F}_q^n$ peut être vu comme un vecteur $\boldsymbol{v} = (\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{n_0-1})$ avec $\boldsymbol{v}_j \in \mathbb{F}_q^\ell$. Une autre transformation importante, qui sera utile dans ce cas, est la permutation $T_\ell^{[n_0]}$ définie par :

$$T_\ell^{[n_0]}(\boldsymbol{v}) = \left(T_\ell(\boldsymbol{v}_0), \ldots, T_\ell(\boldsymbol{v}_{n_0-1})\right).$$

Nous utiliserons aussi les matrices $\oplus T_\ell^{[n_0]}(\boldsymbol{v})$ qui correspondent à une matrice $\ell \times n$ obtenue en considérant les ℓ décalages circulaires des vecteurs $\boldsymbol{v}$:

$$\oplus T_\ell^{[n_0]}(\boldsymbol{v}) = \begin{bmatrix} \boldsymbol{v} \\ T_\ell^{[n_0]}(\boldsymbol{v}) \\ \vdots \\ \left(T_\ell^{[n_0]}\right)^{\ell-1}(\boldsymbol{v}) \end{bmatrix}$$

Nous étendons cette notation à une matrice arbitraire A en appliquant $\oplus T_\ell^{[n_0]}$ à chaque ligne $\boldsymbol{a}_i$ de A.

Définition 7.1.2 (*Produit en couronne*)

Supposons que $n = \ell n_0$ *et identifions* J_n *à* $J_\ell \times J_{n_0}$. *Soit* $\sigma \in \mathcal{S}_{n_0}$ *et* $\pi_1, \ldots, \pi_{n_0} \in \mathcal{S}_\ell$. *Nous noterons* $(\pi_0, \ldots, \pi_{n_0-1}; \sigma)$ *la permutation de* $J_\ell \times J_{n_0} \simeq J_n$ *telle que pour tout* (i, j) *de* $J_\ell \times J_{n_0}$ *on associe*

$$(\pi_0, \ldots, \pi_{n_0-1}; \sigma)(i, j) = (\pi_j(i), \sigma(j)).$$

L'ensemble de permutations $(\pi_0, \ldots, \pi_{n_0-1}; \sigma)$ *où les* π_i *et* σ *décrivent respectivement* $\mathcal{S}_\ell$ *et* $\mathcal{S}_{n_0}$ *est appelé le* produit en couronne *de* $\mathcal{S}_\ell$ *par* $\mathcal{S}_{n_0}$ *et noté* $\mathcal{S}_\ell \wr \mathcal{S}_{n_0}$

Soit k un entier positif tel que $k \leq n$. Un *code linéaire* $\mathscr{C}$ de *longueur* n et de *dimension* k sur $\mathbb{F}_q$ est un sous-espace vectoriel de $\mathbb{F}_q^n$ de dimension k. J_n est appelé le *support* pour un code de longueur n. Le taux R de $\mathscr{C}$ est le ratio $R = k/n$. Les éléments d'un code sont appelés *mots de code*. Une *matrice génératrice* G de $\mathscr{C}$ est une matrice $k' \times n$ avec $k' \geq k$ dont les lignes définissent le code $\mathscr{C}$. Une matrice de *parité* H de $\mathscr{C}$ est une matrice $r \times n$ avec $r \geq n - k$ telle que pour tout mot de code $\boldsymbol{c} \in \mathscr{C}$ nous avons $H \times \boldsymbol{c}^T = 0$. Le sous-espace linéaire $\mathscr{C}^\perp$ défini par les lignes de H est le code *dual* du code $\mathscr{C}$. Une matrice génératrice G est dite sous *forme échelon réduite* si $G = [I_k|A]$ où A est une matrice $k \times (n-k)$. Une matrice de parité est alors $[-A^T|I_{n-k}]$ où A^T est la matrice transposée de la matrice A. La *distance minimale* d d'un code $\mathscr{C}$ est le poids minimum de tout mot non nul du code. Les paramètres d'un code linéaire $\mathscr{C}$ sur $\mathbb{F}_q$ sont notés par le triplet $[n, k, d]_q$.

Définition 7.1.3 (*Code équivalent*)

Un code $\mathscr{C}'$ est dit (monomialement) équivalent à un code $\mathscr{C}$ s'il existe une transformation monomiale $\Phi_{\sigma,\boldsymbol{\lambda}}$ telle que $\Phi_{\sigma,\boldsymbol{\lambda}}(\mathscr{C}) = \mathscr{C}'$.

Un code $\mathcal{C}$ possède trois paramètres fondamentaux :

1. sa longueur n;
2. sa dimension k;
3. sa redondance $r = n - k$.

Chacun de ces paramètres a une interprétation naturelle pour les codes linéaires, mais les six modifications basiques que nous allons présenter, ne se restreignent pas aux codes linéaires. Chacune de ces transformations fixe un paramètre et accroît ou décroît les deux autres paramètres.

(i)	*augmentation*	pour n fixé,	k augmente	r diminue
(ii)	*expurgation*	pour n fixé,	k diminue	r augmente
(iii)	*extension*	pour k fixé,	n augmente	r augmente
(iv)	*poinçonnement*	pour k fixé,	n diminue	r diminue
(v)	*allongement*	pour r fixé,	n augmente	k augmente
(vi)	*raccourcissement*	pour r fixé,	n diminue	k diminue

FIGURE 7.1 – Techniques pour modifier un code

Ces six techniques se retrouvent naturellement en trois paires, chaque membre d'une paire est le processus inverse de l'autre. Comme la redondance d'un code est la *dimension de son dual*, chaque technique possède aussi sa technique duale.

augmentation et expurgation

En augmentant ou en expurgeant un code, nous gardons sa longueur fixée mais sa dimension et sa redondance varient. Quand l'on *augmente* un code $\mathcal{C}$, on lui ajoute des mots de code. Le processus inverse d'expurgation d'un code consiste à lui enlever des mots de code. On peut remarquer que l'augmentation peut faire décroître la distance minimale tandis que l'expurgation ne la fera pas décroître, et peut même l'augmenter. Un code linéaire est facilement augmenté en ajoutant des lignes à la matrice génératrice et expurgé en en otant. Une façon classique d'augmenter un code linéaire

est d'ajouter le vecteur tout-à-un à sa matrice génératrice. Bien sûr, dans certains cas, ce vecteur appartient déjà au code et cette action n'a aucun effet. Augmenter la taille d'un code linéaire est le même processus que de réduire la taille de son dual, donc ces deux techniques sont duales l'une de l'autre aussi bien qu'inverses.

Si l'on note $\mathbb{F}$ un sous-corps d'un corps $\mathbb{K}$ et $\mathcal{C}$ un code sur $\mathbb{F}$, alors nous pouvons expurger $\mathcal{C}$ en prenant seulement ces mots de code dont les paramètres appartiennent à $\mathbb{K}$. Ce sous-code sur un sous-corps hérite de nombreuses propriétés du code originel et peut même avoir de meilleures propriétés.

extension et poinçonnement

En étendant ou poinçonnant un code, sa dimension reste fixée mais sa longueur ainsi que sa redondance varient. Quand l'on *étend* un code, on lui ajoute un symbole de redondance supplémentaire. L'application inverse, à savoir le poinçonnement, consiste à supprimer des symboles de redondance. Poinçonner peut faire décroître la distance minimale, mais étendre ne la diminue pas, cela peut au contraire l'augmenter. Pour étendre un code linéaire on ajoute des colonnes à sa matrice génératrice, et pour poinçonner le code, on enlève des colonnes de cette matrice.

allongement et raccourcissement

En allongeant ou raccourcissant un code, sa redondance est fixée mais sa longueur ainsi que sa dimension varient. Quand on allonge un code $\mathcal{C}$, on augmente sa longueur et on ajoute des mots au code $\mathcal{C}$. Le processus inverse, le raccourcissement, enlève des mots de code et supprime des positions de coordonnées. On peut voir ces opérations comme combinaisons des techniques précédentes. Allonger peut être vu comme une extension suivie d'une augmentation et raccourcir peut être fait en expurgeant puis en poinçonnant. Comme les deux opérations qui constituent l'allongement et le raccourcissement ont tendance à avoir des influences opposées sur la distance minimale, leurs effets sur la distance minimale dépendent de la situation.

Pour un code linéaire $\mathcal{C}$, l'allongement correspond à ajouter des colonnes à une matrice génératrice de $\mathcal{C}$ en lui ajoutant des nouvelles colonnes (extension) et le même nombre de lignes (augmentation). Une méthode standard consiste à ajouter à la matrice génératrice originelle une colonne finale entièrement nulle et ajouter une ligne qui n'est pas nulle sur cette nouvelle colonne, par exemple le vecteur tout-à-un.

Pour raccourcir, nous pouvons inverser la construction précédente. Nous devons tout d'abord trouver une matrice génératrice du code allongé qui a une ligne dans laquelle la dernière colonne est non nulle. On supprime alors cette ligne (expurgation) ainsi que la dernière colonne (poinçonnement), on obtient alors une matrice génératrice du code originel.

Nous nous intéresserons à une technique qui permet à la fois de diminuer n et k, dans le but de réduire les paramètres publics de McEliece, nous optons donc pour le raccourcissement.

Définition 7.1.4 (*Code raccourci*)

Soit $\mathscr{C}$ un code de longueur n défini par une matrice de parité H. Considérons un sous-ensemble J' de J_n de cardinal $n' < n$ et définissons $H(J')$ la matrice obtenue en supprimant les colonnes de H qui ne sont pas dans J'. La matrice de parité $H(J')$ définie le code raccourci *$\mathscr{C}(J')$ sur J'.*

Définition 7.1.5 (*Sous-code sur un sous-corps*)

Soit $\mathscr{C}$ un code linéaire sur $\mathbb{F}_q$ et considérons un sous-corps $\mathbb{F}_{q'}$ de $\mathbb{F}_q$. Le sous-code sur le sous-corps *$\mathscr{C}'$ sur $\mathbb{F}_{q'}$ est l'ensemble de tous les mots du code $\mathscr{C}$ qui ont leurs composantes dans $\mathbb{F}_{q'}$.*

Évidemment un code qui est monomialement équivalent à un code $\mathscr{C}[n, k, d]$ a les mêmes paramètres. En ce qui concerne la distance minimale d'un sous-code sur un sous-corps d'un code raccourci $\mathscr{C}$, celle ci est toujours supérieure ou égale à d.

Supposons que r est le degré de $\mathbb{F}_q$ sur $\mathbb{F}_{q'}$ et soit $\Omega = \{\omega_1, \ldots, \omega_r\}$ une base arbitraire de $\mathbb{F}_q$ sur $\mathbb{F}_{q'}$. Considérons une $(n-k) \times n$ matrice de parité H d'un code $\mathscr{C}$. Une méthode pour construire un sous-code sur un sous-corps $\mathbb{F}_{q'}$ est de décomposer chaque entrée de H dans la base Ω. La nouvelle matrice de parité possède alors $m \times (n-k)$ lignes et on obtient la proposition suivante.

Proposition 7.1.1

La dimension k' d'un sous-code sur un sous-corps d'un code de dimension k et de longueur n vérifie $k' \geq n - r(n-k)$ avec r le degré de l'extension.

Définition 7.1.6 (*Code cyclique*)

Un code linéaire $\mathscr{C}$ de longueur n est cyclique *si et seulement si $T_n(\mathscr{C}) = \mathscr{C}$. Si $n = \ell n_0$ avec $n_0 \geq 2$ alors le code $\mathscr{C}$ est* quasi-cyclique d'ordre n_0 *si et seulement si $T_\ell^{[n_0]}(\mathscr{C}) = \mathscr{C}$.*

Proposition 7.1.2

Le dual $\mathscr{C}^\perp$ d'un code quasi-cyclique $\mathscr{C}$ d'ordre n_0 et de longueur $n = \ell n_0$ est aussi quasi-cyclique d'ordre n_0.

Soit $\mathbb{F}_q[x]$ l'ensemble des polynômes univariés sur $\mathbb{F}_q$. Un code cyclique $\mathscr{C}$ de longueur n peut aussi être vu comme un idéal de l'anneau quotient $R_n[x] = \mathbb{F}_q[x]/(x^n - 1)$. Alors $\mathscr{C}$ a un unique polynôme unitaire $\boldsymbol{g}(x)$ de degré $n-k$, où k est la dimension, qui génère le code $\mathscr{C}$ et tel que $\boldsymbol{g}(x)$ divise $x^n - 1$. Le polynôme $\boldsymbol{g}(x)$ est appelé le *polynôme générateur*.

De manière similaire, un code quasi-cyclique $\mathscr{C}$ de longueur $n = n_0\ell$ et d'ordre n_0 est vu comme un sous-module d'un $R_\ell[x]$-module $(R_\ell[x])^{n_0}$ avec $R_\ell[x]$ l'anneau quotient $\mathbb{F}_q[x]/(x^\ell - 1)$. La caractérisation implique que le code $\mathscr{C}$ est définie par une matrice génératrice $\boldsymbol{G}(x) = [\boldsymbol{g}_{i,j}(x)]$ avec $1 \leq j \leq n_0$ et où $\boldsymbol{g}_{i,j}(x)$ appartient à $R_\ell[x]$.

Une autre façon d'interpréter $\boldsymbol{G}(x)$ est d'associer à chaque polynôme $\boldsymbol{g}_{i,j}(x)$ une matrice $\ell \times \ell$ appelée matrice *circulante* telle que $\boldsymbol{G}(x)$ soit identifié à une matrice *quasi-circulante*.

Définition 7.1.7 (*Matrice circulante*)

Une matrice circulante M *définie par le polynôme* $\boldsymbol{m}(x) = m_0 + m_1 x + \cdots + m_{\ell-1}x^{\ell-1} \mod (x^\ell - 1)$ *est une matrice $\ell \times \ell$ obtenue par permutation circulaire de la première ligne :*

$$M = \begin{pmatrix} m_0 & m_1 & \cdots & m_{\ell-1} \\ m_{\ell-1} & m_0 & \cdots & m_{\ell-2} \\ \vdots & \ddots & \ddots & \vdots \\ m_1 & \cdots & m_{\ell-1} & m_0 \end{pmatrix}. \tag{7.1}$$

Une matrice quasi-circulante $A = [A_{i,j}]$ *est une matrice par blocs où chaque bloc $A_{i,j}$ est une matrice $\ell \times \ell$ circulante.*

Il est important de remarquer que les matrices $\ell \times \ell$ circulantes forment un ensemble stable sous l'action de multiplication, addition et transposition. De plus ces opérations mènent exactement au même résultat lorsque nous traitons les polynômes associés. Ces propriétés sont aussi vérifiées par les matrices quasi-circulantes.

Nous résumons ces propriétés dans les propositions suivantes :

Proposition 7.1.3

Soit $\mathfrak{C}_\ell$ l'ensemble des matrices binaires $\ell \times \ell$ circulantes. L'application qui envoie une matrice $\ell \times \ell$ circulante M vers $\boldsymbol{m}(x)$ est un isomorphisme d'anneaux entre $(\mathfrak{C}_\ell, +, \times)$ et $(R_\ell, +, \times)$.

Proposition 7.1.4

Soit $\mathfrak{B}_\ell$ l'ensemble des matrices quasi-circulantes. Soit $\mathfrak{M}(R_\ell)$ l'ensemble des matrices dont les coefficients sont dans R_ℓ. L'application $G \mapsto \mathbf{G}(x)$ réalise un isomorphisme d'anneaux entre $\mathfrak{B}_\ell$ et $\mathfrak{M}(R_\ell)$.

Une méthode, pour obtenir des matrices quasi-circulantes à partir d'une matrice génératrice $k \times n$ arbitraire G' d'un code quasi-cyclique $\mathscr{C}$ d'ordre n_0, est d'appliquer le processus suivant. On débute avec la première ligne $\boldsymbol{g}'_1$ de G' et on calcule $G = \oplus T_\ell^{[n_0]}(\boldsymbol{g}'_1)$. L'étape suivante consiste à considérer la première ligne $\boldsymbol{g}_{i_2}$ qui n'est pas dans l'espace vectoriel engendré par $\oplus T_\ell^{[n_0]}(\boldsymbol{g}'_1)$. On fait la somme des lignes de $\oplus T_\ell^{[n_0]}(\boldsymbol{g}'_{i_2})$ à G pour obtenir une autre matrice $2\ell \times n$ quasi-circulante. On continue le processus jusqu'à atteindre la dernière colonne de G'. La matrice résultante G engendre évidemment le code $\mathscr{C}$ et sera de taille $\ell k_0 \times n$ pour un entier k_0 tel que $\ell k_0 \geq k$. Cette méthode s'applique aussi pour obtenir des matrices de parité quasi-circulante.

Dans la suite, nous considérerons une méthode pour construire un sous-code quasi-cyclique de dimension k' avec $k - \ell \leq k' \leq k - 1$ à partir d'un code quasi-cyclique $\mathscr{C}$ de dimension k. Soit H une matrice de parité arbitraire d'un code $\mathscr{C}$. L'idée est de prendre aléatoirement un n-uplet $\boldsymbol{a}$ qui n'appartient pas à $\mathscr{C}^\perp$. La matrice de parité $\tilde{H} = \begin{bmatrix} \oplus T_\ell^{[n_0]}(\boldsymbol{a}) \\ H \end{bmatrix}$ définie alors un sous-code de $\mathscr{C}$ avec la dimension attendue.

7.2 Codes alternants quasi-cycliques

7.2.1 Codes de Reed-Solomon généralisés

Définition 7.2.1

Soit α un élément primitif de $\mathbb{F}_q$ et supposons que $n = q - 1$. Le code de Reed-Solomon $\mathscr{R}_k(\boldsymbol{\alpha}^n)$ est le code de longueur n sur $\mathbb{F}_q$ définie par la matrice génératrice $V_k(\boldsymbol{\alpha}^n)$.

Les codes de Reed-Solomon représentent une classe importante de codes cycliques. Il est connu que $\boldsymbol{g}(x) = \prod_{i=1}^{d-1}(x-\alpha^i)$ est un polynôme générateur pour $\mathscr{R}_k(\boldsymbol{\alpha}^n)$ où $d = n-k+1$ est sa distance minimale. Une matrice de parité pour le code $\mathscr{R}_k(\boldsymbol{\alpha}^n)$ est la matrice H définie comme suit :

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & \cdots & (\alpha^2)^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{d-1} & (\alpha^{d-1})^2 & \cdots & (\alpha^{d-1})^{n-1} \end{bmatrix}.$$

On peut remarquer que $H = M_{d-1}(\boldsymbol{\alpha}^n, \boldsymbol{\alpha}^n)$. Les codes de Reed-Solomon représentent un cas particulier d'une classe légèrement plus générale de codes appelés les codes de Reed-Solomon Généralisés (*GRS*).

Définition 7.2.2

Soit $\boldsymbol{\lambda}$ un n-uplet d'éléments non nuls de $\mathbb{F}_q$ avec $n < q$ et soit $\boldsymbol{v}$ un n-uplet d'éléments distincts de $\mathbb{F}_q$. Le code de Reed-Solomon Généralisé $\mathscr{G}_k(\boldsymbol{\lambda}, \boldsymbol{v})$ sur $\mathbb{F}_q$ de dimension k est le code défini par la matrice de parité $M_{n-k}(\boldsymbol{\lambda}, \boldsymbol{v})$.

Il est connu que le dual d'un code GRS est aussi un code GRS. On a alors une définition alternative pour les codes de Reed-Solomon Généralisés, il s'agit de considérer la matrice de Vandermonde généralisée comme matrice génératrice. De plus, les codes de Reed-Solomon Généralisés sont monomialement équivalent aux codes de Reed-Solomon lorsque $n = q - 1$.

Proposition 7.2.1

Soit $n = q - 1$. Pour tout $\mathscr{G}_k(\boldsymbol{\lambda}, \boldsymbol{v})$ il existe σ de $\mathcal{S}_n$ tel que

$$\mathscr{G}_k(\boldsymbol{\lambda}, \boldsymbol{v}) = \Phi_{\sigma,\boldsymbol{\lambda}}(\mathscr{R}_k(\boldsymbol{\alpha}^n)).$$

7.2.2 Construction de codes de Reed-Solomon généralisés quasi-cycliques

Il est possible de construire des codes quasi-cycliques d'ordre n_0 si l'on suppose que $n = q - 1 = \ell n_0$. Il est possible de réordonner le support $\boldsymbol{\alpha}^n$. Tout d'abord posons $\beta = \alpha^{n_0}$. Rappelons que α est un élément primitif de $\mathbb{F}_q$ et que l'on a $\beta^\ell = 1$. Remarquons que les coordonnées de tous les vecteurs $\alpha^j\boldsymbol{\beta}^\ell = (\alpha^j, \alpha^j\beta, \ldots, \alpha^j\beta^{\ell-1})$ lorsque j décrit l'ensemble $\{0, \ldots, n_0-1\}$ forment une partition de $\mathbb{F}_q - \{0\}$. Par conséquent $\boldsymbol{\alpha}^{n_0} \otimes \boldsymbol{\beta}^\ell$ peut être vu comme un élément de $\mathbb{F}_q^n$ où toutes ces coordonnées sont distinctes et non nulles. Cela servira de nouveau support aux codes de longueur n.

Proposition 7.2.2

Le code de Reed-Solomon $\mathscr{R}_k(\boldsymbol{\alpha}^{n_0} \otimes \boldsymbol{\beta}^\ell)$ est un code quasi-cyclique d'ordre n_0.

Preuve On peut remarquer que $\mathscr{R}_k(\boldsymbol{\alpha}^{n_0} \otimes \boldsymbol{\beta}^\ell)$ est quasi-cyclique d'ordre n_0 si et seulement si chaque ligne de sa matrice génératrice après transformation par $T_\ell^{[n_0]}$ est toujours un mot du code. On peut vérifier facilement que :

$$V_k(\boldsymbol{\alpha}_{n_0} \otimes \boldsymbol{\beta}^\ell) = [V_k(\boldsymbol{\beta}^\ell)|V_k(\alpha\boldsymbol{\beta}^\ell)|\dots|V_k(\alpha^{n_0-1}\boldsymbol{\beta}^\ell)]$$

est une matrice génératrice pour $\mathscr{R}_k(\boldsymbol{\alpha}^{n_0} \otimes \boldsymbol{\beta}^\ell)$.

Ainsi, si $\boldsymbol{v}_i = (\boldsymbol{v}_{i,0}, \dots, \boldsymbol{v}_{i,n_0-1})$ est la i-ème ligne de $V_k(\boldsymbol{\alpha}_{n_0} \otimes \boldsymbol{\beta}_\ell)$ et $\boldsymbol{v}_{i,j}$ la i-ème ligne de $V_k(\alpha^j\boldsymbol{\beta}_\ell)$ alors nous pouvons voir que $T_\ell(\boldsymbol{v}_{i,j}) = \beta^{i(\ell-1)}\boldsymbol{v}_{i,j}$, et donc $T_\ell^{[n_0]}(\boldsymbol{v}_i) = \beta^{i(\ell-1)}\boldsymbol{v}_i$ ce qui termine la preuve. □

Définition 7.2.3

Soit $\boldsymbol{a} = (a_0, \dots, a_{n_0-1})$ un n_0-uplet d'éléments non nul de $\mathbb{F}_q$, m un entier $0 \leq m \leq \ell - 1$ et θ un élément de $\mathbb{T}_\ell \wr \mathcal{S}_{n_0}$. Nous noterons $\mathscr{Q}_k(\theta, \boldsymbol{a}, m)$ le code :

$$\Phi_{\theta,\boldsymbol{a}\otimes f_m(\boldsymbol{\beta}^\ell)}(\mathscr{R}_k(\boldsymbol{\alpha}^{n_0} \otimes \boldsymbol{\beta}^\ell)).$$

Il est intéressant de mentionner qu'un code qui est monomialement équivalent à un code quasi-cyclique n'est pas nécessairement quasi-cyclique. Toutefois, il est possible d'obtenir un code quasi-cyclique en réordonnant le support.

Lemme 1

Soit $\mathscr{C}$ un code quasi-cyclique d'ordre n_0 et de longueur $n = \ell n_0$. Pour toute permutation θ de $\mathbb{T}_\ell \wr \mathcal{S}_{n_0}$, le code $\theta(\mathscr{C})$ est quasi-cyclique d'ordre n_0.

Proposition 7.2.3

$\mathscr{Q}_k(\theta, \boldsymbol{a}, m)$ est un code quasi-cyclique d'ordre n_0.

Preuve Posons $\mathscr{C} = \mathscr{R}_k(\boldsymbol{\alpha}^{n_0} \otimes \boldsymbol{\beta}^\ell)$. Nous pouvons tout d'abord observer qu'à l'aide du Lemme 1, la permutation θ peut être la permutation identité id_n parce que si $\mathscr{C}' = \Phi_{\mathsf{id}_n,\boldsymbol{a}\otimes f_m(\boldsymbol{\beta}^\ell)}(\mathscr{C})$ est quasi-cyclique alors $\theta(\mathscr{C}') = \Phi_{\theta,\boldsymbol{a}\otimes f_m(\boldsymbol{\beta}^\ell)}(\mathscr{C})$ est aussi quasi-cyclique. Mais la matrice de parité d'un code $\mathscr{C}'$ est $M_{n-k}(\boldsymbol{a} \otimes f_m(\boldsymbol{\beta}^\ell), \boldsymbol{\alpha}_{n_0} \otimes \boldsymbol{\beta}^\ell) = [B_0|\cdots|B_{n_0-1}]$ avec $B_j = M_{n-k}(a_j f_m(\boldsymbol{\beta}^\ell), \alpha^j\boldsymbol{\beta}^\ell)$:

$$B_j = \begin{bmatrix} a_j & a_j\beta^m & \cdots & a_j\left(\beta^{\ell-1}\right)^m \\ a_j\alpha^j & a_j\beta^m\alpha^j\beta & \cdots & a_j\left(\beta^{\ell-1}\right)^m\alpha^j\beta^{\ell-1} \\ a_j\left(\alpha^j\right)^2 & a_j\beta^m\left(\alpha^j\beta\right)^2 & \cdots & a_j\left(\beta^{\ell-1}\right)^m\left(\alpha^j\beta^{\ell-1}\right)^2 \\ \vdots & \vdots & & \vdots \\ a_j\left(\alpha^j\right)^{n-k-1} & a_j\beta^m\left(\alpha^j\beta\right)^{n-k-1} & & a_j\left(\beta^{\ell-1}\right)^m\left(\alpha^j\beta^{\ell-1}\right)^{n-k-1} \end{bmatrix}.$$

Alors, si nous notons par $\boldsymbol{b}_i$ la i-ème ligne de $M_{n-k}(\boldsymbol{a} \otimes f_m(\boldsymbol{\beta}^\ell), \boldsymbol{\alpha}^{n_0} \otimes \boldsymbol{\beta}^\ell)$ alors nous avons $T_\ell^{[n_0]}(\boldsymbol{b}_i) = \beta^{(i+m)(\ell-1)}\boldsymbol{b}_i$ ce qui prouve que $\mathscr{C}'^\perp$ est un code quasi-cyclique. La Proposition 7.1.2 implique que le code $\mathscr{C}'$ est aussi quasi-cyclique ce qui termine la preuve. □

On peut remarquer qu'un code raccourci d'un code quasi-cyclique n'est pas nécessairement quasi-cyclique. Toutefois, il est possible d'obtenir un code quasi-cyclique raccourci si pour un certain $0 \leq j \leq n_0 - 1$ un bloc entier de colonnes indexées par $\alpha^j \boldsymbol{\beta}^\ell$ est supprimé.

Définition 7.2.4

Soit n_0^* un entier vérifiant $0 < n_0^* < n_0$ et soit J un sous ensemble de J_{n_0} de cardinal n_0^*. $\mathscr{Q}_k(\theta, \boldsymbol{a}, m, J)$ est le code raccourci de $\mathscr{Q}_k(\theta, \boldsymbol{a}, m)$ sur $\boldsymbol{\alpha}^{n_0}(J) \otimes \boldsymbol{\beta}^\ell$ où $\boldsymbol{\alpha}^{n_0}(J)$ est le vecteur $(\alpha^i)_{i \in J}$.

La description que nous donnions jusqu'à présent est de considérer des codes quasi-cycliques à travers leurs matrices composées d'éléments du corps $\mathbb{F}_q$. Comme nous l'avons vu, il est plus simple de manipuler de tels codes au moyen de matrices polynomiales. Cette approche permet de représenter des codes de manière plus compacte puisqu'une matrice circulante nécessite seulement un polynôme pour être complètement décrite. Considérons une matrice $k_0 \times n_0$ de parité polynomiale $\boldsymbol{M}(x)$ d'un code de Reed-Solomon $\mathscr{R}_k(\boldsymbol{\alpha}^{n_0} \otimes \boldsymbol{\beta}^\ell)$ où pour un gain en simplicité $k = \ell k_0$. Alors une matrice de parité quasi-circulante $\boldsymbol{H}(x)$ de $\mathscr{Q}_k(\theta, \boldsymbol{a}, m)$ est :

$$\begin{bmatrix} \Phi_{\pi_0, f_m(\boldsymbol{\beta}^\ell)}(\boldsymbol{m}_{0,0})(x) & \cdots & \Phi_{\pi_{n_0-1}, f_m(\boldsymbol{\beta}^\ell)}(\boldsymbol{m}_{0,n_0-1})(x) \\ \vdots & \vdots & \\ \Phi_{\pi_0, f_m(\boldsymbol{\beta}^\ell)}(\boldsymbol{m}_{k_0-1,0})(x) & \cdots & \Phi_{\pi_{n_0-1}, f_m(\boldsymbol{\beta}^\ell)}(\boldsymbol{m}_{k_0-1,n_0-1})(x) \end{bmatrix} \times P_\sigma \times \mathsf{diag}(a_0, a_1, \ldots, a_{n_0-1})$$

Finalement remarquons que pour supprimer la j-ème colonne de $\boldsymbol{H}(x)$ on doit seulement choisir $a_j = 0$. Ce fait montre qu'une matrice de parité polynomiale de $\mathscr{Q}_k(\theta, \boldsymbol{a}, m, J)$ est obtenue en supprimant les colonnes de $\boldsymbol{H}(x)$ qui ne sont pas dans J.

7.2.3 Codes alternants

Définition 7.2.5 (*Code Alternant*)

Soit $\mathbb{F}_{q'}$ un sous-corps de $\mathbb{F}_q$ et soit $\mathscr{G}_k(\boldsymbol{\lambda}, \boldsymbol{v})$ un code de Reed-Solomon Généralisé de dimension k sur $\mathbb{F}_q$. Le code alternant *$\mathscr{A}_{k'}(\boldsymbol{\lambda}, \boldsymbol{v})$ sur $\mathbb{F}_{q'}$ est le sous-code sur le sous-corps de $\mathscr{G}_k(\boldsymbol{\lambda}, \boldsymbol{v})$ sur $\mathbb{F}_{q'}$ de dimension k'.*

Il est clair qu'un code alternant est quasi-cyclique lorsque le code de Reed-Solomon Généralisé associé est quasi-cyclique. Cette propriété nous amène à donner la définition suivante.

Définition 7.2.6

Soit $\boldsymbol{a} = (a_0, \ldots, a_{n_0-1})$ un n_0-uplet d'éléments de $\mathbb{F}_q$, m un entier tel que $0 \leq m \leq \ell - 1$ et θ un élément de $\mathbb{T}_\ell \wr \mathcal{S}_{n_0}$.

$\mathscr{A}_{k'}(\theta, \boldsymbol{a}, m)$ représente le sous-code sur le sous-corps de $\mathscr{Q}_k(\theta, \boldsymbol{a}, m)$ de dimension k', et $\mathscr{A}_{k'}(\theta, \boldsymbol{a}, m, J)$ est le code raccourci sur $J \subset J_{n_0}$ de $\mathscr{A}_k(\theta, \boldsymbol{a}, m)$ et de dimension k'.

7.2.4 Décoder les codes alternants

Pour décoder les codes alternants, il suffit de décoder les codes de Reed-Solomon associés. Commençons avec un $[n, k, d]$ code de Reed-Solomon. Il y a plusieurs façons de décoder un code de Reed-Solomon. Dans notre cas, nous préférons utiliser l'approche de Berlekamp-Welch qui a

une complexité en $\mathcal{O}(n^2)$ et même en $\mathcal{O}(nd)$ si l'on utilise des techniques de réencodage [137]. Cette approche est moins coûteuse que l'approche basique de Berlekamp-Massey, sans algorithme de recherche de racine, et seulement une étape d'interpolation. En particulier, le fait que l'on parte d'un code de Reed-Solomon plus grand et plus long qui est fortement raccourci, ne touche que les opérations sur le corps qui sont effectuées dans un corps plus grand, mais pas les paramètres de l'étape d'interpolation, qui est seulement fonction de n et d.

7.3 Une nouvelle variante du cryptosystème de McEliece basée sur des codes alternants quasi-cycliques

7.3.1 Description

Un cryptosystème à clé publique utilise une *fonction à sens unique à trappe* E qui servira de *fonction de chiffrement*. Le calcul de l'inverse E^{-1} aussi appelée *fonction de déchiffrement* est seulement possible grâce à un secret K. Ce concept de fonction à sens unique à trappe est à la base de la cryptographie asymétrique dans laquelle la *clé privée* est K et la clé publique est E. De manière plus précise, un cryptosystème à clé publique doit fournir trois algorithmes à savoir *génération de clés*, *chiffrement* et *déchiffrement*.

La *génération de clés* est un algorithme PPT qui prend en entrée 1^κ, où $\kappa \geq 0$ est le paramètre de sécurité, et produit en sortie une paire (pk, sk) de clés publique et privée. La *génération de clés* spécifie également un *espace des messages* fini M_{pk}.

Le *chiffrement* est un algorithme PPT qui prend en entrée 1^κ, pk et un mot $\boldsymbol{x}$ de M_{pk} et produit en sortie un mot $\boldsymbol{c}$.

Le *déchiffrement* est un algorithme PPT qui prend en entrée 1^κ, sk et un mot $\boldsymbol{c}$ et produit en sortie un mot $\boldsymbol{x}$. Le cryptosystème doit satisfaire la propriété de *consistance* (*correctness*) qui permet de dire que le déchiffrement est l'inverse à gauche du chiffrement.

La variante que nous proposons suit la proposition originelle du cryptosystème de Niederreiter à l'exception du fait que l'on remplace les codes de Goppa binaires par des sous-codes de codes alternants quasi-cycliques. Nous supposerons pour la suite que $q = 2^s$ avec $q - 1 = \ell N_0$. On note $\mathfrak{G}_{\mathbb{F}_q}(\ell, N_0, t)$ la classe de tous les codes de Reed-Solomon quasi-cycliques d'ordre N_0, de longueur $N = \ell N_0$ et de taux R sur $\mathbb{F}_q$ capables de corriger jusqu'à t erreurs. Rappelons que la dimension K d'un code dans $\mathfrak{G}_{\mathbb{F}_q}(\ell, N_0, t)$ est $K = N - 2t$. Dans un souci de simplicité, nous supposons que $K = \ell K_0$. L'idée est de choisir au hasard un ensemble $J \subset J_{N_0}$ de cardinal $n_0 < N_0$ et nous considérerons le code de Reed-Solomon raccourci $\mathscr{R} = \mathscr{R}_K(\boldsymbol{\alpha}^{N_0}(J) \otimes \boldsymbol{\beta}^\ell)$ sur $\boldsymbol{\alpha}^{N_0}(J)$. Nous choisissons au hasard un n_0-uplet $\boldsymbol{a} = (a_0, \ldots, a_{n_0-1})$ d'éléments de $\mathbb{F}_q$, un entier m de J_ℓ et une permutation θ de $\mathbb{T}_\ell \wr \mathcal{S}_{n_0}$. Nous choisissons $q' = 2^{s'}$ avec $s = rs'$ et nous considérons une base Ω de $\mathbb{F}_q$ sur $\mathbb{F}_{q'}$. On pose $\mathscr{C}$ le code alternant de $\Phi_{\theta, \boldsymbol{a} \otimes f_m(\boldsymbol{\beta}^\ell)}(\mathscr{R})$ sur $\mathbb{F}_{q'}$. On peut remarquer que la dimension k est alors $\geq n - 2rt$.

La clé publique est alors la matrice de parité polynomiale $\boldsymbol{H}(x)$ de $\mathscr{C}$ et la clé privée est $(\boldsymbol{a}, \theta, m, J)$. L'ensemble des n-uplets sur $\mathbb{F}_q$ de poids t est noté $\mathbb{F}_{q,t}$. La figure 7.3 donne plus de détails sur l'algorithme de génération de clés. Posons $W_f(q, n, k, t)$ le travail du meilleur algorithme pour décoder t erreurs d'un code linéaire arbitraire de longueur n, de dimension k défini sur $\mathbb{F}^q$.

- Choisissons n, k et t tels que $W_f(2, n, k, t) \geq 2^\kappa$.
- Choisissons N_0 et s tels que $n_0 < N_0 < 2^\kappa - 1$ et $\ell N_0 = 2^s - 1$.
- Posons $n = \ell n_0$, $N = \ell N_0$, $q = 2^s$ et $K = N - 2t$.
- Soit α un élément primitif de $\mathbb{F}_q$ et posons $\beta = \alpha^{n_0}$.
- Prenons aléatoirement un sous-ensemble $J \subset J_{N_0}$ de cardinal n_0.
- Supprimons les colonnes de $M_{2t}(\boldsymbol{\alpha}^{N_0} \otimes \boldsymbol{\beta}^\ell, \boldsymbol{\alpha}^{N_0} \otimes \boldsymbol{\beta}^\ell)$ qui ne sont pas dans $\boldsymbol{\alpha}^{n_0}(J) \otimes \boldsymbol{\beta}^\ell$.
- Notons M le résultat.
- Prenons aléatoirement $(\boldsymbol{a}, \theta, m)$ dans $\mathbb{F}_q^{n_0} \times \mathbb{T}_\ell \wr \mathcal{S}_{n_0} \times J_\ell$.
- Soit $s' > 2$ tel que $s = rs'$ et posons $q' = 2^{s'}$.
- Soit H'' la décomposition de $\Phi_{\theta, \boldsymbol{a} \otimes f_m(\boldsymbol{\beta}^\ell)}(M)$ dans la base Ω de $\mathbb{F}_q$ sur $\mathbb{F}_{q'}$.
- Transformons H'' en une matrice quasi-circulante H'. Soit $\boldsymbol{H}'(x)$ la matrice $(n_0 - k_0) \times n_0$ polynomiale associée.
- Prenons une matrice $(n_0 - k_0) \times (n_0 - k_0)$ polynomiale inversible $\boldsymbol{S}(x)$ d'éléments de $\mathbb{F}_{q'}$
- Calculons $\boldsymbol{H}(x) = \boldsymbol{S}(x) \times \boldsymbol{H}'(x)$.
- En sortie, on obtient $pk = \boldsymbol{H}(x)$ et $sk = (\boldsymbol{a}, \theta, m, J, \boldsymbol{S}(x))$

FIGURE 7.2 – Génération de clés $\leftarrow 1^\kappa$

Chiffrement $\leftarrow (pk, \boldsymbol{x} \in \mathbb{F}_{q',t}^n)$

1. Soit H une matrice $(n-k) \times n$ quasi-circulante associée à $pk = \boldsymbol{H}(x)$.
2. *En sortie :* $\boldsymbol{c} = \boldsymbol{x} \times H^T$.

Déchiffrement $\leftarrow (pk, sk, \boldsymbol{c} \in \mathbb{F}_{q'}^n)$

1. Soit dec l'algorithme de décodage de $\mathscr{R}_k(\boldsymbol{\alpha}^{n_0} \otimes \boldsymbol{\beta}^\ell)$.
2. Posons $(\theta, \boldsymbol{a}, m, J, \boldsymbol{S}) = sk$.
3. Calculons $\boldsymbol{z} = \Phi^{-1}_{\theta, \boldsymbol{a} \otimes f_m(\boldsymbol{\beta}^\ell)}(\boldsymbol{c})$.
4. Soit S la matrice $(n-k) \times (n-k)$ quasi-circulante associée à $\boldsymbol{S}(x)$.
5. *En sortie :* $\boldsymbol{x} = \mathsf{dec}(\boldsymbol{z}) \times S^{-1}$

FIGURE 7.3 – Chiffrement $\leftarrow (pk, \boldsymbol{x} \in \mathbb{F}_{q',t}^n)$ et déchiffrement $\leftarrow (pk, sk, \boldsymbol{c} \in \mathbb{F}_{q'}^n)$

7.3.2 Efficacité et paramètres suggérés

Nous illustrons notre construction avec différents ensembles de paramètres. On remarque que comme dans le cryptosystème de McEliece originel, les paramètres sont facilement extensibles. Les deux cas que nous considérerons concernent des codes de Reed-Solomon Généralisés sur $\mathbb{F}_{2^{20}}$ et $\mathbb{F}_{2^{16}}$. Dans les cas suivants, les codes sont générés par quasi-cyclicité à partir de leur première ligne.

Codes de Reed-Solomon Généralisés sur $\mathbb{F}_{2^{16}}$

Paramètres A_{16} (Code sur $\mathbb{F}_{2^8}$) :
- $n_0 = 20$, $\ell = 51$, $t = 12$ et $s' = 8$
- $N_0 = 1285$
- $n = 1020$ et $k = n - 2 \times 2t$
- Taille de la clé publique : 7600 bits

Paramètres B_{16} (Code sur $\mathbb{F}_{2^4}$) :
- $n_0 = 13$, $\ell = 85$, $t = 21$ et $s' = 4$
- $N_0 = 771$
- $n = 1105$ et $k = n - 4 \times 2t$
- Taille de la clé publique : 8300 bits

Paramètres C_{16} (Code sur $\mathbb{F}_2$) :
- $n_0 = 24$, $\ell = 255$, $t = 15$ et $s' = 1$
- $N_0 = 257$
- $n = 6120$ et $k = n - 16 \times 2t$
- Taille de la clé publique : 12000 bits

Codes de Reed-Solomon Généralisés sur $\mathbb{F}_{2^{20}}$

Paramètres A_{20} (Code sur $\mathbb{F}_{2^{10}}$) :
- $n_0 = 8$, $\ell = 93$, $t = 23$ et $s' = 10$
- $N_0 = 11275$
- $n = 744$ et $k = n - 2 \times 2t$
- Taille de la clé publique : 6510 bits

Paramètres B_{20}(Code sur $\mathbb{F}_{2^{10}}$) :
- $n_0 = 10$, $\ell = 75$, $t = 12$ et $s' = 10$
- $N_0 = 13981$
- $n = 750$ et $k = n - 2 \times 2t$
- Taille de la clé publique : 6750 bits

Paramètres C_{20}(Code sur $\mathbb{F}_{2^5}$) :
- $n_0 = 10$, $\ell = 165$, $t = 23$ et $s' = 5$
- $N_0 = 6355$
- $n = 1320$ et $k = n - 4 \times 2t$
- Taille de la clé publique : 5775 bits

FIGURE 7.4 – Paramètres de codes alternants quasi-cycliques

Remarque

Le nombre de codes possibles avec notre construction est très grand : $\binom{N_0}{n_0}\mathbb{F}_{2^{pn_0}}n_0!$ *pour* $\mathbb{F}_{2^p}$ *le corps sur lequel est défini le code de départ. Par exemple pour les paramètres* A_{16} *on obtient* 2^{606} *codes possibles.*

Comme nous le verrons dans la section suivante, le nombre d'opérations dans le corps nécessaires à une attaque par décodage est d'au moins 2^{80} opérations binaires pour tous les exemples donnés. Les premières implémentations que nous avons faites montrent que le système que nous proposons est au moins aussi rapide que le cryptosystème NTRU voire même plus rapide en chiffrement et déchiffrement.

La structure quasi-cyclique de ces codes permet de passer facilement du cryptosystème de McEliece (où l'on donne la matrice génératrice du code) au cryptosystème de Niederreiter (dans lequel on donne une matrice duale du code).

7.4 Analyse de sécurité

Dans cette section nous considérerons les attaques classiques contre le cryptosystème de McEliece dans le but d'établir la sécurité de notre nouvelle variante. Tout cryptosystème à clé publique nécessite tout d'abord d'être résistant face à un attaquant qui désirerait faire un bris total du système : consistant à retrouver les données privées à partir uniquement des données publiques, ou à être capable d'inverser la fonction à sens unique à trappe permettant le chiffrement à partir des messages chiffrés de son choix et des données publiques. Cette seconde notion de sécurité est aussi appelée OW-CPA pour *One-Wayness under Chosen Plaintext Attack*.

Mais avant de considérer en détail les arguments établissant les notions de sécurité, nous avons besoin d'introduire quelques problèmes algorithmiques.

7.4.1 Complexité

Nous présentons tout d'abord deux importants problèmes NP-complets que nous rencontrons en théorie des codes correcteurs d'erreurs. Nous rappelons ici les problèmes présentés section 1.2.2 page 16. Le premier, appelé problème de *décodage par syndrome*, affirme que le décodage d'un code linéaire arbitraire est difficile (dans le pire cas). Il a été montré NP-complet dans [15]. L'autre, appelé *équivalence de codes poinçonnés*, est un problème qui a d'abord été proposé dans [138].

Définition 7.4.1 (*Décodage par Syndrome*)

Soit une matrice $r \times n$ notée H sur $\mathbb{F}_q$, un entier positif $w \leq n$ et un r-uplet $\boldsymbol{z}$ de $\mathbb{F}_q^r$, existe-t-il $\boldsymbol{e}$ dans $\mathbb{F}_q^n$ tel que $\mathsf{wt}(\boldsymbol{e}) \leq w$ et $H \times \boldsymbol{e}^T = \boldsymbol{z}$?

Définition 7.4.2 (*Équivalence de Codes Poinçonnés*)

Soit une matrice $r \times n$ notée H sur $\mathbb{F}_q$, matrice de parité du code $\mathcal{C}$ et une matrice $r \times n'$ notée H' sur $\mathbb{F}_q$, matrice de parité du code $\mathcal{C}'$, avec $n' < n$, existe-t-il σ dans $\mathcal{S}_n$ et $J \subset J_n$ de cardinal n' tel que $\mathcal{C}'$ est égal à la permutation σ du code $\mathcal{C}$ poinconné en les n' positions de J ?

Nous proposons dans ce chapitre un autre problème décisionnel appelé *Décodage par Syndrome Quasi-Cyclique*. Nous prouvons que ce nouveau problème est aussi NP- complet.

Problème 3 (*Décodage par Syndrome Quasi-Cyclique*)

Étant donné $\ell > 1$ (le cas $l = 1$ correspond au cas générique) des matrices $A_1, \ldots, A_\ell$ de taille $r^* \times n^*$ sur $\mathbb{F}_q$, un entier $w < \ell n^*$ et un mot $\boldsymbol{z}$ de $\mathbb{F}_q^{\ell r^*}$. Soit A une matrice $\ell r^* \times \ell n^*$ définie de la manière suivante :

$$A = \begin{bmatrix} A_1 & \cdots & \cdots & A_\ell \\ A_\ell & A_1 & \cdots & A_{\ell-1} \\ \vdots & \ddots & \ddots & \vdots \\ A_2 & \cdots & A_\ell & A_1 \end{bmatrix}$$

Existe-t-il $\boldsymbol{e}$ dans $\mathbb{F}_q^{\ell n^*}$ de poids $\mathsf{wt}(\boldsymbol{e}) \leq w$ tel que $A \times \boldsymbol{e}^T = \boldsymbol{z}$?

Proposition 7.4.1

Le problème de décodage par syndrome quasi-cyclique est NP-complet.

Preuve Considérons des instances H, w et $\boldsymbol{z}$ du problème de décodage par syndrome. Définissons $w^* = 2w$, le $2r$-uplet $\boldsymbol{z}^* = (\boldsymbol{z}, \boldsymbol{z})$ et la matrice $2r \times 2n$ notée A suivante :

$$A = \begin{bmatrix} H & 0 \\ 0 & H \end{bmatrix}.$$

On a A, $\boldsymbol{z}^*$ et w^* constructible en temps polynomial. Supposons maintenant qu'il existe $\boldsymbol{e}$ dans $\mathbb{F}_q^n$ de poids $\mathsf{wt}(\boldsymbol{e}) \leq w$ tel que $H \times \boldsymbol{e}^T = \boldsymbol{z}$. Alors $\mathsf{wt}(\boldsymbol{e}^*) \leq w^*$ et $A \times \boldsymbol{e}^{*T} = \boldsymbol{z}^*$.

Réciproquement, supposons qu'il existe $\boldsymbol{e}^*$ dans $\mathbb{F}_q^{2n}$ de poids $\mathsf{wt}(\boldsymbol{e}^*) \leq w^*$ tel que $A \times \boldsymbol{e}^{*T} = \boldsymbol{z}^*$. Si nous notons $\boldsymbol{e}^* = (\boldsymbol{e}_1, \boldsymbol{e}_2)$ où $\boldsymbol{e}_1$ est formé par les n premiers symboles et $\boldsymbol{e}_2$ par les n derniers symboles. On a clairement que soit $\boldsymbol{e}_1$ soit $\boldsymbol{e}_2$ est de poids $\leq w^*/2$ et pour tous les deux $H \times \boldsymbol{e}_j{}^T = \boldsymbol{z}$. □

Problème 4 (*Équivalence de codes*)

> *Soient deux matrices $k \times n$ notées G et G' sur $\mathbb{F}_q$, existe-t-il une transformation monomiale $\Phi_{\sigma,\boldsymbol{\lambda}}$ et une matrice $k \times k$ inversible S telle que $G' = \Phi_{\sigma,\boldsymbol{\lambda}}(S \times G)$?*

Contrairement aux autres problèmes décisionnels présentés ici, le problème d'*équivalence de codes* n'est pas un problème NP-complet [106]. L'algorithme de *séparation des supports* [123] qui a été conçu pour résoudre ce problème dans le cas particulier où la transformation monomiale est exactement la permutation. La complexité de cet algorithme dépend exponentiellement de la dimension de $\mathscr{C} \cap \mathscr{C}^\perp$, cet algorithme retrouve efficacement la permutation en pratique. Même si le problème n'est toujours pas résolu quand on considère des transformations monomiales qui sont strictement différentes des permutations, on peut raisonnablement supposer que le problème d'*équivalence de codes* est un problème difficile en pratique, pour des alphabets de plus de 4 éléments.

7.4.2 Étude de la sécurité

Recherche de clé par la méthode de force brute

Supposons qu'un attaquant ait seulement à sa disposition une matrice génératrice publique G d'un code $\mathscr{C}$ de dimension $k \geq n - 2rt$ et de longueur $n = \ell n_0$ sur le corps fini $\mathbb{F}_{2^{s'}}$. Supposons qu'il existe k_0 tel que $k = \ell k_0$. Rappelons que le code $\mathscr{C}$ est un code alternant quasi-cyclique issu du raccourcissement d'un code de Reed-Solomon Généralisé de longueur $N = \ell N_0$ et de dimension $K = N - 2t$ sur $\mathbb{F}_{2^s}$ avec $s = rs'$. La clé privée sk est $(\boldsymbol{a}, \theta, m, J, \boldsymbol{S}(x))$ et la clé publique est $pk = \boldsymbol{H}(x)$. L'ensemble J qui est un sous-ensemble de J_{N_0} de cardinal n_0. Le vecteur $\boldsymbol{a}$ est un n_0-uplet d'éléments non nul de $\mathbb{F}_q$ et θ appartient à $\mathbb{T}_\ell \wr \mathcal{S}_{n_0}$. Remarquons que a_0 peut être choisi égal à 1. Alors l'espace des clés privées contient $n_0! q^{n_0-1} \ell^{n_0+1} \binom{N_0}{n_0}$ éléments. Ce qui implique que les clés privées sont hors d'atteinte par recherche exhaustive.

Attaques structurelles

Nous revoyons maintenant toutes les attaques existantes qui ont pour but de retrouver la clé privée de cryptosystèmes basés sur des codes correcteurs d'erreurs. Ces attaques sont :

- l'attaque de *séparation des supports* [123],
- l'attaque de Sidelnikov-Shestakov [126],
- une attaque récente spécialement conçue pour les codes quasi-cycliques [98].

Attaque exploitant l'équivalence de codes Une première attaque consite à exploiter le fait que le code public généré par G est un sous-code de $\mathscr{A}_k(\theta, \boldsymbol{a}, m, J)$ qui est monomialement équivalent à un code de Reed-Solomon (voir Proposition 7.2.1 page 128). Nous pouvons déterminer directement $\Phi_{\theta,\boldsymbol{a}\otimes f_m(\boldsymbol{\beta}^\ell)}$ en appliquant un algorithme tel que l'algorithme de *séparation des supports* qui extrait la transformation à partir de deux matrices génératrices de codes monomialement équivalents. Cependant, un attaquant doit tout d'abord deviner l'ensemble J avant d'utiliser cette stratégie. Le nombre de tels ensembles est trop grand pour être à la portée d'une recherche par force brute. En outre, comme nous l'avons vu, l'algorithme de *séparation des supports* n'est pas adapté pour trouver des transformations monomiales, ce qui rend cette méthode impraticable.

Attaque exploitant la structure des codes de Reed-Solomon Généralisés Sidelnikov et Shestakov ont prouvé dans [126] qu'il est possible de retrouver complètement la structure des codes de Reed-Solomon Généralisés avec une complexité en temps de $\mathcal{O}(n^3)$ où n est la longueur du code public. Cette attaque peut être utilisée contre n'importe quel type de codes de Reed-Solomon Généralisés mais utilise le fait que la matrice sous-jacente d'un code de Reed-Solomon est une matrice de Vandermonde. Dans notre cas, nous n'utilisons pas directement un code GRS mais un sous-code sur un sous-corps. En prenant un tel code nous rendons cette attaque impraticable. Il est intéressant de remarquer que si un algorithme ayant cette efficacité existait pour les codes alternants, on pourrait potentiellement l'utiliser pour des codes utilisés généralement dans le cryptosystème de McEliece : les codes de Goppa sont un cas particulier de codes alternants binaires.

Attaque exploitant la structure quasi-cyclique Récemment une nouvelle attaque structurelle a été décrite dans [98]. Cette attaque extrait la clé privée sur la variante proposée dans [49]. Le cryptosystème utilise des sous-codes quasi-cycliques de code BCH de longueur n comme clé privée. La structure est cachée en utilisant une permutation fortement contrainte dans le but de produire un code public quasi-cyclique. Ceci implique que la permutation est complètement décrite avec n_0^2 entrées binaires avec n_0 l'ordre de quasi-cyclicité au lieu de n^2 entrées. L'idée de l'attaque consiste à se servir du fait que le secret est un sous-code d'un code BCH complètement connu. Cela permet de générer un système d'équations linéaires en exploitant la matrice génératrice publique et une matrice de parité connue d'un code BCH. On en déduit alors un système d'équations linéaires surdéfini vérifié par la matrice de permutation. Nous montrons comment adapter cette attaque à notre variante dans la Proposition 7.2.1 page 128.

Nous commençons avec une matrice de parité $M_{N-K}(\boldsymbol{\alpha}^N, \boldsymbol{\alpha}^N)$ d'un code de Reed-Solomon $\mathscr{R} = \mathscr{R}_K(\boldsymbol{\alpha}^N)$. On ajoute $(N_0 - n_0)$ colonnes nulles à la matrice polynomiale publique $\boldsymbol{H}(x)$ dans le but de former une matrice $(n_0 - k_0) \times N_0$ de parité polynomiale $\tilde{\boldsymbol{H}}(x)$. Nous voulons retrouver l'ensemble J et le N_0-uplet $\boldsymbol{\lambda}$ d'éléments de $\mathbb{F}_q^{N_0}$ (pas nécessairement non nul) tel que le code $\tilde{\mathscr{C}}$ définie par $\tilde{\boldsymbol{H}}(x)$ est équivalent par permutation à un sous-code d'un code de Reed-Solomon Généralisé $\mathscr{G}_k(\boldsymbol{\lambda}, \boldsymbol{\alpha}^{N_0} \otimes \boldsymbol{\beta}^\ell)$. En d'autres termes, en prenant une matrice génératrice quasi-circulante $\tilde{G}$ de $\tilde{\mathscr{C}}$, nous voulons trouver une permutation σ' de $\mathcal{S}_{N_0}$ telle que l'équation suivante est satisfaite :

$$M_{N-K}(\boldsymbol{\alpha}^N, \boldsymbol{\alpha}^N) \times \Big(\Big(P_{\sigma'} \times \mathsf{diag}(\boldsymbol{\lambda}) \Big) \otimes I_\ell \Big) \times \tilde{G} = 0. \tag{7.2}$$

Si nous posons $\Gamma = P_{\sigma'} \times \mathsf{diag}(\boldsymbol{\lambda})$ alors un attaquant doit résoudre le système d'équations linéaires donné dans l'équation 7.2 où les inconnues sont les N_0^2 entrées de Γ. Chaque colonne de $\tilde{G}$ donne $N - K = \ell(N_0 - K_0)$ équations de parité sur $\mathbb{F}_q$. Mais si l'on décompose chaque colonne

de $M_{N-K}(\boldsymbol{\alpha}^N, \boldsymbol{\alpha}^N)$ en une base Ω de $\mathbb{F}_q$ sur $\mathbb{F}_{q'}$, nous obtenons alors $r(N-K) = r\ell(N_0 - K_0)$ équations de parité. Le nombre total d'équations est alors :

$$r(N-K)k = r\ell^2(N_0 - K_0)k_0.$$

Si nous considérons les paramètres A_{16} nous obtenons $594\ 441$ inconnues et $61\ 600$ équations. Un attaquant a alors à réduire le nombre d'inconnues dans le but de deviner Γ. Cependant, avec $r\ell^2(N_0 - K_0)k_0$ équations linéaires, un attaquant peut potentiellement deviner la matrice $\delta \times \delta$ secrète avec $\delta = \ell\sqrt{r(N_0 - K_0)k_0}$.

Une autre stratégie est alors de mettre $N_0 - \delta$ colonnes aléatoires de Γ à zéro et de résoudre le système linéaire résultant. Cette méthode donne la bonne solution si et seulement si les n_0 colonnes qui interviennent dans la construction du code alternant raccourci ne sont pas mises à zéro. La probabilité de succès de cette méthode est alors de :

$$\frac{\binom{N_0 - n_0}{\delta - n_0}}{\binom{N_0}{\delta}}.$$

De plus, pour chaque choix aléatoire, nous devons résoudre un système linéaire avec une complexité en temps de $\mathcal{O}(s'^2\delta^3)$ pour des coefficients dans $\mathbb{F}_{2^{s'}}$. Avec les paramètres A_{16} nous obtenons une probabilité de succès de 2^{-178} et environ 2^{208} opérations.

7.4.3 Sur le côté sens-unique

La propriété de *sens-unique* (*one-wayness*) est la notion de sécurité la plus faible que n'importe quel cryptosytème à clé publique doit satisfaire. Cela suppose essentiellement que l'inversion de la fonction de chiffrement est calculatoirement impossible sans connaître le secret appelé *trappe*. Dans le cas des cryptosystèmes de McEliece ou de Niederreiter, cela consiste à décoder un mot donné en un mot de code du code public. Cet objectif peut se voir comme le fait de résoudre le problème qui consiste à trouver un mot de code de petit poids dans un code approprié. Nous supposons que nous chiffrons avec le cryptosystème de McEliece. Nous avons un générateur public G et un texte chiffré $\boldsymbol{z}$. Nous savons qu'il existe un mot de code $\boldsymbol{c}$ de $\mathscr{C}$ et un vecteur $\boldsymbol{e} \in \mathbb{F}_{q'}^n$ de poids t tels que $\boldsymbol{z} = \boldsymbol{c} + \boldsymbol{e}$. Par hypothèse, la distance minimale de $\mathscr{C}$ est $2t+1$. Par conséquent le code linéaire $\tilde{\mathscr{C}}$ défini par la matrice génératrice $\tilde{G} = \begin{bmatrix} G \\ \boldsymbol{c} \end{bmatrix}$ contient un mot de code de poids t que l'on appelera $\boldsymbol{e}$. Ainsi, inverser le chiffrement revient à trouver des mots de code de petits poids dans le code donné. Nous savons que ce problème est NP-difficile pour un code linéaire arbitraire. Nous avons également prouvé que c'est également le cas pour des codes quasi-cycliques arbitraires. En outre, tous les algorithmes existants, ont une complexité qui s'accroît exponentiellement avec la longueur et le taux du code considéré. Le plus efficace est l'algorithme de Canteaut-Chabaud [22] (section 2.3 page 26) qui est une amélioration de l'algorithme de Stern [128].

Bien que l'on puisse penser qu'il existe des algorithmes plus efficaces pour décoder des codes quasi-cycliques aléatoires, il s'avère que de tels algorithmes n'existent pas dans la littérature pour autant que nous le sachions. Cette situation doit être comparée avec le cas du cryptosystème NTRU pour lequel la meilleure attaque consiste en la recherche de vecteurs de petites normes, dans un réseau quasi-cyclique. Dans ce cas, l'algorithme LLL est utilisé, mais après 10 années de recherche, il n'existe pas de meilleur algorithme que l'application de l'algorithme LLL pour trouver ces vecteurs de petites normes. Il n'existe fondamentalement pas d'algorithme plus efficace dédié au cas

des réseaux quasi-cycliques (sauf une heuristique publiée récemment qui permet de gagner une petite constante d'ordre 7 ou 8 sur l'ensemble du coût du calcul). Le même type de situation pour la théorie des codes correcteurs d'erreurs se produit ici et nous appliquons directement le meilleur algorithme pour le cas des codes aléatoires.

Afin de dériver une borne sur la complexité en temps, nous considérons le second algorithme. Le coût de travail $\Omega_{n,k,w}$ de l'algorithme de Stern pour trouver A_w mots de code de poids w dans un code de longueur n, de dimension k sur $\mathbb{F}_{q'}$ est :

$$\Omega_{n,k,w} = \frac{\binom{n-k}{w}}{\binom{n}{w}}.$$

Le coût de la mise à jour de la matrice est d'au moins $n \times k$, ce qui donne éventuellement le nombre d'opérations dans le corps (multiplications) nécessaires. Pour les paramètres concrets donnés dans la section 4, tous les paramètres de sécurité sont d'au moins 2^{80} opérations dans le corps.

7.5 Conclusion

Dans ce chapitre, on a présenté une nouvelle façon de réduire la taille de la clé publique pour les cryptosystèmes basés sur les codes à savoir le cryptosystème de McEliece ou celui de Niederreiter. L'utilisation de codes quasi-cycliques permet d'obtenir une clé publique de $6\,000$ bits. Les premiers tests de vitesse que nous avons faits montrent que le système possède une vitesse similaire au système NTRU. De tels paramètres et une telle rapidité ouvrent les portes à de nouvelles applications pour la cryptographie basée sur les codes correcteurs d'erreurs, sur carte à puce par exemple, où un tel algorithme peut être utilisé pour l'échange de clés ou l'authentification.

7.6 Programme

```
//////////////////Initialisation des variables
 print "\nParamètres";
 n:= 2^16;
 r:= 85;
 s:= 771;
 k:= n-80;
 K:= GF(n);
 K2:=sub<K | 8> ;
print "Le code de départ";
print "n:=",n, "k:=",k,"\tr:=",r,"\ts:=",s;
print "Le corps de base",K;
print "Le sous-corps",K2;

/////////////////////////// Construction de A
// Je construit ici une famille de r ensembles C_i
// avec C[i]={a^i,a^i*a^r,...,a^i*a^(s-1)}
// Je construit A qui est l'ensemble A:={C[tab[i]]}
// on concatène les C_i.
```

```
// ex : A[1] = le premier élément de C[tab[i]] soit C[tab[i][1]].

function constructA(n,r,s,tab)
 a:=(K.1);
 print "a:=",a;
 B:=[];
 C:=[];
 A:=[];

 for j in [0..r-1] do
 B:=[];
  for i in [0..s-1] do
   B:= B cat [a^(j+i*r)];

  end for;
 C:=Append(C,B);
 end for;

for i in [1..r] do
A:=A cat (C[tab[i]]);
end for;

return A;
end function;

// Construction de A en permutant les indices
// (j'ai juste fait un décalage)

function Permutation(n,r,s)
tab:=[];

tab[1]:=Random(1,r);
//tab[1]=1;

for i in [2..r] do
tab[i]:=((tab[1]+(i-1)) mod (r));
end for;

for i in [1..r] do
if tab[i] eq 0 then tab[i]:=r; end if;
end for;

return tab;
end function;

//////Maintenant au lieu d'utiliser A[j] on utilisera A[tab[j]]
```

```
//////////////////////////// Construction de V

function constructV(n,r,s)

V:=[];

for i in [1..r] do
 V[i]:=K!1;
end for;

for j in [1..s-1] do
 lambda:=Random(2,n);
 for i in [1..r] do
  V[i+j*r]:=K.1^lambda;
 end for;
end for;
return V;
end function;

//Programme principal

tab:=Permutation(n,r,s);

A:=constructA(n,r,s,tab);

V:=constructV(n,r,s);

D := GRSCode(A, V, k);

g:=MinimumDistance(D);
print "MinimumDistance(D) :=",g;
print "Capacité de Correction (D) :=",g/2;

print "On prend le sous-code sur le sous-corps E";

E,F:=SubfieldSubcode(D,K2);

print "Length(E) sous-code sur le sous-corps n :=",Length(E);
print "Dimension(E) sous-code sur le sous-corps k :=",Dimension(E);
h:=MinimumDistance(E);
print "MinimumDistance(E) d :=",h;
```

Chapitre 8

Conclusion et perspectives

Cette thèse m'a donné l'occasion d'explorer quelques nouvelles applications de la théorie des codes correcteurs d'erreurs en cryptographie à clé publique. Ces applications sont parfois concrètes voire utilisables en pratique. Avec les travaux fournis dans cette thèse, j'apporte la preuve (si cela était nécessaire) que la cryptographie basée sur les codes correcteurs d'erreurs possède d'excellentes propriétés en terme d'efficacité dans un contexte à faibles ressources ainsi que dans des contextes plus variés comme la signature de cercle ou bien la cryptographie basée sur l'identité.

Je vais faire le point ici sur les différents chapitres de cette thèse, en apportant des perspectives.

L'implémentation sûre du schéma de Stern

Nous décrivons la première implémentation du protocole de Stern sur carte à puce (en fait, cela constitue aussi, plus généralement, le premier cryptosystème basé sur des codes correcteurs d'erreurs implémenté sur carte à puce avec peu de ressources). Ce protocole est une nouvelle option pour réaliser une identification forte et rapide sur carte à puce. De plus, l'utilisation d'un co-processeur d'algèbre linéaire dédié devrait améliorer sensiblement les performances de notre implémentation. De même qu'une version hardware de SHA-256 au lieu d'une implémentation software améliorerait sérieusement les performances (en temps) de notre protocole. Des travaux futurs envisagent des attaques d'injection de fautes et la mise en oeuvre d'autres variantes du protocole de Stern qui peuvent avoir d'autres avantages pour des variations du protocole.

Le schéma de signature de Kabatianskii Krouk et Smeets

Je pense que notre étude montre les limites de ce schéma de signature. L'utilisation de matrices quasi-circulantes permet de réduire la taille des clés publiques tandis qu'un choix judicieux des paramètres permet d'augmenter le nombre de signatures nécessaires à la découverte de la clé secrète. Je pense néanmoins qu'une preuve de sécurité pourrait apporter un certain crédit à ce schéma de signature dans un contexte où l'on ne nécessite que peu de signatures.

Le schéma de signature basé sur l'identité prouvé sûr

C'est certainement la partie de cette thèse qui est la plus aboutie, du point de vue de la sécurité théorique. Le schéma combine deux schémas bien connus et hérite malheureusement des mauvaises

propriétés de ces deux systèmes, à savoir une grande taille des données publiques et un coût de communication, pour le schéma d'authentification, élevé ainsi qu'une taille de signature également très grande. Mais malgré ces faiblesses, ce système présente la première alternative à la théorie des nombres pour la cryptographie basée sur l'identité et permet d'ouvrir de nouveaux domaines de recherche. De plus, la preuve de sécurité en fait le premier schéma d'authentification et de signature basée sur l'identité n'utilisant pas de problèmes de théorie des nombres prouvé sûr. Ce chapitre montre l'étendue des possibilités des codes correcteurs en cryptographie à clé publique.

La question de l'existence d'un schéma de chiffrement basé sur l'identité utilisant un problème de théorie des codes correcteurs d'erreurs reste ouverte.

Le schéma de signature de cercle à seuil

Nous avons présenté un nouveau schéma de signature de cercle à seuil complètement anonyme basé sur la théorie des codes correcteurs d'erreurs. Notre protocole est une généralisation naturelle du schéma d'authentification de Stern et notre preuve est basée sur la preuve originelle de Stern. Nous avons montré que le choix du poids d'un vecteur particulier rend applicable ce schéma dans le cas des signatures de cercle et que la notion de groupe ad hoc correspond bien à la notion de somme directe de matrices génératrices.

Nous obtenons un protocole complètement anonyme basé sur une preuve de connaissance. Alors que les schémas pré-existants ne pouvaient traiter que des cas où $t \approx \mathcal{O}(\log(N))$, notre schéma permet de traiter n'importe quel t et en particulier $t \approx N/2$.

Le fait que notre construction ne soit pas basée sur des problèmes de théorie des nombres mais sur des problèmes de codes représente, une fois encore, une alternative intéressante.

Nous travaillons actuellement (en collaboration avec Fabien Laguillaumie de Caen) sur la preuve de sécurité de ce schéma dans le modèle de l'oracle aléatoire.

McEliece avec des codes alternants quasi-cycliques

La sécurité du schéma de McEliece que nous proposons repose directement sur les mêmes problèmes que dans le système de McEliece originel. Or, ce système ayant été étudié en détails depuis près de 30 ans, peut être considéré comme suffisamment robuste. Il suffit donc de vérifier que les quelques modifications que nous avons apportées, dues au fait qu'on utilise des codes alternants quasi-cycliques, soient des sous-codes sur un sous-corps de code de Reed-Solomon Généralisé, n'affaiblissent pas le système originel. L'utilisation de codes quasi-cycliques permet d'obtenir une clé publique de $6\ 000$ bits. Les premiers comparatifs de vitesse que nous avons fait montrent que le système possède une vitesse similaire au système NTRU. De tels paramètres et une telle rapidité ouvrent les portes de nouvelles applications pour la cryptographie basée sur les codes correcteurs d'erreurs, sur carte à puce par exemple, où un tel algorithme peut être utilisé pour l'échange de clés ou l'authentification.

Nous avons construit, à l'aide de magma, les codes alternants quasi-cycliques, il reste à implémenter leur algorithme de décodage ainsi que le protocole en lui même.

Autres perspectives

Bien sûr, rien ne sert d'essayer de refaire toute la cryptographie moderne avec des codes correcteurs d'erreurs. Cependant, en faisant cela on a de bonnes chances de faire des découvertes

intéressantes. Cela a par exemple été le cas pour la signature de cercle à seuil : au delà du fait d'être complètement anonyme, ce protocole possède la plus petite taille de clé publique, de signature et la plus faible complexité pour un schéma de signature de cercle à seuil. Il reste donc un énorme travail d'investigation à effectuer pour essayer d'utiliser de nouveaux problèmes difficiles de théorie des codes, ou pour tenter d'adapter les problèmes bien connus à de nouvelles primitives : signatures aveugles, signatures indéniables, schéma de sign-cryption, générateurs d'aléas, . . .

Même si des choses semblent très difficiles, comme par exemple un schéma de chiffrement basé sur l'identité utilisant des codes correcteurs d'erreurs, les codes ont le très grand avantage de présenter peu de contraintes, avec des paramètres facilement variables, tout en possédant de nombreux problèmes difficiles, et même souvent difficiles pour une instance aléatoire moyenne.

Que choisir ?

Si vous deviez choisir un type de cryptographie à conseiller aux générations futures entre la cryptographie classique, basée sur les problèmes de factorisation ou de logarithme discret, et la cryptographie basée sur les codes correcteurs d'erreurs, quel serait votre choix ?

La réponse à cette question n'est pas triviale. La cryptographie classique, dirons nous, possède de nombreux avantages : rapide, énormément étudiée, très variée (multitudes de protocoles différents bien que basés sur les mêmes problématiques), munie de preuve de sécurité. Mais aussi des inconvénients : des calculs lourds (exponentiation modulaire, calcul sur des courbes elliptiques, . . .), fragile face à l'ordinateur quantique (algorithme de Shor), des paramètres difficilement modifiables . . .

La cryptographie basée sur les codes correcteurs quant à elle ne souffre pas des mêmes faiblesses : opérations très rapides car basées sur de l'algèbre linéaire, ne nécessitent pas de crypto-processeur (intéressant dans des milieux à très faible ressource), des paramètres très facilement modifiables, une résistance face à l'ordinateur quantique. Mais elle souffre d'autres faiblesses : très grande taille de clés publiques, peu d'études (par rapport à la cryptographie classique), pas beaucoup de propriétés particulières (signature de groupe, à propriétés spéciales).

Dans cette thèse, j'ai montré que l'on pouvait combler certaines de ces faiblesses en proposant des schémas avec des propriétés particulières, des tailles de clés publiques raisonnables, ainsi que des implémentations effectives.

En conclusion, je me refuse à trancher et je dirai simplement que la cryptographie basée sur les codes correcteurs d'erreurs constitue une alternative aux systèmes de chiffrement à clé publique *classiques* à ne pas négliger.

La cryptographie basée sur les codes correcteurs d'erreurs a encore de beaux jours devant elle !

Publications personnelles et soumissions

[2] C. Aguilar Melchor, P.-L. Cayrel and P. Gaborit : A New Efficient Threshold Ring Signature Scheme Based on Coding Theory. Lecture Notes in Computer Science, Vol. 5299, pages 1-16, *PQ CRYPTO* 2008.

[7] F. Armknecht, P.-L. Cayrel, P. Gaborit and O. Ruatta : Algorithme amélioré de recherche d'équations dans le cas des attaques algébriques des registres avec mémoire. *MAJECSTIC, Prépublications de l'Université de Caen Basse-Normandie*, Fascicule n°2, :239–243, 2007.

[8] F. Armknecht, P.-L. Cayrel, P. Gaborit and O. Ruatta : Improved algorithm to find equations for algebraic attacks for combiners with memory. *Third International Workshop on Boolean Functions : Cryptography and Applications, Proceedings of BFCA 2007,* Jean-Francis Michon, Pierre Valarcher, Jean-Baptiste Yunès Eds., :81–98, 2008.

[10] M. Barbier and P.-L. Cayrel : Attaques algébriques. *MAJECSTIC 2008*.

[14] T. Berger, P.-L. Cayrel, P. Gaborit and A. Otmani : Reducing Key Lengths with Quasi-cyclic Alternant Codes for Code-Based Cryptography 2008.

[24] P.-L. Cayrel, C. Chabot and A. Necer : Suites récurrentes linéaires et codes quasi-cycliques. 2008.

[25] P.-L. Cayrel and L. Dallot : Schémas de signature prouvés sûrs fondés sur la théorie des codes correcteurs d'erreurs. *MAJECSTIC 2008.*

[26] P.-L. Cayrel and N. Elmrabet : Attaques matérielles. *MAJECSTIC 2008.*

[27] P.-L. Cayrel, P. Gaborit, D. Galindo and M. Girault : On code-based identity-based identification and signature schemes. 2008.

[28] P.-L. Cayrel, P. Gaborit and M. Girault : Identity-based identification and signature schemes using correcting codes. In D. Augot, N. Sendrier and J.-P. Tillich, editors, *WCC 2007*. INRIA, 2007.

[29] P.-L. Cayrel, P. Gaborit and M. Girault : Identity-based Identification and Signature Schemes using Error Correcting Codes, *chapter Identity-based Identification and Signature Schemes using Error Correcting Codes*. Marc Joye and Gregory Neven, 2008.

[30] P.-L. Cayrel, A. Otmani and D. Vergnaud : On Kabatianskii-Krouk-Smeets signatures. *WAIFI 2007*, Springer Carlet C. and Sunar B. Lecture Notes in Computer Science :237–251, 2007.

[31] P.-L. Cayrel, P. Gaborit and E. Prouff : Secure Implementation of the Stern Authentication and Signature Schemes for Low-Resource Devices Eighth Smart Card Research and Advanced Application Conference *CARDIS 2008* In G. Grimaud and F.-X. Standaert, editors, Lecture Notes in Computer Science, Vol. 5189, pages 191-205, 2008.

Bibliographie

[1] C. Adams and H. Meijer : Security-related comments regarding McEliece public-key cryptosystem. *IEEE Trans. Inform. Theory*, 35 Nr. 2 S. :454–455, 1989.

[2] C. Aguilar Melchor, P.-L. Cayrel and P. Gaborit : A new efficient threshold ring signature scheme based on coding theory. 2008.

[3] W. Aiello, S. Haber and R. Venkatesan : New constructions for secure hash functions. *Fast Software Encryption Lecture Notes in Computer Science 1372, S. Vaudenay, Ed., Springer-Verlag, 1998, pp 150–167.*

[4] W. Aiello and R. Venkatesan : Foiling birthday attacks in length-doubling transformations. BENES : a non-inversible alternative to Feistel. *Advances in Cryptology Proceedings of Eurocrypt'96, Lecture Notes in Computer Science 1070, U. Maurer, Ed., Springer-Verlag, 1996, pp. 307–320.*

[5] M.-L. Akkar and C. Giraud : An Implementation of DES and AES, Secure against Some Attacks. In Ç. Koç, D. Naccache and C. Paar, editors. *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162, pages 309–318, 2001.

[6] F. Armknecht : Algebraic attacks and annihilators. *Proceedings of WEWORC 2005*, LNI P-74 :13–21, 2005.

[7] F. Armknecht, P.-L. Cayrel, P. Gaborit and O. Ruatta : Algorithme amélioré de recherche d'équations dans le cas des attaques algébriques des registres avec mémoire. *MAJECSTIC, Prépublications de l'Université de Caen Basse-Normandie*, Fascicule n°2, :239–243, 2007.

[8] F. Armknecht, P.-L. Cayrel, P. Gaborit and O. Ruatta : Improved algorithm to find equations for algebraic attacks for combiners with memory. *Third International Workshop on Boolean Functions : Cryptography and Applications, Proceedings of BFCA 2007,* Jean-Francis Michon, Pierre Valarcher, Jean-Baptiste Yunès Eds., :81–98, 2008.

[9] F. Armknecht and M. Krause : Algebraic attacks on combiners with memory. *Advances in Cryptology–CRYPTO 2003,* Lecture Notes in Computer Science 2729 Springer :162–176, 2003.

[10] M. Barbier and P.-L. Cayrel : Attaques algébriques. *MAJECSTIC 2008*.

[11] M. Barot and J. A. de la Pena : Estimating the size of a union of random subsets of fixed cardinality. *Elemente der Mathematik*, 56 :163–169, 2001.

[12] M. Bellare, C.Namprempre and G. Neven : Security proofs for identity-based authentication and signature schemes. *Eurocrypt 2004*, Lecture Notes in Computer Science 3027 :268–286, 2004.

[13] A. Bender, J. Katz and R. Morselli : Ring signatures stronger definitions and constructions without random oracles theory of cryptography. *Lecture Notes in Computer Science 3876* : page 60–79, 2006.

[14] T. Berger, P.-L. Cayrel, P. Gaborit and A. Otmani : Reducing Key Lengths with Quasi-cyclic Alternant Codes for Code-Based Cryptography 2008.

[15] E. Berlekamp, R. McEliece and H. van Tilborg : On the inherent intractability of certain coding problems. *Information Theory, IEEE Transactions on*, 24(3) :384–386, May 1978.

[16] T. Berson : Failure of the McEliece public-key cryptosystem under message-resend and related-message attack. *Crypto 97*, 1997.

[17] T. Beth. and M. Frisch : Public-key Cryptography State of the Art and Future Directions. *E.I.S.S. Workshop, Oberwolfach, Germany, July 3-6, 1991 Final Report (Lecture Notes in Computer Science)* Springer-Verlag.

[18] X. Boyen : Mesh Signatures. *EUROCRYPT 2007 : pp. 210-227*

[19] E. Bresson : Protocoles cryptographiques pour l'authentification et l'anonymat dans les groupes. *PhD thesis, École polytechnique, 2002.*

[20] E. Bresson, J. Stern and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. *Advances in Cryptology Crypto 2002*, 2002.

[21] E. Brier, C. Clavier and F. Olivier : Correlation Power Analysis with a Leakage Model. In M. Joye and J.-J. Quisquater, editors. *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156,pages 16–29, 2004.

[22] A. Canteaut and F. Chabaud : A new algorithm for finding minimum-weight words in a linear code : Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, 44(1) :367–378, 1998.

[23] A. Canteaut and N. Sendrier : Cryptoanalysis of the original McEliece cryptosystem. *in. Lecture Notes In Computer Science ; Vol. 1514* 187–199, 1998. Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security : Advances in Cryptology

[24] P.-L. Cayrel, C. Chabot and A. Necer : Suites récurrentes linéaires et codes quasi-cycliques. 2008.

[25] P.-L. Cayrel and L. Dallot : Schémas de signature prouvés sûrs fondés sur la théorie des codes correcteurs d'erreurs. *MAJECSTIC 2008.*

[26] P.-L. Cayrel and N. Elmrabet : Attaques matérielles. *MAJECSTIC 2008.*

[27] P.-L. Cayrel, P. Gaborit, D. Galindo and M. Girault : On code-based identity-based identification and signature schemes. 2008.

[28] P.-L. Cayrel, P. Gaborit and M. Girault : Identity-based identification and signature schemes using correcting codes. In D. Augot, N. Sendrier and J.-P. Tillich, editors, *WCC 2007*. INRIA, 2007.

[29] P.-L. Cayrel, P. Gaborit and M. Girault : Identity-based Identification and Signature Schemes using Error Correcting Codes, *chapter Identity-based Identification and Signature Schemes using Error Correcting Codes*. Marc Joye and Gregory Neven, 2008.

[30] P.-L. Cayrel, A. Otmani and D. Vergnaud : On Kabatianskii-Krouk-Smeets signatures. *WAIFI 2007*, Springer Carlet C. and Sunar B. Lecture Notes in Computer Science :237–251, 2007.

[31] P.-L. Cayrel, P. Gaborit and E. Prouff : Secure Implementation of the Stern Authentication and Signature Schemes for Low-Resource Devices Eighth Smart Card Research and Advanced Application Conference *CARDIS 2008* In G. Grimaud and F.-X. Standaert, editors, Lecture Notes in Computer Science, Vol. 5189, pages 191-205, 2008.

[32] J. Cha and J. Cheon : An identity-based signature from gap Diffie-Hellman groups. *PKC 2003 6th International Workshop on Theory and Practice in Public Key Cryptography, volume 2567 of Lecture Notes in Computer Science, pages 18–30, 2003.*

[33] F. Chabaud : On the security of some cryptosystems based on error-correcting codes. *Lecture Notes in Computer Science 950* : pp 131–139, 1995. citeseer.ist.psu.edu/chabaud94security.html

[34] N. Chandran, J.Groth and A.Sahai : Ring signatures of sub-linear size without random oracles. *ICALP*, Lecture Notes in Computer Science 4596 :423–434, 2007.

[35] S. Chari, C. Jutla, J. Rao and P. Rohatgi : A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards.

[36] S. Chari, C. Jutla, J. Rao and P. Rohatgi : Towards Sound Approaches to Counteract Power-Analysis Attacks. In M. Wiener, editor. *Advances in Cryptology – CRYPTO '99*, volume 1666,pages 398–412, 1999.

[37] D. Chaum and E. Heyst : Group signatures in advances. *Cryptology Eurocrypt'91*, 1991.

[38] N. Courtois : Fast algebraic attacks on stream ciphers with linear feedback. *Crypto '03*, Lecture Notes in Computer Science 2729 Springer :177–194, 2003.

[39] N. T. Courtois, M. Finiasz and N. Sendrier : How to achieve a McEliece-based digital signature scheme. *Lecture Notes in Computer Science*, 2248 :157–174, 2001.

[40] N. Courtois and J. Pieprzyk : Cryptanalysis of block ciphers with overdefined systems of equations. *Lecture Notes in Computer Science 2501 Springer :267–287*, 2002. citeseer.ist.psu.edu/courtois02cryptanalysis.html

[41] L. Dallot : Towards a concrete security proof of Courtois Finiasz and Sendrier signature scheme. *Weworc 2007*.

[42] Y. Dodis, A. Kiayias, A. Nicolosi and V. Shoup : Anonymous identification in ad-hoc groups. *Advances in Cryptology Eurocrypt 2004*, 2004.

[43] N. Elmrabet : Dis moi ce que tu consommes, je te dirai qui tu es. *Doctiss'08, journée des doctorants de l'école doctorale I2S, Montpellier, France,*, 10 avril 2008.

[44] D. Engelbert, R. Overbeck and A. Schmidt : A summary of McEliece-type cryptosystems and their security. *Cryptology ePrint Archive*, Report 2006/162, 2006.

[45] U. Feige, A. Fiat and A. Shamir : Zero-knowledge proofs of identity. *J. Cryptology 1(2) : 77-94, 1988.*

[46] A. Fiat and A. Shamir : How to prove yourself : practical solutions to identification and signature problems. In A. Odyzko, editor, *Advances in Cryptology – CRYPTO '86*, volume 263, pages 186–194, 1986.

[47] M. Finiasz : Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie à clé publique. *PhD thesis, INRIA – Ecole Polytechnique*, 2004.

[48] M. Franklin and D. Boneh : Identity-based encryption from the Weil pairing. *Advances in Cryptology-Crypto'01, 2001.*

[49] P. Gaborit : Shorter keys for code based cryptography. In *Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005)*, pages 81–91, Bergen, Norway, Mar. 2005.

[50] P. Gaborit and M. Girault : Lightweight code-based identification and signature. In *Proceeedings of ISIT'07*, 2007.

[51] J. Gibson : Equivalent codes and trapdoors to McEliece's public-key cryptosystem. In J. Feigenbaum, editor. *Advances in Cryptology – EUROCRYPT '91*, volume 547, 1991.

[52] S. Goldwasser, S. Micali and C. Rackoff : The knowledge complexity of interactive proof systems. *SIAM, Journal of Computing*, 18 :186–208, 1989.

[53] S. Goldwasser, S. Micali and R. Rivest : A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing, 17(2) :281–308, April 1988*.

[54] L. Goubin and J. Patarin : DES and Differential Power Analysis – The Duplication Method. In Ç. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES '99*, volume 1717, pages 158–172, 1999.

[55] S. Gueron, O. Parzanchevsky and O. Zuk : Masked Inversion in GF(2^n) Using Mixed Field Representations and its Efficient Implementation for AES. In N. Nedjah and L. M. Mourelle, editors, *Embedded Cryptographic Hardware : Methodologies and Architectures*, pages 213–228. Nova Science Publishers, 2004.

[56] R. Heiman : On the security of cryptosystems based on linear error-correcting codes. *Master's thesis, Feinberg Graduate School of the Weizman Institute of Science, 1987*

[57] H. Imai and K. Kobara : Countermeasure against reaction attacks (in japanese). *The 2000 Symposium on Cryptography and Information Security*, 2000.

[58] H. Imai and K. Kobara : Semantically secure mceliece public-key cryptosystems - conversions for McEliece PKC. *Proc. of 4th International Workshop on Practice and Theory in Public Key Cryptosystems (PKC 01)*, pages 19–35, 2001.

[59] H. Imai and K. Kobara : On the One-wayness against chosen-plaintext attacks of the Loidreau's modified Mceliece PKC. *IEEE Trans. Inform. Theory*, 49 Nr. 12 S. :3160–3168, 2003.

[60] G. Kabatianskii, E.Krouk and B. J. M. Smeets : A digital signature scheme based on random error-correcting codes. *IMA Int. Conf.*, Springer Lecture Notes in Computer Science 1355 :161–167, 1997.

[61] J. Kelsey and B. Schneier : "Key-schedule cryptanalysis of DEAL". *Selected Areas in Cryptography*, pages 118–134, 1999.

[62] G. Knoth : A numbering system for permutations and combinations communication of the ACM. *Communications of the ACM (CACM).* Vol. 19, No 6, Juin 1976.

[63] L. Knudsen : DEAL : a 128-bit blok cipher. *Technical Report 151, Department of Informatics, University of Bergen, Norway, Feb. 1998*.

[64] L. Knudsen. and B. Preneel : Attacks on Fast Double Block Length Hash Functions. *Journal of Cryptology* Springer Volume 11, Number 1, septembre 1998 Pages 59-72

[65] L. Knudsen and B. Preneel : Fast and secure hashing based on codes. *appeared in Advances in Cryptology 1997,Lecture Notes in Computer Science 1294, B. S. Kaliski Jr. (ed.), Springer-Verlag, pp. 485–498, 1997*.

[66] L. Knudsen and B. Preneel : Hash Functions Based on Block Ciphers and Quaternary Codes. *ASIACRYPT 1996 : Advances in Cryptology – International Conference on the Theory and Application of Cryptology*. LNCS, Springer-Verlag, 1996. citeseer.ist.psu.edu/knudsen96hash.html.

[67] D. Knuth : The art of computer programming, volume 2 : Seminumerical algorithms. *Addison-Wesley, 1981*.

[68] P. Kocher : Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109, pages 104–113, 1996.

[69] P. Kocher, J. Jaffe and B. Jun : Differential Power Analysis. In M. Wiener, editor. *Advances in Cryptology – CRYPTO '99*, volume 1666, pages 388–397, 1999.

[70] V. Korzhik and A. Turkin : Cryptanalysis of McEliece's public-key cryptosystem. In J. Feigenbaum, editor. *Advances in Cryptology – EUROCRYPT '91*, volume 547,pages 68–70, 1991.

[71] X. Lai : On the design and security of block ciphers. *ETH Series in Information Processing, Vol. 1, J.L. massey, Ed. Hartung-Gorre Verlag, Konstanz, 1992*.

[72] L. Lamport : Constructing digital signatures from a one way function. *Technical Report CSL-98, SRI International*, Oct. 1979.

[73] P. J. Lee and E. F. Brickell : An observation on the security of McEliece's public-key cryptosystem. *Lecture Notes in Computer Science 330/1988 Springer :275–280, 1988.*

[74] K. Lemke, K. Schramm and C. Paar : DPA on n-Bit Sized Boolean and Arithmetic Operations and its Applications to IDEA. In M. Joye and J.-J. Quisquater, editors. *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156, 2004.

[75] Y. Li, R. Deng and X. Wang : On the equivalence of McEliece's and Niederreiter's cryptosystems. *IEEE Trans. on Information Theory IT-40*, 40 no. 1 :271–273, 1994.

[76] J.K. Liu, V.K. Wei and D.S. Wong : A Separable Threshold Ring Signature Scheme. In Jong In Lim and Dong Hoon Lee, editors, Information Security and Cryptology - *ICISC 2003,* 6th International Conference Seoul, Korea, November 27-28, 2003, Revised Papers, volume 2971 of Lecture Notes in Computer Science, pages 352-369. Springer, 2003.

[77] P. Loidreau : Strengthening McEliece cryptosystem. *Advances in cryptology - ASIACRYPT 2000* International conference on the theory and application of cryptology and information security No6, vol.1976, pp. 585-598 2000.

[78] P. Loidreau and N. Sendrier : Weak keys in McEliece public key cryptosystem. *IEEE Trans. Inf. Theory*, 2001.

[79] Y. Lu, W. Meier and S. Vaudenay : The conditional correlation attack a practical attack on Bluetooth encryption. *Crypto 2005*, Lecture Notes in Computer Science 3621 Springer :97–117, 2005.

[80] Y. Lu and S. Vaudenay : Cryptanalysis of Bluetooth keystream generator two-level E_0. *Lecture Notes in Computer Science 3329 Springer :483–499*, 2004.

[81] M. Luby and C. Rackoff : Pseudo-random permutation generators and cryptographic composition. *Symposium on Theory of Computing*, 18 :353–363, 1986.

[82] M. Luby and C. Rackoff : How to construct pseudorandom permutation and pseudorandom functions. *SIAM J. Comput.*, 17 :373–386, 1988.

[83] S. Lucks : On the security of the 128-bit block cipher DEAL. *Fast Software Encryption 1999, pp. 60–70*.

[84] S. Lucks and R. Weis : How to make DES-based smartcards fit for the 21-st century. *Proceedings of IFIP CARDIS 2000, Fourth Smart Card Research and Advanced Application Conference*.

[85] F. J. McWilliams and N. J. A. Sloane : The Theory of Error-Correcting Codes. *North–Holland, Amsterdam, fifth edition*, 1986.

[86] R. J. McEliece : A public-key cryptosystem based on algebraic coding theory. *JPL DSN Progress Report*, pages 114–116, 1978.

[87] R. McEvoy, M. Tunstall, C. Murphy and W. P. Marnane : Differential power analysis of HMAC based on SHA-2 and countermeasures. In S. Kim, M. Yung and H.-W. Lee, editors. *Information Security Applications, 8th International Workshop, WISA 2007, Jeju Island, Korea, August 27-29, 2007, Revised Selected Papers*, volume 4867 of *Lecture Notes in Computer Science*, pages 317–332, Springer, 2008.

[88] U. Maurer, A. Oswald, K. Pietrzak and Sjödin J : Luby-Rackoff ciphers from weak round functions ? *In Advances in Cryptology EUROCRYPT 2006 volume 4004 of Lecture Notes in Computer Science, pp. 391–408, Springer-Verlag, 2006* http://eprint.iacr.org/2006/213.

[89] N. C. W. Meier : Algebraic attacks on stream ciphers with linear feedback. *Proceedings of Eurocrypt 2003*, Lecture Notes in Computer Science 2656 Springer :345–359, 2003.

[90] W. Meier and E. Pasalic and C. Carlet : Algebraic attacks and decomposition of boolean functions. *Eurocrypt 2004*, Lecture Notes in Computer Science 3027 Springer :474–491, 2004.

[91] A. Menezes, P. Oorschot and S. Vanstone : Mceliece public-key encryption. *CRC Press*, S. 299, 1997.

[92] R. C. Merkle : A certified digital signature. *Proceedings (G. Brassard ed.) Lecture Notes in Computer Science vol. 435 Springer 1989 :218–238*, 1989.

[93] T. Messerges : Using Second-Order Power Analysis to Attack DPA Resistant software. In Ç. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965, pages 238–251, 2000.

[94] M. Naor and O. Reingold : On the construction of pseudo-random permutations : Luby-Rackoff revisited. *Journal of Cryptology, vol. 12, 1999* http://www.wisdom.weizmann.ac.il/naor/PAPERS/lr.ps.

[95] R. Niebuhr : Application of Algebraic-Geometric Codes in Cryptography. *PhD thesis, Technischen Universität Darmstadt*, 2006.

[96] H. Niederreiter : Knapsack-type cryptosystems and algebraic coding theory. *Prob. Contr. Inform. Theory*, 15 n°2 :157–166, 1986.

[97] E. Oswald and K. Schramm : An Efficient Masking Scheme for AES Software Implementations. *Information Security Applications* Lecture Notes in Computer Science Volume 3786/2006 Pages 292-305

[98] A. Otmani, J. Tillich and L. Dallot : Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. *Symbolic Computation and Cryptography,, 2008*.

[99] J. Black, P. Rogaway and T. Shrimpton : Black-box analysis of the block-cipher-based hash-function constructions from PGV. *In Advances in Cryptology CRYPTO 2002, (M. Yung, ed.), no 2442 in Lecture Notes in Computer Science, pp 320–335, Springer-Verlag, 2002*.

[100] C. Park : Improving code rate of McEliece's public-key cryptosystem. *Electronic letters 25*, pages 1466–1467, 1989.

[101] J. Patarin : How to construct pseudorandom and super pseudorandom permutation from one single pseudorandom function. In R. Rueppel, editor, *Advances in Cryptology – EUROCRYPT '92*, volume 658, pages 256–266, 1992.

[102] J. Patarin and P. Camion : Design of near-optimal pseudorandom funcitons and pseudorandom permutations in the information-theoric model. http://eprint.iacr.org/2005/135.

[103] J. Patarin and A. Montreuil : Benes and Butterfly schemes revisited. http://eprint.iacr.org/2005/004.

[104] A. Perrig : The BiBa one-time signature and broadcast authentication protocol. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 28–37. ACM Press, 2001.

[105] C. Petit, F.-X. Standaert, O. Pereira, T. G. Malkin and M. Yung : A Block Cipher based PRNG Secure Against Side-Channel Key Recovery. Available at http://eprint.iacr.org/2007/356.pdf.

[106] E. Petrank and R. Roth : Is code equivalence easy to decide ? *Information Theory, IEEE Transactions on*, 43(5) :1602–1604, Sep 1997.

[107] J. Pieprzyk : How to construct pseudorandom permutations from single pseudorandom functions advances. In I. Damgård, editor, *Advances in Cryptology – EUROCRYPT '90*, volume 473, pages 140–150, 1990.

[108] J. Pierce : Limit distributions of the minimum distance of random linear codes. *IEEE Trans. Inf. theory*, IT-13 :595–599, 1967.

[109] D. Pointcheval and J. Stern : Security proofs for signature schemes. *Advances in Cryptology-EuroCrypt*, Lecture Notes in Computer Science 1070 Springer-Verlag :387–398, 1996.

[110] B. Preneel : Hash functions : past, present and future. *ASIACRYPT 2005*. http://homes.esat.kuleuven.be/~preneel/preneel_asia05.pdf

[111] B. Preneel : Hash functions - present state of art. *ECRYPT Report*, 2005.

[112] B. Preneel, R. Govaerts and J. Vandewalle : Hash functions based on block ciphers : A synthetic approach. *in Advanced in Cryptology CRYPTO 93 (D. Stinson, ed.), no 773 in Lecture Notes in Computer Science, pp 368–378, Springer-Verlag, 1994.*

[113] E. Prouff, C. Giraud and S. Aumonier : Provably Secure S-Box Implementation Based on Fourier Transform. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006*, volume 4249, 2006.

[114] E. Prouff and M. Rivain : A Generic Method for Secure SBox Implementation. In S. Kim, M. Yung and H.-W. Lee, editors. *Information Security Applications, 8th International Workshop, WISA 2007, Jeju Island, Korea, August 27-29, 2007, Revised Selected Papers*, volume 4867 of *Lecture Notes in Computer Science*,pages 227–244. Springer, 2008.

[115] T. Rao and K.-H. Nam : Private-key algebraic-code encryption. *IEEE Trans. Inform. Theory*, IT-35 Nr. 4 S. :829–833, 1989.

[116] L. Reyzin and N. Reyzin : Better than BiBa : Short One-Time Signatures with Fast Signing and Verifying. In L. M. Batten and J. Seberry, editors, *Information Security and Privacy, 7th Australian Conference, ACISP 2002*, volume 2384 of *Lecture Notes in Computer Science*, pages 144–153. Springer, 2002.

[117] R. Rivest, A. Shamir and Y. Tauman : How to leak a secret. In E. Boyd, editor. *Advances in Cryptology – ASIACRYPT 2001*, volume 2248,pages 552–565, 2001.

[118] F. Armknecht, C. Carlet, P. Gaborit, S. Kunzli, W. Meier and O. Ruatta : Efficient computation of algebraic immunity for algebraic and fast algebraic attacks. *Eurocrypt '06*, 2006.

[119] A. Rudra, P. K. Bubey, C. S. Jutla, V. Kumar, J. Rao and P. Rohatgi : Efficient Rijndael Encryption Implementation with Composite Field Arithmetic. In Ç. Koç, D. Naccache and C. Paar, editors. *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162, pages 171–184. 2001.

[120] R. A. Rueppel : Correlation immunity and the summation generator. *Crypto 1985*, Lecture Notes in Computer Science 218 Springer :260–272, 1986.

[121] K. Schramm and C. Paar : Higher Order Masking of the AES. *CT-RSA 2006* Lecture Notes in Computer Science Volume 3860/2006 pages 208–225.

[122] N. Sendrier : Cryptosystèmes à clé publique basé sur les codes correcteurs d'erreurs. *Habilitation á diriger des recherches, Inria*, 2002.

[123] N. Sendrier : Finding the permutation between equivalent linear codes : the support splitting algorithm. *Information Theory, IEEE Transactions on*, 46(4) :1193–1203, Jul 2000.

[124] H. Shacham and B. Waters : Efficient ring signatures without random oracles. *Public Key Cryptography - PKC 2007*, 4450 :166–180, 2007.

[125] A. Shamir : Identity-based cryptosystems and signature schemes. *Advances in Cryptology-Crypto'84*, 1984.

[126] V. Sidelnikov and S. Shestakov : On cryptosystems based on Generalized Reed-Solomon codes. *Diskretnaya Math*, 4 :57–63, 1992.

[127] G. J. Simmons : Contemporary cryptology the science of information integrity. *IEEE Press New Jersey*, 1992.

[128] J. Stern : A method for finding codewords of small weight. In G. D. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1988.

[129] J. Stern : A new identification scheme based on syndrome decoding. In D. Stinson, editor. *Advances in Cryptology – CRYPTO '93*, volume 773,pages 13–21, 1993.

[130] J. Stern : A new paradigm for public key identification. *IEEE Transactions on Information Theory, 42 (6) 1996, 1757-1768.*

[131] E. Trichina : Combinatorial Logic Design for AES SubByte Transformation on Masked Data. *Cryptology ePrint Archive, Report 2003/236*, 2003. http://eprint.iacr.org/.

[132] P. Tsang, V. Wei, T. Chan, M.H.Au, J. Liu and D.S.Wong : Separable linkable threshold ring signatures. *In Anne Canteaut and Kapalee Viswanathan editors Progress in Cryptology - INDOCRYPT 2004 5th International Conference on Cryptology in India Chennai India December 20-22 2004*, Proceedings volume 3348 of Lecture Notes in Computer Science . Springer :384–398, 2004.

[133] J. van Tilborg : On the McEliece public-key cryptosystem. Lect. Notes in CS, 1988.

[134] A. Vardy The intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory* 43(6) : 1757-1766 (1997).

[135] Y. L. S. Vaudenay : Faster correlation attack on bluetooth keystream generator E_0. *Crypto 2004*, Lecture Notes in Computer Science 3152 Springer :407–425, 2004.

[136] P. Véron : Problème SD, opérateur Trace, schémas d'identification et codes de Goppa. *PhD thesis, Université Toulon et du Var, France*, 1995.

[137] L.Welch and E.R.Berlekamp : Error correction for algebraic block codes. *U.S. Patent 4 633 470*, Sep 1983.

[138] C. Wieschebrink : Two NP-complete problems in coding theory with an application in code based cryptography. *Information Theory, 2006 IEEE International Symposium on*, pages 1733–1737, July 2006.

[139] D. Wong, K. Li, K. Fung, J. Liu and V. Wei : On the RScode construction of ring signature schemes and a threshold setting of RST. *In Sihan Qing Dieter Gollmann and Jianying Zhou editors Information and Communications Security 5th International Conference ICICS 2003 Huhehaote China October 10-13 2003*, Proceedings volume 2836 of Lecture Notes in Computer Science Springer 2003 :34–46., 2003.

[140] D. Zheng, X. Li and K. Chen : Code-based ring signature scheme. *International Journal of Network Security*, 5 No.2 :154–157, 2007.

Quatrième partie

Annexes

Annexe A

Algorithme amélioré dans le cadre des attaques algébriques

Ce chapitre a donné lieu à l'article suivant [30] :

Improved algorithm to find equations for algebraic attacks for combiners with memory
avec Frederik Armknecht, Philippe Gaborit et Olivier Ruatta
Third International Workshop on Boolean Functions Cryptography and Applications, Proceedings of BFCA 2007, Jean-Francis Michon, Pierre Valarcher, Jean-Baptiste Yunès Eds., pp. 81-98, 2008

Les attaques algébriques ont marqué la cryptanalyse des registres à décalage avec mémoire. Ce qui est important pour réussir une attaque algébrique, c'est de trouver des équations de degré le plus bas possible dépendant des bits de clés. De plus, de faibles degrés sont possibles si suffisamment de bits de chiffrés sont utilisés dans une équation. Un exemple est le registre à décalage avec mémoire E_0 utilisé dans le standard Bluetooth, où des équations de degré 4 existent pour seulement $r = 4$ et 5 tops d'horloge consécutifs, ainsi que des équations de degré 3 pour $r \approx 8\ 822\ 188$ tops d'horloge. L'existence d'équations de degré 3 avec $5 < r \ll 8\ 822\ 188$ tops d'horloge reste une question ouverte. Il est connu que des équations valides correspondent à l'existence d'annulateurs de certains ensembles. L'effort pour calculer ces ensembles et pour trouver des annulateurs sur ces ensembles est exponentiel en r, rendant inefficaces les algorithmes désirés. La plupart des algorithmes proposés utilisent l'élimination gaussienne traditionnelle et ont donc une complexité cubique. Les seules exceptions sont les algorithmes proposés récemment dans [3] et dans l'article de Frederic Didier [12] qui sont de complexité quadratique.

Les résultats de ce papier sont les suivants. Tout d'abord, nous décrirons de nombreuses améliorations pour le calcul des ensembles et de leurs annulateurs. Ensuite, nous utiliserons nos améliorations pour exclure l'existence d'équations de degré 3 pour E_0 avec $5 < r \leq 9$, ce qui constitue les meilleurs résultats jamais déterminés en ce qui concerne la non-existence d'équations de bas degré pour E_0.

A.1 Introduction

Durant les dernières années, les attaques algébriques ont gagné de plus en plus d'importance. De manière brève, une attaque algébrique consiste en la génération et la résolution de systèmes

d'équations *non*-linéaires qui décrivent de manière implicite la clé secrète en dépendance avec des valeurs connues. Dans le cas des générateurs pseudo-aléatoires, cela implique la connaissance de bits chiffrés connus.

Les registres à décalage sont utilisés pour un chiffrement en direct des bits du texte clair qui ont à passer un canal non sûr.

Les générateurs d'aléa les plus utilisés en pratique, sont les systèmes de chiffrement à flot basés sur les registres à décalage [1].

A.1.1 Déroulement

Avant d'échanger des données, le générateur d'aléa est initialisé par $\mathcal{S}_0 \in \{0,1\}^n$ commun à l'expéditeur et au destinataire. Pour chiffrer une suite de bits de clair $p_0, p_1, \ldots$, le générateur est utilisé pour générer une suite de bits de la même longueur $z_0, z_1, \ldots$. Les deux suites de bits sont alors 'Xorées' bit à bit, donnant la suite de bits de chiffrés $c_t = p_t \oplus z_t$. Les bits c_t sont ensuite envoyés au receveur, qui connaît la clé secrète d'initialisation et peut donc produire la même suite de bits z_t pour déchiffrer le message en effectuant : $p_t = c_t \oplus z_t$.

Pour évaluer la sécurité, il est admis qu'un adversaire potentiel connaît les spécifications du générateur et des bits z_t produits par le registre. Une attaque consiste à retrouver la valeur de $\mathcal{S}_0$ à partir des informations données. Depuis ces dernières années, plusieurs sortes d'attaques ont été inventées. Toutes ces attaques ont en commun leur complexité exponentielle en la taille de la clé n.

En 2003 dans [9], une nouvelle sorte d'attaque contre les systèmes de chiffrement à flot a été proposée : les attaques algébriques. Pour certains systèmes de chiffrement, les attaques algébriques améliorent toutes les attaques précédemment connues (i.e. [9, 1]). L'idée de base est de générer un système d'équations dont les solutions sont les bits de la clé secrète. Bien que le fait de résoudre un système d'équations non linéaires soit un problème NP-difficile, dans le cas dans lequel nous nous plaçons, les équations ont des propriétés particulières.

Chaque équation donne de l'information concernant la clé secrète dépendant de r bits du texte chiffré consécutifs $z_t, \ldots, z_{t+r-1}$ et le degré des équations est majoré par une valeur d indépendante de n. Si le nombre d'équations linéairement indépendantes est assez élevé, une attaque peut être réalisée à l'aide de $\mathcal{O}(n^{3d})$ opérations. Remarquons que, contrairement aux autres attaques, la complexité est seulement polynomiale en n mais exponentielle en d.

Par conséquent, un des sujets majeurs en ce qui concerne les attaques algébriques est la recherche de ces équations de bas degré. Alors que l'existence de telles équations a été prouvée dans [1], seulement peu de choses sont connues sur comment trouver ou prouver qu'il n'existe pas d'équations de degré d. Par exemple, dans le cas de E_0, une équation de degré 4 avec $r = 4$ tops d'horloge a été développée dans [1], ce qui donne l'attaque la plus rapide pour l'instant.

Plus tard, dans [7], ces équations sont combinées pour donner une nouvelle équation de degré 3 avec $r \approx 8\,822\,198$ tops d'horloge, ce qui réduit le temps de calcul. Pour autant que nous le savons, les attaques algébriques sont toujours les plus rapides contre E_0. [2] De plus, ces résultats montrent la possibilité que pour $r > 4$, des équations de degré < 3 peuvent exister. Il serait particulièrement intéressant que des équations de degré 2 voire 1 existent ce qui mènerait à des attaques réalisables en pratique. Alors qu'il a été prouvé qu'il n'existait pas d'équations de degré < 4 pour E_0 avec $r \leq 5$, à notre connaissance il n'y a pas de résultat pour $r \geq 6$.

1. Linear Feedback Shift Registers (LFSR)
2. Des attaques plus rapides sur le Bluetooth ont été décrite dans [18, 19, 20] mais elles dépendent de la linéarité du programme de clé du système de chiffrement utilisé dans le Bluetooth.

De plus, toutes les méthodes publiées pour trouver toutes les (ou exclure l'existence d') équations de degré d sont basées sur le calcul d'annulateurs de certains ensembles (voir [1, 21] et pour un point de vue plus général [2]). Cependant, comme la génération de ces ensembles et l'effort de calcul des annulateurs grandissent de manière exponentielle en r, des algorithmes efficaces pour chaque étape sont cruciaux.

Nous pouvons effectuer la recherche d'annulateurs à l'aide d'une élimination gaussienne. Récemment, un algorithme amélioré a été proposé dans [3] qui réduit la complexité de l'élimination de Gauss du cubique au quadratique, autorisant le calcul de l'immunité algébrique pour des fonctions booléennes avec plus de variables que ce qu'il était possible avant. Malheureusement, il est difficile de déterminer cette complexité en pratique et nécessite une gestion de la mémoire habile.

Nos résultats sont les suivants. Tout d'abord, nous décrirons comment calculer une base de toutes les équations de bas degré avec r tops d'horloge consécutifs et nous présenterons plusieurs améliorations sur la façon de construire l'ensemble X_Z introduit dans [2] et sur la façon de prouver la non-existence de certaines équations de bas degré. Alors, nous donnerons des résultats expérimentaux sur la non existence d'équations de degré 3 pour E_0 pour un certain nombre de tops d'horloge entre 5 et 9.

Ce chapitre est structuré de la manière suivante. Après avoir explicité les attaques algébriques et la connexion entre les équations et les annulateurs dans la section A.2, nous décrirons comment calculer les ensembles correspondants et quelques améliorations dans les sections 3 et 4.

A.1.2 Registres à décalage

Nous dirons d'un système de chiffrement à flot qu'il est basé sur des LFSRs s'il consiste en un certain nombre k de registres à décalage à rétroaction linéaire (LFSRs) et un combineur non linéaire qui transforme l'état interne, produit par les LFSRs et une suite de bits de sorties non linéaire. Étant donné la simplicité des LFSRs et des excellentes propriétés statistiques des bits de sorties que des LFSRs bien déterminés possèdent, les systèmes de chiffrement basés sur les LFSRs sont très utilisés en pratique.

Un registre (k, l) est constitué de k LFSRs, d'un automate fini à k entrées, d'un bit de sortie et de l bits de mémoire. Soit n la somme des longueurs des k LFSRs. À partir d'un secret $x^* \in \{0,1\}^n$, les LFSRs produisent un état interne linéaire $L(x^*)$, construit par blocs x^t de k bits parallèles à l'instant t.

À partir d'un état initial des bits de mémoire secret $c^1 \in \{0,1\}^l$, à chaque instant t l'automate produit le t-ième bits z_t correspondant à x^t et c^t et change l'état interne en c^{t+1} (voir figure A.1).

L'information secrète est donnée par x^* et c^1. De nombreux registres de ce type sont utilisés en pratique. Nous pouvons remarquer par exemple, que le générateur de clé E_0 utilisé dans le standard Bluetooth wireless LAN system (see Bluetooth SIG (2001)) est un registre $(4, 4)$.

Le but de ce chapitre est d'analyser la sécurité d'un registre (k, l) par rapport aux attaques algébriques.

Les attaques algébriques existent contre l'AES et Serpent (Courtois et Pieprzyk [8]) et Toyocrypt (Courtois [7]).

Courtois et Meier [9] traitent des attaques algébriques dans le cas des systèmes de chiffrement à flot basés sur des LFSRs et présentent la meilleure attaque connue contre Toyocrypt et LILI-128. Récemment, Courtois a introduit les attaques algébriques rapides contre les systèmes de chiffrement à flot basés sur des LFSRs, une version améliorée des attaques algébriques.

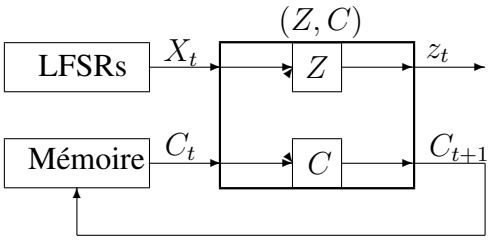


FIGURE A.1 – Un registre (k, l)

Une attaque algébrique est basée sur des relations non triviales de bas degré pour r tops d'horloge, c'est-à-dire une relation vraie pour n'importe quelle suite de r bits consécutifs produits par le registre et pour les kr bits internes correspondants. Connaissant une telle relation p de petit degré d et une suffisamment longue suite de bits produits par le registre $Z(x^*, c^1), p$ peut être utilisée pour produire un système d'équations surdéfini de T équations non linéaires en les bits initiaux des LFSRs, qui peuvent être vues comme un système d'équations linéaires en les monômes de longueur au plus d. Si T est assez grand alors nous avons une solution unique qui est induite par x^* et de laquelle x^* peut être dérivée d'une autre manière.

Évidemment, une plus grande valeur de d accroît la complexité en temps de manière significative. C'est pourquoi, la non-existence de relations non triviales de bas degré est un critère important pour des registres (k, l). Une contribution de ce papier est de donner un algorithme qui calcule pour un registre (k, l) donné, pour d et r fixés, l'ensemble des relations non triviales de degré d pour r tops d'horloges. Une conséquence est que des relations non triviales de degré $\lceil k(l+1)/2 \rceil$ pour $l + 1$ tops d'horloges ne peuvent être évitées. En remarquant que la complexité en temps est seulement polynomiale en n si k et l sont supposés constants. Alors pour chaque registre (k, l) il existe une valeur n', telle que l'attaque algébrique est plus efficace qu'une attaque exhaustive si $n \geq n'$.

C'est-à-dire, l'existence d'une relation non triviale de degré 10 pour 5 tops d'horloges consécutifs pour le générateur E_0, pour $n = 128$, donne une attaque algébrique de complexité en temps de 2^{141}, ce qui est moins bien qu'une attaque exhaustive dans ce cas.

Par contre l'attaque est meilleure qu'une attaque exhaustive si $n \geq 142$.

De manière étonnante, une relation non triviale de degré 4 pour 4 tops d'horloge existe.

Ce qui implique une attaque algébrique d'une complexité en temps aux alentours de $2^{67.58}$ opérations binaires et représente une sérieuse faiblesse de sécurité pour ce système de chiffrement à flot. D'un autre côté en utilisant notre méthode, nous pouvons prouver la non existence de relations non triviales de degré inférieur à 4, au moins pour 4 et 5 tops d'horloges. Dans le chapitre suivant, nous donnons les définitions de base sur les fonctions booléennes, les LFSRs, et quelques notions autour des attaques algébriques.

A.2 Attaques algébriques/annulateurs

Dans cette partie, nous décrirons les idées de base des attaques algébriques sur des générateurs aléatoires et le lien entre les équations valides et les annulateurs de certains ensembles.

Par conséquent, un annulateur d'un ensemble $X \subseteq \mathbb{F}^n$ avec $\mathbb{F}$ un corps fini à deux éléments est défini comme étant une fonction $g : \mathbb{F}_2^n \to \mathbb{F}_2$ avec $g(x) = 0$ pour tout $x \in X$.

Tout d'abord, considérons un registre (k, ℓ) :

Définition A.2.1

Un registre (k, ℓ) avec $k \geq 1$ et $\ell \geq 0$ est constitué d'un état interne $\mathcal{S} \in \mathbb{F}_2^\ell \times \mathbb{F}_2^n$, d'une matrice régulière L sur $\mathbb{F}_2$ de taille $n \times n$, appelée la matrice de rétroaction du LFSR, d'une matrice de projection P sur $\mathbb{F}_2$ de taille $n \times k$.

Un exemple pratique d'un tel registre (k, ℓ) est le générateur de pseudo-aléa E_0 utilisé dans le standard Bluetooth, qui est un registre $(4, 4)$ avec $n = 128$.

La génération des bits de pseudo-aléa fonctionne de la manière suivante : l'état interne $\mathcal{S}$ est initialisé à $\mathcal{S}_0 = (Q_0, K) \in \mathbb{F}_2^\ell \times \mathbb{F}_2^n$ où $K \in \mathbb{F}_2^n$ est l'état initial des LFSRs et le registre de $Q_0 \in \mathbb{F}_2^\ell$ est appelé mémoire, puis le contenu des LFSRs est mis à jour par une fonction linéaire, représentée ici par la matrice L. Dans la plupart des cas, seulement quelques LFSRs sont mis à jour et utilisés dans le calcul des bits de mémoire du registre ainsi que dans le calcul des bits de pseudo-aléa.

Ceci est exprimé par la matrice de projection P et signifie que seulement les valeurs de $K_t = K \cdot L^t \cdot P$ sont utiles dans le calcul du bit produit à l'instant t.

La mémoire est mise à jour via $Q_{t+1} = \Psi(Q_t, K_t)$, où le bit produit z_t est calculé par $z_t = f(Q_t, K_t)$.

Evidemment, les valeurs des bits produits $z_t, \ldots, z_{t+i}$ dépendent seulement de Q_t et de $K_t, \ldots, K_{t+i}$. Nous exprimons ce fait en définissant la fonction de sortie étendue :

$$f^r(Q_t, K_t, \ldots, K_{t+i}) = (z_t, \ldots, z_{t+i}).$$

De manière plus formelle, cela donne :

$$f^r(Q_t, K_t, \ldots, K_{t+i}) = (f(Q_t, K_t),\ f(\Psi(Q_t, K_t), K_{t+1}),\ f(\Psi(\Psi(Q_t, K_t), K_{t+1}), K_{t+2}),\ \ldots)$$

$$f^r(Q_t, K_t, \ldots, K_{t+i}) = (f(Q_t, K_t), f(Q_{t+1}, K_{t+1}), \ldots, f(Q_{t+r-1}, K_{t+r-1}))$$

avec $Q_{i+1} = \Psi(Q_i, K_i)$.

Une attaque algébrique est basée sur la recherche de fonctions $F_1, \ldots, F_s : \mathbb{F}_2^{k \cdot r} \times \mathbb{F}_2^r \to \mathbb{F}_2$ telles que :

$$F_i(K_t, \ldots, K_{t+r-1}, z_t, \ldots, z_{t+r-1}) = 0 \tag{A.1}$$

est vraie pour tous les instants $t \geq 0$ et $1 \leq i \leq s$.

Connaissant de telles fonctions, un système d'équations peut être généré par :

$$F(K_{t_i}, \ldots, K_{t_i+r-1}, z_{t_i}, \ldots, z_{t_i+r-1}) = 0$$

où $z_{t_i}, \ldots, z_{t_i+r-1}$ représente r valeurs consécutives connues de bits produits par le registre. Si suffisamment d'équations sont connues, l'état initial des LFSRs noté K est déterminé de manière unique et peut être retrouvé en résolvant un sytème d'équations. Une fois que K est connu, Q_0 peut être reconstruit assez facilement ou bien déterminé par une recherche exhaustive. Nous avons K la clé secrète et n la taille de la clé.

Pour calculer la solution, la méthode de linéarisation est possible. Sachant que chaque équation est de degré $\leq d$ avec $d = \max_i\{\deg_{K_t, \ldots, K_{t+r-1}}(F_i)\}$ indépendant de n, le nombre de monômes est borné supérieurement par $\mu(n, d) = \binom{n}{0} + \ldots + \binom{n}{d}$. Si le nombre d'inéquations linéairement indépendantes est égal au nombre de monômes utilisés, une solution peut être calculée en faisant une élimination gaussienne où chaque monôme est traité comme une variable indépendante. Comme $\mu(n, d) \in \mathcal{O}(n^d)$, la complexité totale est en $\mathcal{O}(n^{3d})$, ce qui est polynomial en la taille de la clé n mais exponentiel en d.

Alors, il est préférable de composer le système d'équations avec des équations linéairement indépendantes de degré le plus bas possible. En particulier, pour des $z_t^{t+r-1} = (z_t, \ldots, z_{t+r-1})$ connus et les $K_t^{t+r-1} = (K_t, \ldots, K_{t+r-1})$ inconnus correspondants, on voudrait éviter le cas où $F_1(K_t^{t+r-1}, z_t^{t+r-1})$, …, $F_s(K_t^{t+r-1}, z_t^{t+r-1})$ sont linéairement dépendants et exclure les expressions $F_i(K_t^{t+r-1}, z_t^{t+r-1})$ qui ne sont pas de degré minimal. De plus, pour être sûr que l'on a uniquement des équations linéaires indépendantes pour n'importe quel choix de z_t^{t+r-1}, il est possible de dériver des équations, pour chaque valeur possible de z_t^{t+r-1}, indépendantes. Ce qui nous amène à la notion de Z-fonctions :

Définition A.2.2

Soit un registre (k, ℓ) et $Z \in \mathbb{F}_2^r$. Une fonction $F : \mathbb{F}_2^{r \cdot k} \to \mathbb{F}_2$ est appelée une Z-fonction si pour n'importe quel $z_t^{t+r-1} = (z_t, \ldots, z_{t+r-1})$ de bits produits égal à Z, alors $F(K_t, \ldots, K_{t+r-1}) = 0$ est valide. De manière plus formelle, nous avons :

$$\forall t; z_t^{t+r-1} = Z \Rightarrow F(K_t, \ldots, K_{t+r-1}) = 0.$$

Si un adversaire connaît (pour une valeur de r fixée), des Z-fonctions $F_{Z,1}, \ldots, F_{Z,s_Z}$ pour chaque $Z \in \mathbb{F}_2^r$, alors il obtient le système d'équations suivant :

$$0 = F_{(z_{t_i}^{t_i+r-1}, j)}(K_{t_i}, \ldots, K_{t_i+r-1}), \quad 1 \leq j \leq s_Z$$

où $z_{t_i}^{t_i+r-1}$ représente une suite de r bits connus produits par le registre à partir du temps t_i. Plus d'information sur les Z-fonctions et un exemple concret se trouvent dans l'appendice Section A.4.

Bien sûr, maintenant arrive la question :

Comment trouver une base de Z-fonctions de degré minimal pour n'importe quel $Z \in \mathbb{F}_2^r$ dans le cas d'un registre (k, ℓ) donné et une valeur fixée de r?

Dans ce but, le théorème est donné. La preuve est assez simple et peut être trouvée par exemple dans [2].

Théorème A.2.1

Soit un registre (k, ℓ) et f^r la fonction de sortie étendue. Alors, une fonction $F : \mathbb{F}_2^{r \cdot k} \to \mathbb{F}_2$ est une Z-fonction avec $Z \in \mathbb{F}_2^r$ si et seulement si F est un annulateur de l'ensemble :

$$X_Z = \{(X_1, \ldots, X_r) \in \mathbb{F}_2^{r \cdot k} \mid \exists Q \in \mathbb{F}_2^\ell : f^r(Q, X_1, \ldots, X_r) = Z\}. \tag{A.2}$$

Pour résumer, une méthode générale pour trouver une base de Z-fonctions linéairement indépendantes pour un registre (k, ℓ) donné est de d'abord générer les ensembles X_Z et ensuite de calculer une base des annulateurs sur ces ensembles avec un degré minimal. [3] Dans la section suivante, nous allons d'abord analyser l'étape de construction des ensembles X_Z et proposer plusieurs améliorations. Après cela, nous tournerons notre attention sur le fait de calculer les annulateurs d'un ensemble donné et discuterons de quelques améliorations, en incluant une méthode quadratique qui

3. Actuellement, dans certains cas des méthodes plus directes existent pour trouver des équations valides (e.g., [9, 1, 7]) mais ces méthodes ne sont pas applicables en général et ne garantissent pas que l'on puisse trouver toutes les équations avec les plus petits degrés possibles.

remplace l'élimination gaussienne dans les algorithmes précédents et que l'on peut implémenter facilement.

Avant de commencer, nous voulons montrer qu'une approche similaire est possible pour dériver des équations implicites pour les S-boxes utilisées dans les systèmes de chiffrement par blocs. De telles équations furent importantes et controversées mais malgré cela les attaques algébriques restent des attaques importantes sur l'AES présentées dans [8].

A.3 Calcul des ensembles X_Z

Dans cette partie, nous considérons la question suivante :

Comment construire l'ensemble X_Z dans le cas d'un registre (k, ℓ) particulier et pour $Z \in \mathbb{F}_2^r$?

Une possibilité est d'effectuer une recherche exhaustive. Ce qui veut dire, de calculer pour toutes les possibilités $(Q, X_1, \ldots, X_r) \in \mathbb{F}_2^{\ell+r\cdot k}$ la valeur $Z' = f_\psi(Q, X_1, \ldots, X_r)$ et de garder les valeurs $(X_1, \ldots, X_r)$ avec $Z' = Z$. Le coût principal ici est de $\mathcal{O}(2^{k\cdot r+\ell})$ évaluations de f^r.

Bien sûr, si $Z' \neq Z$, il n'est pas nécessaire de gaspiller cette information comme cela implique $(X_1, \ldots, X_r) \in X_{Z'}$.

Alors, on peut calculer tous les 2^r ensembles possibles X_Z en même temps en calculant les sorties de $f^r(Q, X_1, \ldots, X_r)$ et en triant $(X_1, \ldots, X_r)$ en regardant les sorties. Alors, pour de grandes valeurs de r, i.e. $r \geq 10$, on peut préférer de générer les ensembles X_Z de manière indépendante, ce qui veut dire que seulement ces valeurs sont stockées avec les sorties correctes, ou même en les calculant en même temps que les annulateurs.

Dans le cas de E_0 les fonctions f et Ψ dépendent seulement du poids de Hamming des entrées X_t mais pas de leur valeurs concrètes. De manière plus précise, cela donne que pour tout $X_1, \ldots, X_r, Y_1, \ldots, Y_r \in \mathbb{F}_2^4$ avec $|X_i| = |Y_i|$, $f^r(Q, X_1, \ldots, X_r) = f^r(Q, Y_1, \ldots, Y_r)$ pour tout $Q \in \mathbb{F}_2^4$.

Alors, il suffit de considérer seulement le poids de Hamming des entrées, en réduisant le nombre d'entrées possibles de $16 \cdot 16^r$ valeurs possibles $(Q, X_1, \ldots, X_r) \in \mathbb{F}_2^4 \times \mathbb{F}_2^{4\cdot r}$ à $16 \cdot 5^r$ valeurs possibles $(Q, |X_1|, \ldots, |X_r|) \in \mathbb{F}_2^4 \times \{0, \ldots, 4\}^r$, en réduisant le nombre d'appels à f^r.

Indépendamment de cela, on peut réduire le nombres d'appels à f_Ψ à l'aide d'une approche divisée pour mieux régner (divide-and-conquer). Ainsi, en exploitant la simple observation qu'une entrée $(X_1, \ldots, X_r) \in \mathbb{F}_2^{k\cdot r}$ ne peut pas mener à une sortie spécifique $Z \in \mathbb{F}_2^r$, i.e. $(X_1, \ldots, X_r) \notin X_Z$, alors aucune des extensions possibles $(X_1, \ldots, X_r, X_{r+1}, \ldots, X_s)$ ne peut mener à n'importe quelle sortie $Z||Z' \in \mathbb{F}_2^s$ avec $Z' \in \mathbb{F}_2^{s-r}$; $Z||Z'$ étant une concaténation de Z et Z'.

Pour formaliser cela, on introduit quelques notations. Les fonctions de mise à jour étendues $\Psi^r : \mathbb{F}_2^\ell \times \mathbb{F}_2^{r\cdot k} \to \mathbb{F}_2^\ell$ sont définies par $Q_{r+1} = \Psi^r(Q_1, X_1, \ldots, X_r)$ où $Q_{t+1} = \Psi(Q_t, X_t)$. On peut voir Ψ^r comme r fois l'application de Ψ sur les entrées $Q_1, X_1, \ldots, X_r$. Par exemple, pour $r = 2$, cela donne :

$$\hat{\Psi}(Q_1, X_1, X_2) = \Psi(\Psi(Q_1, X_1), X_2).$$

Définition A.3.1

Soit un registre (k, ℓ) et $Z \in \mathbb{F}_2^r$ et $Q \in \mathbb{F}_2^\ell$ fixés. Posons les trois ensembles suivants :

$$X_{Q,Z} = \{(X_1,\ldots,X_r) \mid f^r(Q,X_1,\ldots,X_r) = Z\} \quad \text{(A.3)}$$
$$X_{Z,Q'} = \{(X_1,\ldots,X_r) \mid \exists Q : f^r(Q,X_1,\ldots,X_r) = Z \text{ et } \Psi^r(Q,X_1,\ldots,X_r) = Q'\} \quad \text{(A.4)}$$
$$X_{Q,Z,Q'} = \{(X_1,\ldots,X_r) \mid f_\Psi(Q,X_1,\ldots,X_r) = Z \text{ et } \Psi^r(Q,X_1,\ldots,X_r) = Q'\} \quad \text{(A.5)}$$

Ces trois ensembles spécifient encore un (sous-)système de toutes les entrées qui peut mener à la sortie fixée Z. Par exemple, pour les éléments de $X_{Q,Z}$ il est nécessaire qu'ils mènent à la sortie Z si le registre de mémoire est initialisé par Q au début du calcul.

Les ensembles $X_{Q,Z}$, $X_{Z,Q'}$ et $X_{Q,Z,Q'}$ peuvent être utilisés pour construire les ensembles X_Z de manière itérative :

Théorème A.3.1

Considérons un registre (k,ℓ). Alors, nous avons pour les valeurs $Q, Q' \in \mathbb{F}_2^\ell$, $Z \in \mathbb{F}_2^r$ et $Z' \in \mathbb{F}_2^s$:

$$X_Z = \bigcup_Q X_{Q,Z} = \bigcup_Q X_{Z,Q}, X_{Q,Z||Z'} = \bigcup_{Q'} X_{Q,Z,Q'} \times X_{Q',Z'},$$
$$X_{Z||Z',Q'} = \bigcup_Q X_{Z,Q} \times X_{Q,Z',Q'}, X_{Z||Z'} = \bigcup_Q X_{Z,Q} \times X_{Q,Z'}.$$

Si $r = r'$ nous réduisons le coût de calcul de $2^{k\cdot 2r}$ appels à f^{2r} à $2 \cdot 2^{k\cdot r}$ appels à f^r. Nous pouvons bien sûr réduire ce coût en divisant Z en de plus petites parties.

La dernière équation est importante car elle implique une manière itérative de calculer X_Z pour $Z \in \mathbb{F}_2^r$ pour de grandes valeurs de r. Soit par exemple $Z \in \mathbb{F}_2^r$ et $Z' \in \mathbb{F}_2^{r'}$, alors au lieu de calculer $X_{Z||Z'}$ directement avec $2^{k\cdot(r+r')}$ appels à $f^{r+r'}$, on peut calculer les ensembles $X_{Z,Q}$ et $X_{Q,Z'}$ de manière indépendante avec $2^{k\cdot r}$ exécutions de f^r et $2^{k\cdot r'}$ exécutions de $f^{r'}$, respectivement. Alors, si $r = r'$, nous réduisons l'effort de $2^{k\cdot 2r}$ invocations de f^{2r} à $2\cdot 2^{k\cdot r}$ invocations à f^r. Nous pouvons bien sûr réduire le coût de calcul en divisant l'ensemble Z en de plus petites parties.

A.4 Z-fonctions et l'ensemble X_Z

Dans cette section, nous illustrons le concept de Z-fonction et les ensembles X_Z. Nous considérerons le générateur somme (summation generator) [23] qui est un registre $(k, \lceil \log_2 k \rceil)$, basé sur l'addition d'entiers. Cela signifie qu'à la fois les bits d'entrée et l'état mémoire sont traités comme des entiers et ajoutés entre eux. Le résultat est formé de la sortie et l'état de mémoire suivant.

De manière plus formelle, à chaque top d'horloge t nous avons k bits d'entrées $x_{t,1},\ldots,x_{t,k}$ et de l'état de mémoire $Q_t \in \mathbb{F}_2^{\lceil \log_2 k \rceil}$. La somme entière de ces valeurs est calculée c'est-à-dire $S_t = x_{t,1} + \ldots + x_{t,k} + Q_t$ où Q_t est pris comme une valeur dans $\{0,\ldots,2^{\lceil \log_2 k \rceil}\}$. Alors, la sortie et le bit de mémoire suivant Q_{t+1} sont calculés de la manière suivante :

$$z_t = S_t \bmod 2 = (x_{t,1} + \ldots + x_{t,k} + Q_t) \mod 2 \quad \text{(A.6)}$$
$$Q_{t+1} = S_t \operatorname{div} 2 = \left\lfloor \frac{(x_{t,1} + \ldots + x_{t,k} + Q_t)}{2} \right\rfloor \quad \text{(A.7)}$$

Par exemple, dans le cas $k = 2$ et $\ell = 1$, nous avons :

$$z_{t+1} = f(Q_t, x_{t,1}, x_{t,2}) = Q_t \oplus x_{t,1} \oplus x_{t,2} \text{ and} \quad (A.8)$$
$$Q_{t-1} = \Psi(Q_t, x_{t,1}, x_{t,2}) = Q_t \cdot (x_{t,1} \oplus x_{t,2}) \oplus x_{t,1} \cdot x_{t,2}. \quad (A.9)$$

Nous considérons maintenant l'exemple $k = 2$ et $\ell = 1$. Dans la table A.1, un survol de toutes les valeurs possibles d'entrées $(X_1, X_2) \in \mathbb{F}_2^4$ sur deux tops d'horloge et un état mémoire initial $Q \in \mathbb{F}_2$ sont donnés, ensemble avec les sorties correspondantes $Z \in \mathbb{F}_2^2$. Par exemple, si l'état initial de la mémoire est 0 et si les entrées (venant des LFSRs) sont 00 au premier top d'horloge et 01 dans le second top d'horloge, alors la sortie Z du générateur somme est $Z = 01$. De manière alternative, on peut exprimer cela par $f^2(Q, X_1, X_2) = f^2(0, 00, 01) = (0, 1)$.

D'après la table A.1, on peut dériver directement des 4 différents ensembles X_Z :[4]

$$\begin{aligned} X_{(0,0)} &= \{(0000), (0011), (1101), (1110), (0101), (0110), (1001), (1010)\} \\ X_{(0,1)} &= \{(0001), (0010), (1100), (1111), (0100), (0111), (1000), (1011)\} \\ X_{(1,0)} &= \{(0100), (0111), (1000), (1011), (0000), (0011), (1101), (1110)\} \\ X_{(1,1)} &= \{(0101), (0110), (1001), (1010), (0001), (0010), (1100), (1111)\} \end{aligned} \quad (A.10)$$

Supposons maintenant que nous nous intéressons à une Z-fonction F avec $Z = (00)$. C'est-à-dire, F doit être nulle pour toutes les entrées (X_1, X_2) qui mènent à la sortie 00. Par définition, c'est l'ensemble $X_{(0,0)}$. Une (00)-fonction possible est

$$F(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}) = x_{1,1} \cdot x_{1,2} \oplus x_{1,1} \oplus x_{1,2} \oplus x_{2,1} \oplus x_{2,2}.$$

On peut remarquer que f et Ψ ne dépendent pas de la valeur exacte de X mais seulement de son poids de Hamming. Comme décrit dans la section A.3, nous pouvons exploiter cela pour économiser de la mémoire en stockant seulement les poids de Hamming des éléments de X_Z. Par exemple, nous pouvons reécrire les ensembles X_Z (A.10) comme :

$$\begin{aligned} X_{(0,0)} &= \{[00], [02], [21], [11]\} \\ X_{(0,1)} &= \{[01], [20], [22], [10], [12]\} \\ X_{(1,0)} &= \{[10], [12], [00], [02], [21]\} \\ X_{(1,1)} &= \{[11], [01], [20], [22]\} \end{aligned} \quad (A.11)$$

Ici, $[w_1, w_2]$ représente l'ensemble de toutes les entrées (X_1, X_2) telles que le poids de Hamming de X_1 est égal à w_1 et la même chose pour X_2 et w_2. Nous noterons ce type d'ensemble comme des ensembles symétriques par blocs des entrées et peut être divisés en blocs où chaque bloc est invariant sous l'action de permutations.

Si l'on souhaite utiliser les méthodes itératives décrites à la fin de la section A.3, nous avons besoin de calculer les ensembles $X_{Q,Z}$ et $X_{Z,Q}$. D'après la table A.1, on peut dériver immédiatement les ensembles $X_{Q,Z}$. Par exemple, pour $Q = 1$ et $Z = 11$, nous avons

$$X_{1,(11)} = \{[01], [20], [22]\}$$

qui est un sous-ensemble naturel de $X_{(11)}$.

4. les virgules seront enlevées pour un gain de clareté.

Q	0	0	0	0	0	0	0	0
X_1	(0,0)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)
X_2	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)
Z	00	01	01	00	10	11	11	10

Q	0	0	0	0	0	0	0	0
X_1	(1,0)	(1,0)	(1,0)	(1,0)	(1,1)	(1,1)	(1,1)	(1,1)
X_2	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)
Z	10	11	11	10	01	00	00	01

Q	1	1	1	1	1	1	1	1
X_1	(0,0)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)
X_2	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)
Z	10	11	11	10	01	00	00	01

Q	1	1	1	1	1	1	1	1
X_1	(1,0)	(1,0)	(1,0)	(1,0)	(1,1)	(1,1)	(1,1)	(1,1)
X_2	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)
Z	01	00	00	01	11	10	10	11

TABLE A.1 – Toutes les combinaisons entrée/sortie possibles pour 2 tops d'horloge dans le cas du générateur somme avec $k = 2$ entrées et $\ell = 1$ bits de mémoire

A.5 Programme

Je donne ici le code source du programme réalisé en magma.

```
// Déclaration
R := FiniteField(2);
nbvar:=4;

K := PolynomialRing(R, nbvar);
A := quo<K | [$.i^2-$.i : i in [1..nbvar]]>;
C := UniverseCode(R,4);
CC := UniverseCode(R,nbvar);
deg:=2; //deg est le degré maximal des monomes

// Nous construisons l'ensemble des mots de poids fixé
// L[i]:= ensemble des mots de poids i-1.
L := [SetToSequence(Words(C,i-1)) : i in [1,2,3,4,5]];
L := [[ [L[i][j][k] : k in [1..4]] : j in [1..#(L[i])]] : i in [1..5]];

// Fonction de mixage d'une séquence qui à partir de deux séquences
// L et M retourne la concaténation de L[i] avec M[j]
// pour tout i et j.
MixSeq := function(L,M)
  N := [];
  for i := 1 to #L do
    N := N cat [L[i] cat M[j] : j in [1..#M]];
  end for;
  return N;
end function;

// Fonction qui prend en entrée un ensemble de la forme
// I:=[I1,I2,I3,I4,I5,I6,I7] et retourne M
// de longueur nbvar M:=[m1,m2,m3,m4,...,m25,m26,m27,mnbvar]
// avec poids de (m1,m2,m3,m4):=I1.
GenPoints := function(I,L,R,K)
  N := #I;
  M := L[I[1]+1];
  for i := 2 to N do
    M := MixSeq(M,L[I[i]+1]);
  end for;
  return M;
end function;

// Fonction qui construit une séquence L[] contenant l'ensemble
// des éléments de poids w fixé.
WordsOfWeight := function(w,C,R)
```

```
  L := [];
  for i := 0 to w do
    L := L cat SetToSequence(Words(C,i));
  end for;
  return L;
end function;

// Fonction qui fait le produit
ProdBidon := function(A,V)
  r := A!1;
  for i := 1 to #V do
    r := r*V[i];
  end for;
  return r;
end function;

// Fonction qui détermine l'ensemble des monomes de degré
// le poids de L
AllMonomials := function(A,L,nbvar)
  return [ProdBidon(A,[(A.j)^(IntegerRing()!L[i][j]) :
                       j in [1..nbvar]]) : i in [1..#L]];
end function;

// Construction de la matrice de Vandermonde associé à notre ensemble
// de Points et aux monomes (Mons).
// Cette fonction construit la matrice avec en ligne les poits et
// en colonne les monomes.
// Matrix(i,j):= la valeur du monome Mons[j] au point Points[i].
Vander := function(Mons,Points)
  M := [];
  for i := 1 to #Points do
    M := M cat [Evaluate(Mons[j],Points[i]) : j in [1..#Mons]];
  end for;
  return Matrix(#Points,#Mons,M);
end function;

// Fonction qui lit dans un fichier l'ensemble des points considérés.
getl := function(FP,fin,nbvar)
  I:=[];
  b:=(IntegerRing()! (nbvar/4));
  for i := 1 to b do
    c:=Getc(FP);
   if IsEof(c) then fin:=c; else
    I[i]:=StringToInteger(c);
   end if;
  end for;
```

```
  k:=Getc(FP);
  return(I);
end function;

MyMult := function(V,M)
  N := #M;
  T := 0;
  for i := 1 to N do
    T := T + V[i][1] * M[i];
  end for;
  return T;
end function;

ChercheAnn := function(P,M) //P pour Points et M pour Monomes
  print("In ChercheAnn \n");
  MM := Vander(M,P);
  MM := Transpose(KernelMatrix(MM));
 print "Dimension du noyau -> ", NumberOfColumns(MM);
 print "Length du noyau -> ", NumberOfRows(MM);
  PP := [];
  for i := 1 to NumberOfColumns(MM) do
    V := ColumnSubmatrix(MM,i);
    PP := PP cat [MyMult(V,M)];
  end for;
  return PP;
end function;

ComputAnnil := function(FP,M,L,R,K)
  print("In ComputAnnil \n");
  t := false;
  fin:=IntegerToString(0);
  while ( (t eq false) and (#M gt 1) ) do
    P := [];
    while (#P lt #M) do
      l := getl(FP,fin,nbvar);
      if IsEof(fin) then
        t := true;
      else
        P := P cat GenPoints(l,L,R,K);
      end if;
    end while;
    M := ChercheAnn(P,M);
  end while;
  return M;
end function;
```

```
//Programme principal
for i :=1 to 128 do

Fsortie:=Open("sortie" cat IntegerToString(i) cat ".txt", "w");

FP:=Open("fichier" cat IntegerToString(i) cat ".txt", "r");

fin:=IntegerToString(0);

M := AllMonomials(A, WordsOfWeight(deg,CC,R),nbvar);

print "nombre de monomes consideres",#M;

M:=ComputAnnil(FP,M,L,R,K);

end for;
```

A.6 Calcul d'annulateurs

Bien que le calcul des ensembles X_Z soit faisable pour de petites valeurs de r, le calcul des annulateurs avec un degré minimal est bien plus coûteux en temps. Dans cette section nous donnons une manière de réduire ce coût.

Tout d'abord, nous devons remarquer que calculer des annulateurs de degré $\leq d$ pour un ensemble S donné peut être réduit à un problème d'algèbre linéaire.

Soit $S = \{x_1, \ldots, x_s\} \subset \mathbb{F}_2^n$ et $m_1, \ldots, m_{\mu(n,d)}$ l'ensemble des monômes en n variables de degré $\leq d$. On définit une matrice M sur $\mathbb{F}_2$ de taille $s \times \mu(n,d)$ en posant $M_{i,j} = m_j(x_i)$. Alors, on a que chaque annulateur $\bigoplus_{j=1}^{\mu(n,d)} c_j \cdot m_j$ de S de degré $\leq d$ donne un vecteur $V = (c_1, \ldots, c_{\mu(n,d)})^T$ tel que $M \cdot V = \vec{0}$ et réciproquement. En effet, une façon naturelle de regarder ce problème est de calculer le noyau de M. Cela a un coût en temps en $\mathcal{O}(s \cdot \mu(n,d)^2)$. De plus, dans notre cas particulier, on doit avoir $s \gg \mu(n,d)$. Par exemple, les ensembles X_Z pour E_0 avec $Z \in \mathbb{F}_2^7$ ont tous une taille $\geq 1\ 200\ 000$ avec $\mu(4 \cdot 7, 3) = 3\ 683$. Vérifier que des annulateurs de degré ≤ 3 existent pour $r = 7$ tops d'horloge nécessiterait environ $1\ 200\ 000 \cdot 3\ 683^2 \approx 2^{43.67}$ opérations.

Nous proposons ici une approche différente. Tout d'abord prenons $\mu(n,d)$ points aléatoires dans l'ensemble S et calculons M' de la même manière que M mais avec $\mu(n,d)$ points.

Le noyau K' de M' est calculé, nécessitant environ $\mu(n,d)^3$ opérations. Dans le cas de E_0, cette valeur est $\approx 2^{35.54}$. Si le noyau est le vecteur nul, nous savons immédiatement qu'il n'y a pas d'annulateur non triviaux de degré $\leq d$. Si le noyau contient d'autres vecteurs, alors chacun de ces vecteurs définit une fonction de degré $\leq d$ qui n'est nulle qu'en les $\mu(n,d)$ éléments choisis. Nous les appelons des *annulateurs potentiels*. S'il n'y en a pas beaucoup nous pouvons vérifer pour chaque annulateur potentiel s'il s'annule sur les $s - \mu(n,d)$ éléments restants. Si oui, nous avons trouvé un annulateur de l'ensemble S. Si non, nous avons prouvé qu'il n'en existait pas.

Admettons que le nombre d'annulateurs potentiels soit trop gros pour une recherche exhaustive. Tout d'abord on choisit un autre ensemble de $\mu(n,d)$ points aléatoires dans S, donnant une autre matrice M'' de taille $\mu(n,d) \times \mu(n,d)$. Ici aussi, le noyau K'' est calculé, coûtant un effort de calcul

additionnel de $\mu(n,d)^3$ opérations. Alors, l'intersection des deux noyaux K' et K'' est calculée, nécessitant environ $\mu(n,d)\cdot(\dim(K')+\dim(K''))^2$ opérations ce qui est significativement moins que $\mu(n,d)^3$ dans la plupart des cas. L'intersection nous donne maintenant tous les annulateurs pour les $2\cdot\mu(n,d)$ éléments choisis, cela constitue un nouvel ensemble d'annulateurs potentiels. A partir de là nous procédons comme décrit précédemment.

Nous avons vu que le coût de calcul est principalement dominé par le calcul des annulateurs pour les ensembles à $\mu(n,d)$ éléments. En utilisant une élimination gaussienne ceci est relativement rapide, mais possède deux désavantages : tout d'abord, nous devons deviner la valeur de d avant de commencer ; et ensuite le coût du calcul est cubique en $\mu(n,d)$. Nous pourrions remplacer cette étape par une meilleure méthode. Un candidat naturel est l'algorithme quadratique introduit dans [3]. Mais cet algorithme, bien que très efficace, nécessite des méthodes de programmation sophistiquées et une manipulation de la mémoire astucieuse.

Cette méthode permet d'éviter l'utilisation de l'algorithme introduit dans [3] qui contrairement à l'élimination de Gauss que nous faisons ici (coût cubique) a un coût quadratique, et permet de se servir du fait que la dimension de la base de monômes est bien plus petite que la dimension de l'ensemble X_Z. Dans la section suivante nous décrirons les résultats pratiques que nous avons obtenus dans le cas de E_0.

A.7 Résultats expérimentaux

Le cas de 6 tops d'horloge peut être traité facilement. Nous nous concentrerons dans la suite aux cas de $7, 8$ et 9 tops d'horloge pour lesquels nous appliquons la méthode par intersection décrite dans la section précédente :

A.7.1 Recherche d'annulateurs de degré 3

Pour la recherche d'annulateurs de degré 3 nous obtenons $4\times r$ variables et une base de $\mu(4\times r,d)$ monômes. Nous construisons les 2^r ensembles X_Z de taille $\approx 2^{r\times 3}$. Nous faisons une éliminaiton gaussienne dans le but de trouver des polynômes annulateurs de $\mu(4\times r,d)$ points, nous avons 2 cas :

- soit la matrice est inversible et il n'y a pas d'annulateur ;
- soit elle ne l'est pas et nous avons une base des annulateurs potentiels : nous gardons cette base en mémoire et, en choisissant un autre ensemble de $\mu(4\times r,d)$ points, nous calculons une nouvelle base d'annulateurs potentiels et nous appliquons la méthode par intersection décrite dans la section précédente.

tops	#var.	#mon.	#ens.	essais (moyenne)	temps (1 essai)	temps (total)
7	28	3683	128	2	3 min	12 heures
8	32	5489	256	2	3 min 30	29 heures
9	36	7807	512	3	4 min	102 heures

En appliquant la méthode par intersection, nous sommes capables de construire des ensembles de points aléatoires associés à la matrice ayant des noyaux à intersection nulle, ce qui prouve qu'il n'existe pas d'annulateurs de degré 3 pour $7, 8, 9$ tops d'horloge. Cette implémentation a été réalisée sur un PC à un 1 Ghz avec un programme en C.

A.7.2 Recherche d'annulateurs de degré 4

Nous savons que dans le cas de la recherche d'annulateurs de degré 4, nous avons un annulateur valide pour chaque ensemble.

Alors pour chaque ensemble nous allons chercher tous les annulateurs existants.

En appliquant la méthode par intersection de manière similaire au cas précédent, nous avons déterminer 4 annulateurs pour chaque ensemble considéré.

tops	#var.	#mon.	#annulateurs	#ensembles	temps(pour un ensemble)
7	28	24150	4	128	18 heures

Nos résultats sont les suivants :

– dans le cas d'annulateurs de degré 3 nous avons montré qu'il n'en existait pas ;

– dans le cas d'annulateurs de degré 4 nous avons trouvé 4 annulateurs pour chaque ensemble.

Bibliographie

[1] Frederik Armknecht, Matthias Krause : *Algebraic attacks on Combiners with Memory*, Proceedings of Crypto 2003, LNCS 2729, pp. 162-176, Springer, 2003.

[2] Frederik Armknecht : *Algebraic Attacks and Annihilators*, Proceedings of WEWORC 2005, LNI P-74, pp. 13-21, 2005.

[3] Frederik Armknecht, Claude Carlet, Philippe Gaborit, Simon Künzli, Willi Meier, Olivier Ruatta : *Efficient computation of algebraic immunity for algebraic and fast algebraic attacks*, Eurocrypt '06.

[4] Bluetooth SIG, *Specification of the Bluetooth system*, Version 1.1, 1 February 22, 2001, available at http ://www.bluetooth.com

[5] Alex Biryukov, Adi Shamir : *Cryptanalytic Time/Memory/Data tradeoffs for Stream Ciphers*, Proceedings of Asiacrypt 2000, LNCS 1976, pp. 1-13, Springer, 2000.

[6] Vladimor V. Chepyzhov, Ben Smeets : *On A Fast Correlation Attack on Certain Stream Ciphers*, Proceedings of Eurocrypt 1991, LNCS 547 pp. 176-185, Springer, 1991.

[7] Nicolas Courtois : *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback*, Proceedings of Crypto '03, LNCS 2729, pp. 177-194, Springer, 2003.

[8] Courtois, Pieprzyk : *Cryptanalysis of block ciphers with overdefined systems of equations*, Asiacrypt 2002, LNCS 2501, pp. 267-287, Springer, 2002.

[9] Nicolas Courtois, Willi Meier : *Algebraic attacks on Stream Ciphers with Linear Feedback*, Proceedings of Eurocrypt 2003, LNCS 2656, pp. 345-359, Springer, 2003. An extended version is available at http ://www.cryptosystem.net/stream/

[10] Jean-Charles Faugère : *A new efficient algorithm for computing Gröbner bases (F_4)*, Journal of Pure and Applied Algebra 139, 1-3 (1999), 61–68.

[11] Jean-Charles Faugère, Gwénolé Ars : *An algebraic cryptanalysis of nonlinear filter generators using Gröbner bases*, 2003. Available at http ://www.inria.fr/rrrt/rr-4739.html.

[12] Frédéric Didier *Using Wiedemann's algorithm to compute the immunity against algebraic and fast algebraic attacks*, Indocrypt 2006, Cryptology ePrint Archive 2006/261.

[13] Scott R. Fluhrer, Stefan Lucks : *Analysis of the E_0 Encryption System*, Proceedings of Selected Areas of Cryptography '01, LNCS 2259, pp. 38-48, Springer, 2001.

[14] Jovan Dj. Golic : *Cryptanalysis of Alleged A5 Stream Cipher*, Proceedings of Eurocrypt 1997, LNCS 1233, pp. 239-255, Springer, 1997.

[15] Thomas Johansson, Fredrik Joensson : *Fast Correlation Attacks Based on Turbo Code Techniques*, Proceedings of Crypto 1999, LNCS 1666, pp. 181-197, Springer, 1999.

[16] Thomas Johansson, Fredrik Joensson : *Improved Fast Correlation Attacks on Stream Ciphers via Convolutional Codes*, Proceedings of Eurocrypt 1999, pp. 347-362, Springer, 1999.

[17] Matthias Krause : *BDD-Based Cryptanalysis of Key stream Generators* ; Proceedings of Eurocrypt 2002, pp. 222-237, LNCS 2332, Springer, 2002.

[18] Yi Lu, Serge Vaudenay : *Faster Correlation Attack on Bluetooth Keystream Generator* E_0, Proceedings of Crypto 2004, pp. 407-425, LNCS 3152, Springer, 2004.

[19] Yi Lu, Serge Vaudenay : *Cryptanalysis of Bluetooth Keystream Generator Two-Level* E_0, Proceedings of Asiacrypt 2004, pp. 483-499, LNCS 3329, Springer, 2004.

[20] Yi Lu, Willi Meier, Serge Vaudenay : *The Conditional Correlation Attack : A Practical Attack on Bluetooth Encryption*, Proceedings of Crypto 2005, pp. 97-117, LNCS 3621, Springer, 2005.

[21] Willi Meier, Enes Pasalic, Claude Carlet : *Algebraic attacks and decomposition of Boolean functions*, Eurocrypt 2004, LNCS 3027, pp. 474-491, Springer, 2004.

[22] Willi Meier, Othmar Staffelbach : *Fast Correlation Attacks on certain Stream Ciphers*, Journal of Cryptology, pp. 159-176, 1989.

[23] Rainer A Rueppel : *Correlation immunity and the summation generator*,Proceedings of Crypto 1985, pp. 260–272, LNCS 218, Springer, 1986.

[24] Adi Shamir, Jacques Patarin, Nicolas Courtois, Alexander Klimov : *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*, Proceedings of Eurocrypt '00, Springer LNCS 1807, pp. 392-407.

[25] Erik Zenner : *On the Efficiency of the Clock Control Guessing Attack*, Proceedings of ICISC 2002, LNCS 2587, Springer, 2002.

[26] Erik Zenner, Matthias Krause, Stefan Lucks : *Improved Cryptanalysis of the Self-Shrinking Generator ACISP 2001*, LNCS 2119, Springer, 2001.

Annexe B

Codes cycliques sur des anneaux de matrices

Après une interprétation naturelle des codes quasi-cycliques comme des codes cycliques sur des anneaux de matrices, on étendra des résultats connus dans le cas scalaire au cas de suites récurrentes linéaires et des codes cycliques sur des anneaux de matrices. On caractérisera, en particulier, le dual d'un code cyclique sur un anneau de matrices sur un corps fini.

Mots clés : suites récurrentes linéaires, suites récurrentes linéaires à coefficients matriciels, codes quasi-cycliques, codes cycliques.

B.1 Introduction, motivations

Soient p un nombre premier, $r \in \mathbb{N}$, $q = p^r$ et $\mathbb{F}_q$ le corps de Galois à q éléments.
Soit $n \in \mathbb{N}$. On définit l'application décalage (shift) de $\mathbb{F}_q^n$ dans $\mathbb{F}_q^n$ notée T par :

$$\forall c = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n, \quad Tc = (c_{n-1}, c_0, \ldots, c_{n-2}).$$

Soit $n \in \mathbb{N}$, et $\mathcal{C}$ un code de longueur n sur $\mathbb{F}_q$ (un sous espace vetoriel de $\mathbb{F}_q$). Rappelons que l'on dit que $\mathcal{C}$ est un code quasi cyclique d'indice ℓ sur $\mathbb{F}_q$ si $T^\ell \mathcal{C} = \mathcal{C}$ (si $\ell = 1$, $\mathcal{C}$ est alors cyclique).

Soit $m \in \mathbb{N}^*$ tel que $n = \ell m$. On pose $\mathbb{A} = \mathbb{F}_q^\ell$. Pour $v = (v_0, \ldots, v_{\ell-1})$ un vecteur ligne de $\mathbb{A}$, on désigne par ${}^t(v_0, \ldots, v_{\ell-1})$ le vecteur colonne correspondant.
On considère l'isomorphisme d'espaces vectoriels Θ de $\mathbb{F}_q^n$ dans $\mathbb{A}^m$ donné par :

$$\forall c = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n, \qquad \Theta c = ({}^t(c_0, \ldots, c_{\ell-1}), {}^t(c_\ell, \ldots, c_{2\ell-1}), \ldots, {}^t(c_{(m-1)\ell}, \ldots, c_{m\ell-1})).$$

On étend l'application décalage définie ci-dessus à $\mathbb{A}^m$ par :

$$\forall v = ({}^t v_0, \ldots, {}^t v_{m-1}) \in \mathbb{A}^m, \quad Tv = ({}^t v_{m-1}, {}^t v_0, \ldots, {}^t v_{m-2}).$$

On vérifie alors qu'étant donné un code $\mathcal{C}$ de longueur n sur $\mathbb{F}_q$, $\mathcal{C}$ est un code quasi-cyclique d'indice ℓ sur $\mathbb{F}_q$ si et seulement si $\Theta(\mathcal{C})$ est un code cyclique sur $\mathbb{A}$.
Comme $\Theta(\mathcal{C})$ est formé de mots de "vecteurs", le formalisme matriciel s'invitait naturellement. On regardera donc $\Theta(\mathcal{C})$ comme code à la fois sur $\mathbb{F}_q$ (approche classique) et sur $\mathbb{M}_\ell(\mathbb{F}_q)$ l'anneau non commutatif des matrices carrées d'ordre ℓ à coefficients dans $\mathbb{F}_q$.

En fait on définira sur $\Theta(\mathcal{C})$ une structure de module à gauche sur l'algèbre $\mathbb{M}_\ell(\mathbb{F}_q)[x]$ des polynômes à coefficients matriciels. Ce qui permettra de décrire de manière originale les codes quasi cycliques dont "l'annulateur" est engendré par un seul polynôme matriciel.

La correspondance entre suites récurrentes linéaires (suites générées par un registre à décalage linéaire) et codes cycliques est bien connue : le polynôme caractéristique d'une suite récurrente linéaire u de période $L \in \mathbb{N}^*$ est, à réciproque près, le quotient du polynôme $x^L - 1$ par le polynôme générateur du code cyclique correspondant à u. Le formalisme ci-dessus, nous permettra également de retrouver ce résultat et démontrer de *manière directe* l'existence de l'exposant d'un polynôme matriciel et donc de la période d'une suite récurrente linéaire.

Signalons que le fait de considérer dans la partie 2 ci-après des suites de vecteurs (engendrées par des polynômes matriciels) permet dans le cas binaire ($q = 2$) la génération de vecteurs de bits par une implémentation matricielle qui pourrait être une alternative à la manière classique utilisant des registres à décalages linéaires (LFSR) en parallèle.

B.2 Suites récurrentes linéaires à coefficients matriciels

Soit comme ci-dessus $\mathbb{A} = \mathbb{F}_q^\ell$ et soit $S(\mathbb{A}) = \mathbb{A}^{\mathbb{N}}$ l'ensemble des suites de $\mathbb{A}$. Dans cette section, après avoir défini sur $S(\mathbb{A})$ une structure de modules à gauche sur l'anneau des polynômes à coefficients dans $\mathbb{M}_\ell(\mathbb{F}_q)$, nous montrerons "directement" que tout polynôme (réversible) à coefficients dans cet anneau a un exposant et par conséquent toute suite récurrente linéaire à coefficients matriciels est périodique et donnerons la dimension du code cyclique associé.

Soit $v = (v(n))_{n\geq 0}$ dans $S(\mathbb{A})$. On définit la suite Tv par : $\forall n \in \mathbb{N}, Tv(n) = v(n+1)$. Ce morphisme d'espaces vectoriels va nous permettre de munir $S(\mathbb{A})$ d'une multiplication par les éléments de $\mathbb{M}_\ell(\mathbb{F}_q)[x]$ en posant :

$$\forall p \in \mathbb{M}_\ell(\mathbb{F}_q)[x], \quad \forall v \in S(\mathbb{A}), \quad p(x)v = p(T)v.$$

Exemple 3

Pour $\ell = 2$ et $p(x) = Ax + B$ où $A = (a_{i,j})_{1\leq i,j\leq 2}$ et $B = (b_{i,j})_{1\leq i,j\leq 2}$ sont deux matrices 2×2 et pour $(v(n))_n =^t (v_1(n)_n, v_2(n)_n)$, si on pose $p.v =^t (w_1, w_2)$ alors :

$$\begin{array}{rrcl} \forall n \in \mathbb{N}, & w_1(n) & = & a_{11}v_1(n+1) + a_{12}v_2(n+1) + b_{11}v_1(n) + b_{12}v_2(n) \\ & w_2(n) & = & a_{21}v_1(n+1) + a_{22}v_2(n+1) + b_{21}v_1(n) + b_{22}v_2(n). \end{array}$$

On peut vérifier facilement que l'on obtient alors la proposition suivante.

Proposition B.2.1

Avec les notations ci-dessus, $S(\mathbb{A})$ muni des opérations usuelles et de la multiplication par des polynômes matriciels est un $\mathbb{M}_\ell(\mathbb{F}_q)[x]$-module à gauche.

Définition B.2.1

Soit $v \in S(\mathbb{A})$. On dit que v est une suite récurrente linéaire (à coefficients matriciels) si son annulateur dans $\mathbb{M}_\ell(\mathbb{F}_q)[x]$ contient un polynôme de coefficient dominant inversible.

Remarque

1. *On démontre (voir [7] ou bien [8]) qu'une suite sur $\mathbb{A}$ est récurrente linéaire si et seulement si les suites composantes "scalaires" sont des suites récurrentes linéaires sur $\mathbb{F}_q$. Le calcul de la période de chacune de ses suites permet alors d'avoir la période de la suite des vecteurs. Nous allons cependant donner une preuve directe de ce résulat via le calcul de l'exposant d'un polynôme matriciel.*

2. *L'annulateur d'une suite récurrente linéaire, contrairement au cas classique $\ell = 1$, n'est pas un idéal principal (à gauche ou à droite). Cependant nous arrivons à dire des choses sur cet idéal dans certains cas particuliers.*

Définition B.2.2

Un polynôme $f \in \mathbb{M}_\ell(\mathbb{F}_q)[x]$ est dit réversible si ses coefficients, dominant et constant, sont des matrices inversibles.

Nous sommes en mesure de donner maintenant la proposition qui suit et dont la démonstration est basée en particulier sur le fait que la division euclidienne (à droite ou à gauche) est toujours possible quand le diviseur est unitaire.

Proposition B.2.2

Soit $f \in \mathbb{M}_\ell(\mathbb{F}_q)[x]$. On suppose que f est réversible. Alors, il existe $e \in \mathbb{N}^*$ tel que $f|(x^e - 1)$.

Preuve On peut supposer sans perte de généralité que f est unitaire.

Considérons la suite $(x^n)_{n\in\mathbb{N}}$. L'ensemble quotient $\mathbb{M}_l(\mathbb{F}_q)[x]/f\mathbb{M}_l(\mathbb{F}_q)[x]$ a une structure de module et de $\mathbb{F}_q-$espace vectoriel. De plus, c'est un $\mathbb{F}_q-$espace vectoriel de dimension finie. Il existe donc deux entiers s et $t \in \mathbb{N}^*$ tels que : $x^t = x^s \bmod f$. Par conséquent, on a f divise $x^t - x^s$. On suppose que $t > s$; alors $f|x^s(x^{t-s} - 1)$.

Montrons à présent que :

$$f|x^s(x^{t-s} - 1) \Rightarrow f|(x^{t-s} - 1).$$

Supposons que $x^s - x^t = q(x)f(x)$ avec $f(0)$ inversible, f unitaire et $s < t$. Tout d'abord, si $s = 0$, il n'y a rien à démontrer. On suppose donc $s \geq 1$. Alors :

$$x^s(1 - x^{s-t}) = q(x)f(x) \tag{B.1}$$

D'autre part, la division euclidienne de $(1 - x^{t-s})$ par $f(x)$, qui est possible car f est unitaire, s'écrit :

$$(1 - x^{t-s}) = b(x)f(x) + r(x) \text{ avec } \deg(r) < \deg(f).$$

Ainsi :

$$x^s(1 - x^{s-t}) = x^s b(x)f(x) + x^s r(x) \tag{B.2}$$

En évaluant cette expression en $x = 0$, on trouve, $q(0)f(0) = 0$. Or comme $f(0)$ est inversible, il vient que $q(0) = 0$.

Par conséquent, $q(x) = xq'(x)$. Puisque $s \geq 1$, on a :

$$x([x^{s-1}b(x) - q'(x)]f(x) + x^{s-1}r(x)) = 0.$$

Nous sommes donc ramenés à l'égalité :

$$[x^{s-1}b(x) - q'(x)]f(x) + x^{s-1}r(x) = 0.$$

Si $s = 1$, alors le résultat est démontré : $(1 - x^{s-t}) = q'(x)f(x)$.
Sinon, on itère le raisonnement jusqu'à obtenir $q(x) = x^s h(x)$. Alors, en mettant x^s en facteur, on obtient :

$$[b(x) - h(x)]f(x) + r(x) = 0.$$

Si $b(x) \neq h(x)$, alors on observe que $\deg([b(x) - h(x)]f(x)) \geq \deg(f)$ car f est unitaire (donc le terme dominant ne peut pas s'annuler) $\deg(r) < \deg(f)$ ce qui est impossible.
Par conséquent, $b(x) = h(x)$ et $r(x) = 0$. Ainsi on a bien

$$f|(x^e - 1).$$

□

Définition B.2.3

L'exposant d'un polynôme matriciel réversible f est le plus petit entier non nul e tel que f divise à droite $x^e - 1$.

Notation B.2.1

Soit $f \in \mathbb{M}_\ell(\mathbb{F}_q)[x]$. On pose $\Omega(f) = \{u \in S(\mathbb{A}) \quad / \quad f.u = 0\}$.

Proposition B.2.3

Soit $f \in \mathbb{M}_\ell(\mathbb{F}_q)[x]$. On suppose que f est réversible.
1. Il existe $m \in \mathbb{N}$ tel que tout élément de $\Omega(f)$ soit périodique de période m.
2. L'ensemble $\mathcal{C}(f) = \{(u(0), \ldots, u(m-1)) \in \mathbb{A}^m \quad ; \quad u \in \Omega(f)\}$ est un code cyclique (sur $\mathbb{F}_q$ et sur $\mathbb{M}_\ell(\mathbb{F}_q)$).
3. On a : $\dim_{\mathbb{F}_q} \Omega(f) = \ell \deg(f)$.

Preuve Pour le 1., il suffit de prendre m égal à l'exposant de f. On vérifie facilement le 2 et on exhibe une base pour montrer le troisième point. □

Remarque

1. *Le calcul de la période d'une suite récurrente linéaire à coefficients matriciels peut être obtenu, comme dans [8], en passant par le calcul du "polynôme déterminant" de la matrice compagnon de la suite récurrente.*
2. *Quand on remplace $\mathbb{M}_\ell(\mathbb{F}_q)$ par $\mathbb{F}_q$ on sait que $\Omega(f)$ est un $\mathbb{F}_q$-espace vectoriel de dimension $\deg(f)$ et que $\mathcal{C}(f)$ est un code cyclique engendré par $\left(\frac{x^m-1}{f}\right)^*$ où pour un polynôme p, le polynôme p^* désigne son réciproque. Nous verrons dans la section suivante comment généraliser ce résultat au cas des codes cycliques matriciels.*
3. *On sait, dans le cas scalaire, qu'une suite est récurrente linéaire si et seulement si sa série génératrice est une fraction rationnelle. En utilisant des méthodes analogues à celles développées dans la preuve de la proposition B.2.2, on obtient une preuve directe d'une proposition qui constitue une généralisation de ce résultat au cas de suites de vecteurs à coefficients matriciels.*

B.3 Codes quasi-cycliques comme codes cycliques sur un anneau de matrices

La structure des codes quasi-cycliques a déjà été étudiée, mais dans ces résultats, une autre représentation de la quasi-cyclicité est utilisée. En effet, on effectue le réagencement suivant des indices des colonnes afin d'obtenir des blocs cycliques : $a\ell + b \longmapsto bm + a$, avec $a \in \{0, \ldots, m-1\}$, $b \in \{0, \ldots, \ell - 1\}$. Ainsi le *shift* de ℓ positions devient un *shift* d'une seule position dans chacun des ℓ blocs de m colonnes. Chacun des blocs définit donc un code cyclique sur $\mathbb{F}_q$ et ainsi, une représentation par des polynômes sur $\mathbb{F}_q$ est possible. Dans [3], *Kristine Lally* et *Patrick Fitzpatrick* utilisent les bases de *Gröbner* afin de classifier les codes et d'en donner une forme canonique. Dans [11], *San Ling* et *Patrick Solé* quant à eux utilisent la décomposition $\mathbb{F}_q[x]/ < x^m - 1 >= \oplus_{i=1}^{s} (\mathbb{F}_q[x]/ < g_i(x) >)$, où $x^m - 1 = g_1(x) \ldots g_s(x)$.

Grâce à la représentation que nous utilisons, nous allons essayer de généraliser les notions de polynômes générateurs et annulateurs, que nous avons pour les codes cycliques, aux codes quasi-cycliques. Nous verrons que, comme pour les suites récurrentes linéaires à coefficients matriciels, quelques résultats restent vrais. Cependant, il faut prendre des précautions à cause de la non commutativité de certains objets que nous manipulerons (par exemple des anneaux de matrices).

Soient $n = \ell m$ et $\mathscr{C}$ un code ℓ-quasi-cyclique sur $\mathbb{F}_q$. Posons $\mathbb{A} = \mathbb{F}_q^\ell$ comme précédemment. Si on considère l'isomorphisme d'espaces vectoriels suivant

$$\begin{array}{rccc} \Theta : & \mathbb{F}_q^n & \longrightarrow & \mathbb{A}^m \\ & (c_0, c_1, \ldots, c_{n-1}) & \longmapsto & \Theta c = ({}^t(c_0, \ldots, c_{\ell-1}), {}^t(c_\ell, \ldots, c_{2\ell-1}), \ldots, {}^t(c_{(m-1)\ell}, \ldots, c_{m\ell-1})) \end{array}$$

$\Theta\mathscr{C}$ est un code cyclique sur $\mathbb{A}$.

Remarque

Tout code cyclique $\mathscr{C}$ de longueur m sur $\mathbb{A}$ peut être vu comme un sous-espace vectoriel particulier de $S(\mathbb{A})$ l'ensemble des suites sur $\mathbb{A}$ de la façon suivante :
À $c = (c_0, \ldots, c_{m-1}) \in \mathscr{C}$, on associe $u \in S(\mathbb{A})$, telle que $u_i = c_{i \bmod m}, \forall i \in \mathbb{N}$.

Dorénavant, nous supposerons que $f \in \mathbb{M}_\ell(\mathbb{F}_q)[x]/ < x^m - 1 >$ et que f divise à gauche $x^m - 1$ (f est un diviseur de 0). C'est-à-dire qu'il existe $Q \in \mathbb{M}_\ell(\mathbb{F}_q)[x]$ tel que $fQ = x^m - 1$. Nous supposerons de plus que f et Q sont réversibles, et ainsi $\deg(f) + \deg(Q) = m$.

De cette façon, les résultats obtenus sur les annulateurs pour les suites récurrentes linéaires à coefficients matriciels s'appliquent naturellement aux codes cycliques sur $\mathbb{A}$.
Soient $f \in \mathbb{M}_\ell(\mathbb{F}_q)[x]/ < x^m - 1 >$ et $\mathscr{C}$ un $\mathbb{F}_q$-sous espace vectoriel de $\mathbb{A}^m$, alors

- $\Omega(f) = \{y \in \mathbb{A}^m / f * y = 0\}$ est un $\mathbb{F}_q$-espace vectoriel de dimension $\ell \deg(f)$ (voir proposition B.2.2.
- $Ann(\mathscr{C}) = \{P \in M[x]/ < x^m - 1 >$ t.q. $P * c = 0, \forall c \in \mathscr{C}\}$ est un idéal à gauche de $\mathbb{M}_\ell(\mathbb{F}_q)[x]/ < x^m - 1 >$.

Nous avons vu précédemment que $Ann(\mathscr{C})$ n'était pas un idéal principal en général, cependant nous avons le résultat suivant dans notre cas.

Proposition B.3.1

$Ann(\Omega(f))$ est un idéal principal et est exactement l'idéal engendré par f :

$$Ann(\Omega(f)) =< f >$$

Preuve Repose sur le fait que $Ann(\Omega(f))$ ne peut contenir de polynôme de degré plus petit que celui de f. □

Rappelons que l'idée de cette section est d'essayer de retrouver des résultats connus des codes cycliques pour les codes quasi-cycliques, en particulier le résultat suivant.
Si $g, h \in \mathbb{F}_q[x]$ sont tels que $gh = x^n - 1$ avec $\gcd(q, n) = 1$, alors $< g(x) >^\perp =< h^*(x) >$.
Un résultat analogue existe pour les codes quasi-cycliques, mais nous avons tout d'abord besoin de quelques lemmes techniques.

Notation B.3.1

Si $Q \in \mathbb{M}_\ell(\mathbb{F}_q)[x], Q(x) = \sum_{i=0}^{m-1} q_i x^i$, *avec* $q_i \in \mathbb{M}_\ell(\mathbb{F}_q)$ *et on note* $q_i = (q^i_{a,b})_{a,b=0,\ldots,\ell-1}$.

Si $c \in \mathbb{A}^m, c = (c_0, c_1, \ldots, c_{m-1})$ *avec* $c_i \in \mathbb{A}, \forall i = 0, \ldots, m-1$ *et pour* $i \in \{0, \ldots, m-1\}$, *on note* $c_i = (c_{i,0}, \ldots, c_{i,l-1})$ *avec* $c_{i,j} \in \mathbb{F}_q, \forall i = 0, \ldots, m-1\ , \forall j = 0, \ldots, l-1$.

Si $Q(X) = \sum_{i=0}^{m-1} q_i x^i \in \mathbb{M}_\ell(\mathbb{F}_q)[x]$, *on assimile* q_j *à* $q_{j \mod m}$ *(pris entre 0 et* $m-1$*).*

De même si $c = (c_0, \ldots, c_{m-1}) \in \mathbb{A}^m$, *on assimile* c_j *à* $c_{j \mod m}$ *(pris entre 0 et* $m-1$*).*

Proposition B.3.2

$$\begin{aligned} \Omega(f) &= Q.\mathbb{A}^m \\ &= \{Q.x | x \in \mathbb{A}^m\} \end{aligned}$$

Preuve Soit $y \in Q * \mathbb{A}^m$, alors $\exists\ y_0 \in \mathbb{A}^m$ tel que $y = Q.y_0$ et donc $f.y = f.Q.y_0 = (x^m - 1).x = 0$ donc $y \in \Omega(f)$. D'où $Q.\mathbb{A}^m \subseteq \Omega(f)$.

Considérons :

$$\begin{aligned} \Phi_Q : \mathbb{A}^m &\to \mathbb{A}^m \\ y &\mapsto Q * y \end{aligned}$$

Φ_Q est $\mathbb{F}_q$-linéaire.

$\ker(\Phi_Q) = \Omega(Q)$ et $\mathrm{im}(\Phi_q) = Q * \mathbb{A}^m$ donc

$$\begin{aligned} \dim(Q.\mathbb{A}^m) = &= \ell m - \dim(\Omega(Q)) \\ &= \ell m - \ell \deg(Q) \\ &= \ell m - \ell(m - \deg(f)) \\ &= \ell \deg(f) \\ &= \dim(\Omega(f)). \end{aligned}$$

Comme nous avons $Q.\mathbb{A}^m \subseteq \Omega(f)$ et $\dim(Q.\mathbb{A}^m) = \dim(\Omega(f))$ on en déduit :

$$\Omega(f) = Q.\mathbb{A}^m$$

□

Lemme B.3.1

Soit $d \in (\mathbb{F}_q^l)^m$, *alors :*

$$d \in \Omega(f)^{\perp} \Leftrightarrow \sum_{i=0}^{m-1}\sum_{j=0}^{m-1}\sum_{a=0}^{\ell-1}\sum_{b=0}^{\ell-1} q_{a,b}^{j} y_{i+j,b} d_{i,a} = 0 \; \forall y \in \mathbb{A}^m$$

Preuve

On notera $< .;. >$ le produit scalaire usuel et $. * .$ le produit entre un élément de $\mathbb{M}_l(\mathbb{F}_q)[x]$ et un élément de $\mathbb{A}^m$

$$\begin{aligned}
d \in \Omega(f)^{\perp} &\Leftrightarrow \forall c \in \Omega(f) < c; d >_{\mathbb{A}^m} = 0 \\
&\Leftrightarrow \forall y \in \mathbb{A}^m < Q * y; d >_{\mathbb{A}^m} = 0 \text{ d'après le Lemme B.3.2} \\
&\Leftrightarrow \forall y \in \mathbb{A}^m \sum_{i=0}^{m-1} < (Q * y)_i; d_i >_{\mathbb{A}} = 0 \\
&\Leftrightarrow \forall y \in \mathbb{A}^m \sum_{i=0}^{m-1} < \sum_{j=0}^{m-1} q_j . y_{i+j}; d_i >_{\mathbb{A}} = 0 \\
&\Leftrightarrow \forall y \in \mathbb{A}^m \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} < q_j . y_{i+j}; d_i >_{\mathbb{A}} = 0 \\
&\Leftrightarrow \forall y \in \mathbb{A}^m \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} < (\sum_{b=0}^{\ell-1} q_{a,b}^{j} . y_{i+j,b})_{a=0,\ldots,\ell-1}; (d_{i,a})_{a=0,\ldots,\ell-1} >_{\mathbb{A}} = 0 \\
&\Leftrightarrow \forall y \in \mathbb{A}^m \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \sum_{a=0}^{\ell-1} \sum_{b=0}^{\ell-1} q_{a,b}^{j} y_{i+j,b} d_{i,a} = 0
\end{aligned}$$

□

Lemme B.3.2

Soit $d \in \mathbb{A}^m$, *alors ;*

$$d \in \Omega({}^tQ^*) \Leftrightarrow \forall j, k = 0, \ldots, m-1 \sum_{i=0}^{m-1} \sum_{a=0}^{\ell-1} q_{a,j}^{-i} d_{i+k,a} = 0.$$

Preuve Soient $R \in \mathbb{M}_\ell(\mathbb{F}_q)[x]$ et $d \in \mathbb{A}^m$.

$$\begin{aligned} R * d &= \textstyle\sum_{i=0}^{m-1} r_i x^i * d \\ &= \textstyle\sum_{i=0}^{m-1} r_i x^i * (d_0, \ldots, d_{m-1}) \\ &= (\textstyle\sum_{i=0}^{m-1} r_i d_{i+k})_{k=0,\ldots,m-1} \\ &= (\textstyle\sum_{i=0}^{m-1} (\sum_{a=0}^{\ell-1} r_{j,a}^i d_{i+k,a})_{j=0,\ldots,m-1})_{k=0,\ldots,m-1} \end{aligned}$$

$$\text{D'où } R * d = 0 \quad \Leftrightarrow \quad \forall j, k = 0, \ldots, m-1 \textstyle\sum_{i=0}^{m-1} \sum_{a=0}^{\ell-1} r_{j,a}^i d_{i+k,a} = 0$$

Calculons ${}^tQ^* * d$, on a

$$Q(x) = \sum_{i=0}^{m-1} q_i x^i, q_i = (q_{j,a}^i)_{j=0,\ldots,m-1;a=0,\ldots,m-1}$$

$${}^tQ(x) = \sum_{i=0}^{m-1} {}^tq_i x^i, \text{ avec } {}^tq_i = (q_{a,j}^i)_{j=0,\ldots,m-1;a=0,\ldots,m-1}$$

$${}^tQ^*(x) = \sum_{i=0}^{m-1} {}^tq_{\deg(Q)-i} x^i, \text{ avec } {}^tq_{\deg(Q)-i} = (q_{a,j}^{\deg(Q)-i})_{j=0,\ldots,m-1;a=0,\ldots,m-1}$$

d'où,

$$\begin{aligned} {}^tQ^* * d \quad &\Leftrightarrow \quad \forall j, k = 0, \ldots, m-1 \textstyle\sum_{i=0}^{m-1} \sum_{a=0}^{\ell-1} q_{a,j}^{\deg(Q)-i} d_{i+k,a} = 0 \\ &\Leftrightarrow \quad \forall j, k = 0, \ldots, m-1 \textstyle\sum_{i=0}^{m-1} \sum_{a=0}^{\ell-1} q_{a,j}^{-i} d_{i+\deg(Q)+k,a} = 0 \\ &\Leftrightarrow \quad \forall j, k = 0, \ldots, m-1 \textstyle\sum_{i=0}^{m-1} \sum_{a=0}^{\ell-1} q_{a,j}^{-i} d_{i+k,a} = 0 \end{aligned}$$

□

Nous avons donc maintenant tous les outils pour démontrer le théorème principal.

Théorème B.3.1

Soient $f, Q \in \mathbb{M}_\ell(\mathbb{F}_q)[x]$ réversibles tels que $f.Q = x^m - 1$. Alors :

$$\Omega(f)^\perp = \Omega({}^tQ^*)$$

Preuve Montrons tout d'abord $\Omega(f)^\perp \subset \Omega({}^tQ^*)$

Soit $d \in \Omega(f)^\perp$, $\Omega(f)^\perp$ étant ℓ-quasi-cyclique,

$$\forall k = 0, \ldots, m-1, x^k.d \in \Omega(f)^\perp$$

D'où d'après le Lemme B.3.1,

$$\forall y \in \mathbb{A}^m,\ \forall k = 0, \ldots, m-1, \sum_{i=0}^{m-1}\sum_{j=0}^{m-1}\sum_{a=0}^{\ell-1}\sum_{b=0}^{\ell-1} q_{a,b}^j y_{i+j,b} d_{i+k,a} = 0 \tag{B.3}$$

Soit $j \in \{0, \ldots, m-1\}$, pour $y = (e_j; 0; \ldots; 0)$, (e_j : $j^{\text{ème}}$ vecteur de la base canonique de $\mathbb{A} = \mathbb{F}_q^\ell$)
on a :

$$y_{i+j,b} \quad = \quad \delta_b^j \text{ si } i + j = 0 \mod m \text{ et 0 sinon}$$

Donc pour ce y, l'équation B.3 nous donne :

$$\forall k = 0, \ldots, m-1, \sum_{i=0}^{m-1}\sum_{a=0}^{\ell-1} q_{a,j}^{-i} d_{i+k,a} = 0$$

Or ceci est vrai pour j arbitraire.
D'où :

$$\forall k = 0, \ldots, m-1,\ \forall j = 0, \ldots, m-1, \sum_{i=0}^{m-1}\sum_{a=0}^{\ell-1} q_{a,j}^{-i} d_{i+k,a} = 0$$

Donc d'après le Lemme B.3.2, $d \in \Omega({}^tQ^*)$. Ainsi

$$\Omega(f)^\perp \subset \Omega({}^tQ^*).$$

D'où

$$\begin{aligned} \dim(\Omega(f)^\perp) &= \ell m - \dim(\Omega(f)) \\ &= \ell m - \ell \deg(f) \text{ via la proposition B.2.3} \\ &= \ell(m - \deg(f)) \\ &= \ell \deg(Q) \end{aligned}$$

et,

$$\begin{aligned} \dim(\Omega({}^tQ^*)) &= \ell \deg({}^tQ^*) \text{ via la proposition B.2.3} \\ &= \ell \deg(Q) \text{ car } Q \text{ est réversible} \\ &= \dim(\Omega(f)^\perp) \end{aligned}$$

D'où l'égalité. □

Bibliographie

[1] M. CAMUS *Suites Récurrentes Linéaires A Coefficients Matriciels*, Mémoire de DEA-Université de Limoges, 2004.

[2] S. COUTEAUD Mémoire de DEA-Université de Limoges, 2007.

[3] P. FITZPATRICK, K. LALLY *Algebraic structure of quasicyclic codes* Disc. Appl. Math., 111(2001), 157-175.

[4] T. HERLESTAM *On functions of linear shift register sequences F. ; Pichler, , Advances in Cryptology - EUROCRYPT ;'85, 219 Lecture Notes in Computer Science, 119-129. Springer-Verlag, 1986.*

[5] E.L. KEY *An analysis of the structure and complexity of nonlinear binary sequence generators IEEE Transactions on Information Theory, 22 :732-736, 1976.*

[6] J.L. MASSEY *The ubiquity of Reed-Muller Codes Applied Algebra, Algebraic Algorithms and Error-Correcting Codes - AAECC-14, 2227 Lecture Notes in Computer Science, 1-12. Springer-Verlag, 2001.*

[7] A. NECER *Systèmes récursifs et algèbre de Hadamard de suites récurrentes linéaires sur des anneaux commutatifs* Communications in Algebra, 27 (12), 6175-6189, 1999.

[8] H. NIEDERREITER *The Multiple-Recursive Matrix Method for Pseudorandom Number Generation* Finite fields and their applications 1, 3-30, 1995.

[9] R.A. RUEPPEL *Analysis and Design of stream ciphers. Springer-Verlag, 1986.*

[10] R.A. RUEPPEL, O.J. STAFFELBACH *Products of Linear Recurring Sequences with Maximum Complexity IEEE Transactions on Information Theory, 33(1) :124-131, 1987.*

[11] S. LING, P. SOLÉ *On the algebraic structure of quasy-cyclic codes I : finite fields* IEEE Transactions on Information Theory, 47 :2751-2760,2001.

Glossaire

AES
: Advanced Encrytion Standard – Algorithme de chiffrement symétrique développé par Rijmen et Daemen et retenu en octobre 2000 comme norme de chiffrement remplaçant le DES.

Anonymat
: Incapacité d'identifier un utilisateur au vu de ses communications.

Attaque à messages chiffrés choisis
: Attaque dans laquelle on a accès à un oracle qui déchiffre des messages arbitraires.

Attaque à messages choisis
: Attaque dans laquelle on peut faire signer des messages arbitraires.

Attaque à textes clairs choisis
: Attaque dans laquelle on a accès à un oracle qui chiffre des textes arbitraires.

Attaque sans message connu
: Attaque dans laquelle on ne dispose que de données publiques.

Authenticité
: Garantie de l'identité d'une personne ou du contenu d'un message.

CBC
: Cipher Block Chaining – Processus consistant à combiner par OU exclusif le texte clair avec le cryptogramme précédent avant de le chiffrer, ce qui ajoute un mécanisme de rétroaction à un chiffrement par blocs.

CFS
: Courtois, Finiasz et Sendrier – Schéma de signature basé sur le FDH proposé en 2001.

Certificat
: Donnée (signatures, ...) produite par une autorité reconnue et servant à s'assurer de la légitimité de données publiques.

Chiffrement
: Opération transformant un texte clair en message chiffré.

Clé publique
: Information personnalisante, connue de tout le monde.

Clé secrète
: Information personnalisante, connue seulement des personnes autorisées.

Code correcteur d'erreurs
: Code qui permet, outre la détection, la correction automatique de certaines fautes détectées.

Confidentialité
: Garantie du caractère secret d'un message.

Cryptanalyse
: Art de décrypter les messages chiffrés.

Cryptographie
: Art de chiffrer et de déchiffrer les messages.

Cryptologie
: Science qui étudie les deux arts précédents.

DES
: Data Encryption Standard – Chiffrement symétrique développé par IBM et normalisé en 1977.

DPA
: Differential Power Analysis – Cryptanalyse basée sur les différences de consommation électrique.

Déchiffrement
: Opération consistant à récupérer le texte clair à partir du chiffré à l'aide d'une clé.

Décryptage
: Opération consistant à retrouver le texte clair à partir du chiffré sans connaître la clé.

Diffie-Hellman
: Premier algorithme à clé publique, inventé en 1976. Il utilise les logarithmes discrets sur un corps fini.

Elgamal
: Utilisé pour les signatures numériques et le chiffrement ; basé sur les logarithmes discrets dans un corps fini ; peut être utilisé par la fonction DSA.

FDH
: Full Domain Hash – Technique de formatage des messages utilisée pour signer.

Falsification
: Action d'un ennemi qui génère la signature d'autrui.

Fonction à sens unique (forte)
: Fonction facile à calculer mais dont l'inverse est difficile à calculer.

Fonction de hachage
: Fonction transformant un message de taille arbitraire en une empreinte de taille fixée.

GD
: Goppa Code Distinguishing – Problème de distinguabilité des codes de Goppa.

GPBD
: Goppa Parameterized Bounded Decoding – Cas particulier du problème SD.

Indistinguabilité
: Incapacité de décider de la provenance d'une donnée parmi plusieurs provenances possibles.

Intégrité
: Garantie de non-modification d'un message.

KKS
: Kabatianski, Krouk et Smeets – Schéma de signature développé en 1998.

Key escrow
: Mécanisme qui permet à un tiers de récupérer des clés cryptographiques utilisées pour la confidentialité de données, dans le but de déchiffrer ces données cryptées.

MAC
: Message Authentification Code – Code d'authentification de messages.

Mise en accord de clé (ou échange de clés)
: Protocole permettant à plusieurs utilisateurs de construire un secret commun connu d'eux seuls.

Non-malléabilité
: Incapacité pour un adversaire de produire le chiffrement d'un texte clair 'relié' au texte clair d'un message donné.

Non-répudiation
: Impossibilité pour le signataire de renier la signature d'un document après l'avoir produite.

One- time pad
: Masque jetable (ou clé aléatoire une fois) Grand ensemble de lettres réellement aléatoire, non répétitif, considéré comme le seul système de chiffrement parfait, inventé par le Major J. Mauborgne et G. Vernam en 1917.

PKI
: Public-Key Infrastructure – Infrastructure de clé publique.

Partage de secret
: Mécanisme de division d'un secret en plusieurs parts, nécessitant un nombre minimal de parts pour être reconstitué.

Prédicat
: Formule booléenne prenant la valeur 'Vrai' ou 'Faux' en fonction de une ou plusieurs variables.

Preuve interactive
: Dialogue entre un prouveur P et un vérifieur V au cours duquel P convainc V de la validité d'un prédicat ou de sa connaissance d'un secret satisfaisant un prédicat.

RSA
: Rivest-Shamir-Adleman – Chiffrement assymétrique proposé en 1978.

SD
: Syndrome Decoding – Problème NP-complet du décodage par syndrome.

Sécurité sémantique
: Caractérise un système où un même message peut être chiffré de plusieurs façons.

Schéma à seuil
: Système nécessitant le concours d'un nombre minimal de participants pour être valide.

Secure channel
: Canal sûr – Moyen de communication entre deux entités tel qu'un adversaire ne puisse pas changer l'ordre, supprimer, insérer ou lire de l'information (exemples : SSL, IPsec, chuchoter dans l'oreille de quelqu'un).

Signature de cercle
: Signature permettant à un utilisateur de signer anonymement au nom d'un ensemble arbitraire de personnes.

Signature de groupe
: Signature autorisant un membre d'un groupe à signer des documents, anonymement, au nom de ce groupe.

Signature
: Sceau apposé à un message permettant d'en garantir l'authenticité et l'intégrité.

Significativité
: Sécurité pour le vérifieur dans une preuve interactive : un prouveur ignorant le secret ne le convainc qu'avec probabilité négligeable.

Triple DES
: Configuration de chiffrement dans lequel l'algorithme DES est utilisé trois fois de suite avec trois clés différentes.

Index

R

S

T

Résumé

Dans cette thèse, on s'intéresse à l'étude de systèmes de chiffrement ainsi que de schémas de signature dont la sécurité repose sur des problèmes difficiles de théorie des codes correcteurs d'erreurs. Ces activités de recherche ont été motivées, d'une part d'un point de vue théorique par la création de nouveaux schémas de signature avec des propriétés spéciales ainsi que d'une manière de réduire la taille de clés du schéma de McEliece, et d'autre part, d'un point de vue pratique visant à utiliser des propriétés structurelles afin d'obtenir des implémentations effectives d'un schéma de signature fondé sur les codes correcteurs d'erreurs.

Comme l'indique son titre, cette thèse traite de la construction et de l'optimisation des cryptosystèmes basés sur des codes correcteurs d'erreurs et plus particulièrement de cinq nouveaux protocoles. On présente ici une version sécurisée du schéma de Stern dans un environnement à faibles ressources, une nouvelle construction du schéma de Kabatianski, Krouk et Smeets, un schéma de signature basé sur l'identité prouvé sûr dans le modèle de l'oracle aléatoire, un schéma de signature de cercle à seuil et enfin une réduction de la taille de clés du schéma de McEliece à l'aide de codes alternants quasi-cycliques.

En annexe, on présente un travail traitant des attaques algébriques de registre à décalage avec mémoire. On présente aussi brièvement une étude des codes cycliques sur des anneaux de matrices.

Mots clés : Cryptologie, codes correcteurs d'erreurs, authentification, signature, preuve de sécurité, attaque DPA, signature basée sur l'identité, signature de cercle.

Abstract

In this thesis, we are interested in the study of encryption systems as well as signature schemes whose security relies on difficult problems of error-correcting codes. These research activities have been motivated, a part of a theoretical point of view by creating : new signature schemes with special properties and a way of reducing the size of the key of the McEliece scheme, and on the other hand, a practical point of view to use structural properties to obtain effective implementations of a signature scheme which is based on error-correcting codes.

As its title indicates, this thesis deals with the construction and optimization of cryptosystems based on error-correcting codes and more particularly five new protocols. It presents a secure version of the Stern scheme in a low-resources environment, a new construction of the Kabatianski, Krouk and Smeets scheme, a signature scheme based on the identity proved secure in the random oracle model, a threshold ring signature scheme and a reduction of the size of the key of the McEliece scheme using quasi-cyclic alternant codes.

In the annex, this work deals with algebraic attacks against linear feedback shift register with memory. It also presents a brief study of cyclic codes on rings of matrices.

Keywords : Cryptology, error correcting codes, authentication, signature, security proof, attack DPA, identity-based signature, ring signature.