UNIVERSITÉ CHEIKH ANTA DIOP

Ecole Doctorale de Mathématiques et Informatique

FACULTE DES SCIENCES ET TECHNIQUES

FORMATION DOCTORALE : Codage, Cryptologie, Algèbre et Applications

Thèse de Doctorat Unique en Mathématiques

pour obtenir le grade de docteur délivré par

École doctorale Doctorale de Mathématiques et Informatique

Mention: Mathématiques et Modélisation

Option: Codage, Cryptologie, Algèbre et Applications

Année : 2017-2018 N° d'ordre :

CHIFFREMENT HYBRIDES ET AUTHENTIFICATION FORTE BASES SUR LES CODES CORRECTEURS D'ERREURS

présentée et soutenue publiquement par **EL Hadji Modou MBOUP**

Composition du jury

Président:		
	Mamadou SANGHARE	Professeur, UCAD
Rapporteurs:		
	Thierry BERGER	Professeur, Univ. de Limoges(France)
	Sylvain GUILLEY	Professeur Télécom-Paristech (France)
Examinateurs:		
	Oumar DIANKHA	Professeur, UCAD
	Mamadou L. NDIAYE	Professeur, UCAD
	Babacar A. NDAW	Docteur, STCC-SSI (Présidence)
Directeur:		
	Cheikh Thiécoumba GUEYE	Professeur, UCAD

THESE

Année académique 2017-2018

Remerciements

Préparer cette thèse, la rédiger et la soutenir, ont été des privilèges pour lesquels je souhaite exprimer ma gratitude et ma reconnaissance. Malgrés mes contraintes professionnelles, j'ai reussi à finaliser cette thése grace à ALLAH et aussi mon directeur de thèse Professeur **Cheikh Thiécoumba Gueye** sans lui rien n'aurait été possible, et qui m'a encadré et encouragé par ses conseils.

Mes sincéres remerciements vont à l'endroit de tous les membres de ma famille mon épouse Aida SECK et mes enfants, Yama NDAO MBOUP et Abdou MBOUP. Je remercie très chaleureusement les autres membres de ma famille mon pére Momath MBOUP, mes deux méres Ndiatté Niang et Gass Dieng, le Ministre Abdou MBOUP, Adama Sidibé et ces enfants sans oublié mon homonyme Modou Mboup, Malick Mboup, Babacar Mboup, Maissa MBOUP, Yama Ndao MBOUP (soeur), mes petits fréres Lamine Mboup, Ousseynou, Abdoulaye Seck, ainsi mes gendres Loly Mbaye, Kéba Mbaye, Babacar Seck ... qui m'ont soutenu et supporté durant ces années de thése, riches de rebondissements pour eux aussi. Aida, en particulier, me mettait souvent une pression énorme pour la finalisation de cette thèse.

- En premier lieu, je remercie vivement le Professeur **Mamadou Sangharé** pour l'honneur qu'il m'a fait en acceptant de présider mon jury de thèse.

- Je tiens à remercier très sincèrement, au fond de moi même, mon directeur de thèse le Professeur **Cheikh Thiécoumba Gueye**, en lui exprimant ma profonde gratitude pour l'excellente formation que j'ai reçue auprès de lui sur tous les plans (didactique, recherche, humain,..). Professeur **Cheikh Thiécoumba Gueye** m'a encadré au quotidien et a aussi été l'initiateur de cette thèse. Il m'a aussi aiguillé vers des voies de recherches et, tout au long de cette thèse, n'a jamais cessé de me conseiller et diriger pour que ce travail soit de la plus grande qualité.

- J'exprime ma profonde gratitude aux Professeurs **Thierry Berger** et **Sylvain GUILLEY** qui ont accepté d'être rapporteurs de cette thèse et examinateurs dans ce jury. Merci pour ces remarques pertinentes qui du point de vue fond et forme ont contribué à l' amélioration de ce document.

- Je tiens à remercier le Professeur **Oumar Diankha** d'avoir accepté d'être dans le jury en qualité d'examinateur. Merci d'avoir accueilli parfois, un petit curieux que je suis, dans votre équipe de recherche. Merci d'avoir confiance en moi et de me confier la charge de cours en Master TDSI et aussi les multiples recommandations faites à mon égard!

- Je tiens à remercier le Docteur **Mamadou L. NDIAYE** d'avoir accepté d'être dans le jury en qualité d'examinateur. Je suis marqué par votre pédagogie à travers les cours de VHDL.

- Mes sincères remerciements sans limites vont aussi à l'endroit du Docteur **Babacar A. NDAW** d'avoir accepté d'être dans le jury en qualité d'examinateur. De plus, il a participé à la correction de cette thése malgrès son emploi du temps trop chargé. Merci d'avoir corrigée cette thèse.

- Mes sincères remerciements à l'endroit du Professeur **Djiby SOW** de ses conseils et de m'avoir confié la charge de cours en TDSI.

- Mes gravitudes aussi à l'endroit du Professeur **Mamadou Barry** le chef département Mathématiques et Informatique de ses conseils et de m'avoir confié la charge de cours travaux dirigés (TD) en analyse et algébre. Ces scéances TD m'a permis de familiariser avec les étudiants et aussi d'améliorer mes bases en mathématiques.

- Au début de cette thèse, j'ai eu l'opportunité d'effectuer un séjour d'un mois à l'université d'Alméria-Espagne, sous la supervision du Professeur **Elamin Kaidi Lhachmi**, envers lequel va ma gratitude.

- J'en profite pour remercier mes autres co-auteurs sans lesquels cette thèse n'aurait pas vu le jour : Pierre-Louis Cayrel, Ousmane Ndiaye et Edoardo Persichetti .

  - A tous les membres du département de Mathématiques et Informatique ainsi qu'aux membres du Laboratoire d'Algèbre de Cryptographie de Géométrie Algébrique et Applications (LACGAA) pour la sympathie qu'ils m'ont témoignée durant ces années de thèse.

  - Aux autres membres plus que sympathiques de notre équipe de recherche Equipe de Recherche en Codes et Implémentations Sécurisées post-quantique (**ERCISpq**) dirigée M. Cheikh Thiécoumba Gueye: Anta Niane Gueye, Ousmane Ndiaye, Adiawa Haikréo, Mbouye Khady Diagne, Jean Klamti Belo, Brice Odilon Boidje, Gilbert Dione, Cherif Fall, Yakhya Diop. A l'équipe de la présidence dirigée par M. Babacar Alassane Ndaw: Omar Ndoye, Amapenda Gueye, Abdoul Aziz Kane ...

  - Les autres membres du labo LACGAA: Chérif DEME, Mouhamadou Lamine DIOUF, Abdoul Aziz CISS, Demba Sow , G. CAMARA, Régis Babindamana,Demba Sow , Youssef, Ahmed Khalifa, Amadou TALL.

- Mes remerciements à l'endroit :

– de mes amis d'enfance de tous les jours : Mor Seck, Kara Mboup, Assane Seye, Mami Seye, Moussa Gassama, Modou Ndiéguéne, Pape Mor Ndiogou, Matar Sarr, Petit Ndao, Menirou Diop, Sokhna Sarr, Khalifa Diop, Amath Seck, Bass Noble, Malick Sarr, Balla Fall, les autres sans oubliés personne ....!

– de mes étudiants, je veux nommer mes étudiants: TDSI, ESSA, EPT, UADB, ISM. Merci d'avoir pu supporter toutes mes explications kilométriques et théoriques, mes débordments et les pauses oubliées.

– Au personnel administratif du département: M. Famara Massaly, Dabo, Mme Mbaye, Mme Ba, Gnima et Kiné les nouvelles recrues pour leur gentillesse et leur disponibilité.

– Mes gratitudes au personnel de Gainde2000: Ibrahima N. Diagne, Souhaibou Diedhiou, Ibrahima Kamara, Louis Dieme, Daniel Sarr, Assane Sarr, Edouard Faye, Sandrine, Abdoulaye Sane, Souleymane Sidibe, Fatou M. Mane (Mme Mane), Ngone Ka, Isabel Diatta, Abdou Karim Diouf, Alassane Diene, Keita, Sabaly, Alyou Mbacké, Mor Talla Diop, Mohamed Diouf, Aissatou Dieng,...

DEDICACES

“ Madame MBOUP Aida SECK,
Mes enfants Yama et Abdou Mboup,
Mon pére Momath Mboup,
Mes mamans Ndiatté et Gasse,
Aussi Modou MBOUP mon homonyme
Enfin, j’ai une pensée émue pour mon
ami Amath BA DEME qui aurait
souhaité voir ce moment, je lui dédie
au fond de mon coeur cette thése. ”

Table des matières

Liste des Figures

Liste des Tableaux

Chapitre 1

Introduction

“ Mon plus grand souhait dans cette vie, c’est que mes enfants soient heureux ”

E. M. MBOUP

Sommaire

1.1 Généralité

La communication à travers un canal de transmission (réseau, satellite), pose souvent un problème de fiabilité, de confidentialité, d'intégrité, d'authencité, de disponibilité ou parfois de non-répudiation du message transmis. A cet effet, nous assistons à la mise en place des disciplines comme la théorie des codes et la cryptographie pour assurer la sécurité de ces communications. Par conséquence, la cryptographie est présente dans la vie quotidienne, comme par exemple les cartes de crédit, le paiement internet, elle permet aussi de chiffrer des informations secrètes. Ce chiffrement se fait avec des systèmes de chiffrement appelé Cryptosystème. Nous distinguons deux cryptosystèmes à savoir le cryptosytème à clé public (schéma asymétrique) et celui à clé sécrete (schéma symétrique).
La quasi totalité des cryptosysèmes à clé public sont basés sur des calculs complexes de la théorie des nombres. Nous pouvons citer le cryptosystème de RSA (Rivest, Adi Shamir), les cryptosystèmes basés sur les courbes elliptiques... En 1994, P.W. Shor dans [32] a montré que les ordinateurs quantiques peuvent casser la plupart des cryptosystèmes classiques ceux basés sur le problème de la factorisation des nombres entiers ou sur le problème du logarithme discret. Il est donc crucial de développer des cryptosystèmes résistants aux attaques informatiques quantiques. C'est dans cette perspective que la cryptographie basée sur les codes correcteurs d'erreurs est considérée comme un schéma très prometteur pour la cryptographie post-quantique. Ces schémas sont caractérisés par sa rapidité, ne nécessitent pas de matériel spécial et aussi spécifiquement de co-processeur cryptographique. les crytosystèmes basés sur les codes correcteurs d'erreurs possédent aussi comme avantages :

- la résistance face aux ordinateurs quantiques
- la rapidité et facilité de l'implémentation car basé sur des calculs matriciels.
- la NP-completude des problèmes.

Par ailleurs, il faut noter aussi l'existence des cryptosystèmes hybrides. Le cryptosystème hybride, appelé KEM/DEM avec KEM (KEY ENCAPSULATION MECHANISM) et DEM (DATA ENCAPSULATION DATA), fut inventé par Cramer et Shoup dans [10]. Il utilise les systèmes de chiffrements asymétrique, symétrique et aussi les fonctions de hachage. Ces schémas hybrides sont plus avantageux dans la mesure où ils utilisent les avantages des schémas de chiffrement qui le composent.
Ainsi, dans [8] les auteurs ont démontré la cryptographie hybride basé sur les codes utilisant le PKCS de Niederreiter comme chiffrement asymétrique et AES comme schéma symétrique est beaucoup plus rapide que l'hybride de RSA.

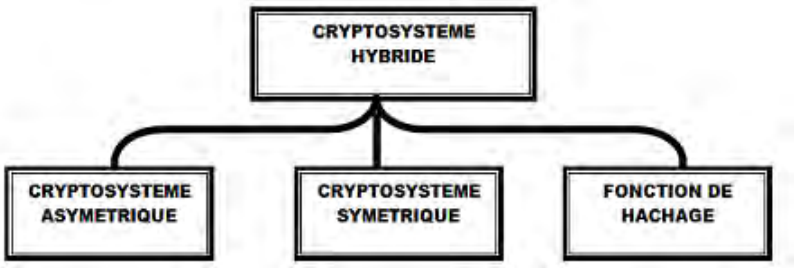


Figure 1.1 – Composantes d'un schéma de chiffrement hybride

1.2 Contributions

Le Cryptosystème RSA, inventé en 1977, est un exemple de Cryptosystème à clé publique. En 1978 le Cryptosystème de McEliece [20] a été inventé. Ce Cryptosystème utilise les codes correcteurs d'erreurs en particulier les codes de GOPPA.

C'est ainsi que le Cryptosystème de Sidel'nikov [33], basé sur les codes de Reed Muller, a été inventé en 1994. En 2007, ce Cryptosystème a été cassé par Lorentz Minder et Amin Shokrollahi [21] en utilisant deux propriétés des codes de Reed Muller (la filtration 3.3.13 et le mot de plus petit poids 3.3.14).

Ainsi dans la première partie de notre thèse nous nous sommes intéressés sur l'amélioration du cryptosystème de Sidel'nikov [17] afin de réparer la cryptanalyse de L. Minder et A. Shokrollahi. A ce sens, nous avons proposé une nouvelle variante de ce Cryptosystème avec une modification du code Reed Muller. Le but de cette modification des codes Reed Muller est de rendre quasiment imposible l'application de ces deux propriétés 3.3.14 et 3.3.13 du code de Reed Muller. Ce qui rend impraticable l'attaque présentée par L. Minder et A. Shokrollahi.
Cependant, cette version améliorée de Sidel'nikov a été cassée en 2015 par A.Otami et al. dans [24].

De plus, les cryptosystèmes basés sur les codes correcteurs d'erreurs présentent un inconvéniant majeur à savoir la grande taille des clés. C'est pour cela, on assiste à l'invention des schémas de chiffrement Hybride. Les fondateurs de ces schémas sont R. Cramer et V. Shoup dans [10] en 2004. En effet, cette invention nous a permis dans la deuxème partie de notre thése d'orienter notre recherche dans les schémas de chiffrment Hybride. Nous nous sommes intéressés sur l'amélioration d'un schéma Hybride basé sur le PKCS Niederreiter étudié dans [25]. Dans ces travaux, nous avons réparé ce schéma hybride, afin qu'il resiste à l'attaque de Bernstein et al. [5] en utilisant une permutation, par la suite im-

plémenté ce schéma en C. Cette implémentation est faite dans un ordinateur de caractèristique CPU 1.80 GHz, Intel core i3 et de RAM 4 GB avec un système d'exploitation Debian/Linux 3.5.2 dans gcc 4.7. Nous avons prouvé dans [8] que le chiffrement du schéma Hybride basé sur Niederreiter est beaucoup plus rapide que celui basé sur l'hybride de RSA.
Enfin, nous avons implémenté un protocole d'authentification basé sur les codes en utilisant le principe du KEM/DEM. Dans ce protocole, nous avons utilisé la fonction de hachage SHA3 pour construire un secret partagé, le chiffrement de Niederreiter pour le chiffrement du challenge et le chiffremnt symétrique AES pour chiffrer le secret partagé.

1.3 Organisation de la thése

Les travaux présentés dans cette thèse s'articulent autour de trois axes principaux :

- Un rappel sur les outils de mathématiques, cryptographie et théorie de code utilisés dans cette thése.
- Une présentation de la modification des codes de Reed Muller dans le but de réparer la propriété filtration de ces codes. Cette modification nous permet de proposer une version améliorée du cryptosytème de Sidel'nikov.
- Une amélioration et implémentation du schéma hybride basé sur les codes correcteurs d'erreurs en particulier le PKCS de Niederreiter.
- De plus, une présentation d'un protocole d'authentification KEM/DEM basé sur les codes correcteurs d'erreurs.

La thése est organisée en deux parties. Dans une première partie, intitulée Préliminaire, nous ferons un rappel sur l'ensemble des outils scientifiques (mathématiques, cryptographie, théorie des codes,..) utilisés dans cette thése. Elle est composée de quatre chapitres (chapitre 2, chapitre 3, chapitre 4 et chapitre 5). Dans le chapitre 2, nous ferons un rappel sur les outils mathématiques. Le troisième et quatrième chapitre porteront respectivement sur la présentation des notions de base de la théorie des codes et de la cryptographie. Au niveau du cinquième chapitre, nous présenterons certains systèmes de chiffrement basés sur les codes correcteurs. Dans ce chapitre, nous allons décrire les différents algorithmes ainsi que les avantages et inconvénients de ces Cryptosystèmes.Pour bien comprendre les attaques sur les Cryptosystèmes, nous allons faire une étude détaillée sur les attaques au niveau de ce chapitre. En ce sens, nous donnerons des explications détaillées sur les attaques par décodage et attaques structurelles.

La deuxième partie, intitulée Resultats, décrit les trauvaux scientifiques réalisés au cours de la recherche. Elle comprend trois chapitres à savoir le chapitre 6, chapitre 7 et chapitre 8. Dans le chapitre 6, nous ferons une description détaillée du nouveau cryptosystème défini dans [17]. Notre objectif dans ce chapitre est de faire une présentation du code de Reed Muller modifié, du cryptosystème basé sur ce nouveau code et enfin des preuves de sécurité qui justifient la résistance de ce cryptosystème. De plus, le chapitre 7 portera sur les schémas de chiffrement hybrides. Nous présenterons en détails une amélioration et implémentation du schéma hybride basé sur les codes en particulier le PKCS de Niederreiter [8]. Enfin, au niveau du chapitre 8, nous présenterons une implémentation d'un protocole d'authentification basé sur les codes.

1.4 Resumé des contributions

Les principales contributions de cette thèse sont les suivantes :

- **Un nouveau schéma de chiffrement basé sur les codes Reed Muller modifié**:Ce schéma est une amèlioration de Sidel'nikov afin que ce dernier resiste contre l'attaque présenté dans [21]. Ce papier a été publié dans *International Journal of Security and Its Applications, vol. 7, no. 3, (2013), pp. 55-64.* [17]

- **Une implantation logicielle efficace d'un schéma hybride basé sur les codes**: Dans ce travail, nous avons amélioré le schéma hybride basé sur Niederreiter en utilisant une permutation et puis implémenté ce schéma en C. Cet article a été publié dans Said El Hadji et al. (Eds) C2SI 2017, Lecture Notes in Computers Sciences 10194 pp 254–264, springer [8].

- **Une implantation d'un protocole d'authentification basé sur les codes**: Dans ce travail, nous avons utilisé le principe du KEM-DEM pour mettre en place un protocole d'authentification forte qui sera utilisé dans une application web.

Part I

Préliminaires

Chapitre 2

Rappel sur les outils mathématiques

“ La vie n’est bonne qu’à étudier et à enseigner les mathématiques. ”

De Blaise Pascal

Sommaire

2.1 Introduction

Dans ce chapitre nous allons parler des notions de mathématiques que nous utiliserons dans la suite. Nous ferons une présentation succincte sur les anneaux, corps finis, idéaux, espaces vectoriels qui sont les notions de base de la théorie des codes et de la cryptologie. Etant donné les codes Reed muller sont décrits sous forme de fonctions booléennes, nous ferons une description des fonctions booléennes dans ce chapitre.
Enfin de chappitre, nous ferons une présentation sur la probabilité qui sera utilisée dans le calcul du niveau de sécurité des cryptosystèmes.

2.2 Anneau

Définition 2.2.1 *Un anneau est une structure* $(\mathrm{A}, +, *)$ *telle que:*

(i) $(\mathrm{A}, +)$ *est groupe commutatif*

(ii) $\forall\ (a,b,c) \in \mathrm{A}\times\mathrm{A}\times\mathrm{A}$, $a*(b*c) = (a*b)*c$ *(**associativité de** $*$)*

(iii) $\forall (a,b,c) \in \mathrm{A}\times\mathrm{A}\times\mathrm{A}$, $(a+b)*c = (a*c)+(b*c)$ *(**ditributivité à droite**)*

(iv) $\forall (a,b,c) \in \mathrm{A}\times\mathrm{A}\times\mathrm{A}$, $a*(b+c) = (a*b)+(a*c)$ *(**ditributivité à gauche**)*

(v) *Il existe un élément noté* 1 *dans* A *tel que* $\forall\, a \in \mathrm{A}$, $1*a = a*1 = a$ *(**élément neutre**)*

On dit que $(\mathrm{A}, +, *)$ *est un anneau commutatif si de plus* $\forall (a,b) \in \mathrm{A}\times\mathrm{A}$, $a*b = b*a$ *(**commutativité de la loi** $*$).*

Proposition 2.2.2 *(**Propriétés arithmétiques sur les anneaux**)*
Soit $(\mathrm{A}, +, \cdot)$ *un anneau. Pour tout* $x, y \in \mathrm{A}$*, on a:*

1. $0 \cdot x = 0$
2. $(-1)\cdot x = -x$
3. $(-1)\cdot(-1) = 1$
4. $(-x)\cdot y = -x\cdot y$

Preuve:

1. $0\cdot x + x = 0\cdot x + 1\cdot x = (0+1)\cdot x = 1\cdot x = x$
   Donc $0\cdot x = 0$
2. $0 = 0\cdot x = (1-1)\cdot x = 1\cdot x - 1\cdot x = x - 1\cdot x$
   Donc $-x = -1\cdot x$.

3. *On multiplie par* -1 *l'égalité* $(-1)+1=0$.
   Cela donne $(-1)\cdot(-1)+(-1)\cdot(1)=0$ *et donc* $(-1)\cdot(-1)+(-1)=0$
   Ce qui prouve que $(-1)\cdot(-1)=1$.

4. $x\cdot y+(-x)\cdot y=(x+(-x))\cdot y=(x-x)\cdot y=0\cdot y=0$
   Donc l'opposé de $x\cdot y$ *qui est, par convention d'écriture,* $-x\cdot y$, *est égal à* $(-x)\cdot y$.

Proposition 2.2.3 *(**Formule du binome**)*
Soit $(\mathrm{A},+,*)$ *un anneau commutatif, a et b deux éléments de* A *et n un entier positif. Alors,*

$$(a+b)^n=\sum_{i=0}^{n}\mathrm{C}_n^i a^i b^{n-i}$$

où $\mathrm{C}_n^i=\frac{n!}{i!(n-i)!}$.
Preuve:
La preuve de cette proposition se fera par recurrence.

- *Si* $n=1$ *la formule est triviale,*
- *Supposons la formule est vraie à l'ordre* $n-1$*;*
  Démontrons que la formule est vraie à l'ordre n:

$$(a+b)^n=(a+b)(a+b)^{n-1}=(a+b)\sum_{i=0}^{n-1}\mathrm{C}_{n-1}^i a^i b^{n-1-i}$$

$$=\sum_{i=0}^{n-1}\mathrm{C}_{n-1}^i a^{i+1} b^{n-1-i}+\sum_{i=0}^{n-1}\mathrm{C}_{n-1}^i a^i b^{n-i}$$

or $\sum_{i=0}^{n-1}\mathrm{C}_{n-1}^i x^{i+1}y^{n-1-i}=\sum_{i=0}^{n}\mathrm{C}_{n-1}^{i-1}x^i y^{n-i}$, *en posant* $\mathrm{C}_n^k=0$ *si* $k<0$
Alors,

$$(a+b)^n=\sum_{i=0}^{n}(\mathrm{C}_{n-1}^{i-1}+\mathrm{C}_{n-1}^i)a^i b^{n-i}$$

Mais comme $\mathrm{C}_{n-1}^{i-1}+\mathrm{C}_{n-1}^i=\mathrm{C}_n^i$, *d'où la formule est démontrée.*

Définition 2.2.4 *Soit* B *un sous-ensemble non vide d'un anneau* A. *On dit que* B *est une partie stable de* A *si* B *est stable pour les deux lois internes de* A.
On dit qu'une partie stable B *de* A *est un sous anneau de* A *si* B *est un anneau pour la restriction à* B *des deux lois internes de* A, *et si son élément unité est le meme que celui de A.*

2.3 Idéaux

Définition 2.3.1 *Soit* $(\mathrm{A},+,\cdot)$ *un anneau et* I *un sous ensemble de* A. I *est un **idéal à gauche** (resp. **à droite**) de* A *si et seulement si:*

- I *est un sous groupe abélien de* A *pour la loi* $+$.

- *Pour tout élément a de* A *et x de* I, $a \cdot x$ *(resp.* $x \cdot a$*) est un élément de* I.

Définition 2.3.2 *Soit* A *un anneau et* I *un sous ensemble de* A. I *est un* ***idéal bilatère*** *de* A *si et seulement si* I *est à la fois un idéal à gauche et un idéal à droite de* A. *On utilisera de manière générale le mot idéal pour idéal bilatère.*

Définition 2.3.3 *Soit* A *un anneau et* I *un idéal de* A. I *est un* ***idéal principal*** *de* A *si et seulement si* I *est engendré par un seul élément a de* A.
Autrement dit $\mathrm{I} = \{x \cdot a, a \in \mathrm{A}\}$.
On notera dans ce cas (a), *l'idéal engendré par l'élément a de* A.

Définition 2.3.4 *L'idée* (0) *engendré par l'élément* 0 *d'un anneau* A *sera appelé l'****idéal nul*** *de A.*

Définition 2.3.5 *Un anneau est dit* ***principal*** *s'il est intégre et que tous ses idéaux sont principaux.*

Définition 2.3.6 *Un idéal* I *dans un anneau A est dit* ***strict*** *ou* ***propre*** *dans* A *s'il n'est pas égal à l'anneau tout entier.*

Définition 2.3.7 *Un idéal* I *dans un anneau A est dit* ***strict*** *ou* ***propre*** *dans* A *s'il n'est pas égal à l'anneau tout entier.*

Définition 2.3.8 *Un idéal est* ***maximal*** *s'il est* ***strict*** *et s'il n'est contenu dans aucun idéal autre que l'anneau tout entier.*

Proposition 2.3.9 *Si un idéal d'un anneau* A *contient l'élément unité de l'anneau alors cet idéal est égal à l'anneau tout entier.*
Preuve:
Supposons que l'idéal I *de l'anneau* A *contienne l'élément* 1 *de* A. *Alors pour tout* $a \in \mathrm{A}$, $a = a \cdot 1$ *est, par définition d'un idéal, un éléemnt de* I.
Donc $\mathrm{A} \subseteq \mathrm{I}$ *, d'où* $\mathrm{A} = \mathrm{I}$.

2.4 Corps

Définition 2.4.1 *Soit* IK *un ensemble et soient* $+$ *et* $\cdot$ *deux lois internes sur* IK. *Le triplet* $(\mathrm{IK}, +, \cdot)$ *posséde une structure de corps si:*

- $(\mathrm{IK}, +, \cdot)$ *a une structure d'anneau commutatif unitaire.*
- $(\mathrm{IK} \setminus \{0\}, \cdot)$ *a une structure de groupe.*

Si la loi $\cdot$ de l'anneau IK *est commutative. On dit que* IK *est un corps commutatif.*
Si IK *a un nombre fini d'éléments, on dit que* IK *est un corps fini.*

Exemple 2.4.2 Q *(ensemble des nombres rationnels),* IR *(ensemble des nombres réels) et* C *(ensemble des nombres complexes) ont des structures de corps pour leur addition et multiplication respectives.*

Theoreme 2.4.3 *L'anneau quotient* $(\mathrm{Z}/p\mathrm{Z}, +, *)$ *(où p est un nombre premier et* Z *l'ensemble des nombres entiers relatifs), est un corps.*

Définition 2.4.4 *(Caractérisation d'un anneau et d'un corps)*
Soit IK *un corps dont 1 est l'élément neutre pour la multiplication. Le plus petit entier* $n \in \mathrm{IN}^*$*, s'il existe tel que* $n \cdot 1 = \left(\underbrace{1+1+\ldots+1}_{n\,fois} \right) = 0$*, est appelé caractéristique de* IK.
Si un tel entier n'existe pas, on dit que IK *est de caractéristique 0.*

Exemple 2.4.5 *(Exemple de corps)*

- Q *(ensemble des nombres rationnels),* IR *(ensemble des nombres réels) et* C *(ensemble des nombres complexes) sont de caractéristique* 0.

- $\mathrm{Z}/p\mathrm{Z}$ *est un corps de caractéristique* p*. Ce corps est noté* $\mathrm{GF}(p)$ *pour Galois Field ou bien* F_p.

Theoreme 2.4.6 *Soit* IK *un corps. Alors sa caractéristique est soit* 0 *ou un nombre premier.*
Preuve: *Soit* p *la caractéristique du corps* IK.
Soient p_1 *et* p_2 *deux entiers positifs inférieurs à* p *tel que* $p = p_1.p_2$.
Alors on a $p.1 = (p_1.p_2) = (p_1.1).(p_2.1)$ *puis que* IK *est un anneau.*
De plus p caractéristique de IK *entraine que* $p.1 = 0$,
IK *est intégre (car* IK *est un corps), on en déduit que* $p_1.1 = 0$ *ou* $p_2.1 = 0$.
Les deux facteurs p_1 *et* p_2 *étant inférieurs à* p*, alors les seules possibilités sont:*

- *Soit l'un des facteurs est égal à* 0*, et dans ce cas on a* p *le caractéristique est* 0.

- *Soit l'un de ces facteurs est égal à* 1 *et l'autre à* p*. Et dans ce cas* p *sera un nombre premier.*

D'où la démonstration du théorème.

2.5 Espace Vectoriel

Définition 2.5.1 *(**Espace vectoriel**) On appelle espace vectoriel sur un corps* IK *tout ensemble* E *muni de deux lois* $+$ *et* $\cdot$ *vérifiant:*

1. $(E,+)$ *est un groupe commutatif (c-à-d,* $+$ *est une loi interne qui est associative, possédant un élément neutre, et tel que tout élément a un opposé),*

2. *pour tout* (λ,μ) *dans* IK^2*, pour tout* x *dans* E, $\lambda\cdot(\mu\cdot x)=(\lambda\cdot\mu)\cdot x$ *et* $1\cdot x=x$*,*

3. $\cdot$ *est distributive par rapport à* $+$ *dans* E *et par rapport à* $+$ *dans* IK.

Exemples 2.5.2 *(Exemples d'espaces vectoriels)*

- IR^n *et* C^n *sont des espaces vectoriels quelque soit l'entier* n*. Ce sont des espaces vectoriels de dimension finie.*

- IR[X]*, l'ensemble des polynomes à coefficients réels, e st un* $IR-espace$*.*

Définition 2.5.3 *(**Sous espace vectoriel**) Soit* E *un espace vectoriel. On dit que* F *est un sous-espace vectoriel de* E*, si c'est un espace vectoriel et que* $F\subset E$*.*

Exemple 2.5.4 IR^2 *est un sous-espace vectoriel de* IR^3*.*
Pour montrer qu'un ensemble est un espace vectoriel, il suffit souvent de montrer que c'est un sous-vectoriel d'un espace vectoriel connu. Pour cela, on utilise le théorème suivant.

Theoreme 2.5.5 *(**Caractérisation des sous-espaces**) Soient* E *un espace vectoriel et* F *un ensemble tel que:*

(i) $F\subset E$,

(ii) $0_E\in F$,

(iii) $\forall(\alpha,\beta)\in IK^2,\forall(x,y)\in F^2,\alpha x+\beta x\in F$.

Alors F *est un espace vectoriel, c'est un sous-espace vectoriel de* E*.*

Proposition 2.5.6 *L'intersection quelconque de sous-espaces vectoriels est un sous-espace vectoriel.*

Remarque 2.5.7 *Ceci est généralement faux pour l'union !!!!!!!*

2.5.1 Familles libres, génératrices, bases

Dans toute la suite sauf mention contraire, E désigne un IK espace vectoriel quelconque. Soit I un ensemble quelconque d'indices.

Définition 2.5.8 *Soit* $(a_i)_{i\in I}$ *une famille d'éléments de* E. *On dit que la famille* $(a_i)_{i\in I}$ *est:*

- ***libre*** *si pour toute famille finie de scalaires* $(\lambda_i)_{i\in J}$ *(ici* $J \subset I$*), on a* $\Sigma_{j\in J}\lambda_j a_j = 0$ *implique* $\forall\, j \in J, \lambda_j = 0$. *Si elle n'est pas libre, on dit que la famille est* ***liée****.*
- ***génératrice*** *si* $\forall\, x \in E$, *il existe une famille finie de scalaires* $(\lambda_j)_{j\in J}$ *(ici* $J \subset I$*), telle que* $x = \Sigma_{j\in J}\lambda_j a_j$.
- ***une base de E*** *si elle est à la fois libre et génératrice.*

Proposition 2.5.9 *La famille* $(a_i)_{i\in I}$ *est une base de* E *si et seulement si tout élément de* E *s'écrit de maniètre unique comme une combinaison linèaire finie d'éléments de* $(a_i)_{i\in I}$.

Proposition 2.5.10 *Une famille de vecteurs est liée si et seulement si un des vecteurs s'exprime comme une combinaison linèaire des autres.*

Theoreme 2.5.11 *Soit* $\mathfrak{F}$ *une famille de vecteurs, les points suivants sont équivalents:*

1. $\mathfrak{F}$ *est une base (i.e. libre et génératrice).*
2. $\mathfrak{F}$ *est libre et maximale (i.e. toute sur-famille strict. n'est plus libre).*
3. $\mathfrak{F}$ *est générateur et minimum (i.e. toute sous-famille strict. n'est plus génératrice).*

2.5.2 Espace vectoriel en dimension finie

Theoreme 2.5.12 *(et définition) Si l'espace vectoriel admet une base et que cette base comporte un nombre fini d'éléments n, alors toute base de* E *admet le meme nombre d'éléments. On dit alors que* E *est de dimension finie et on le note* $dim(E) = n$. *Dans le cas contraire, on dit que* E *est de dimension infinie.*

Theoreme 2.5.13 *(Base incompléte et fabrication de bases en dimension finie)*
Toute famille libre peut etre complétée en une base.
De toute famille génératrice, on peut extraire une base.

Theoreme 2.5.14 *(Caractérisation des bases en dimension finie)*
Soit E *un espace vectoriel de dimension finie n.*

- *Toute famille génératrice est de cardinal au moins n.*

- *Toute famille libre est de cardinal au plus n.*
- *Toute famille libre de cardinal n est une base de* E.
- *Toute famille génératrice de cardinal n est une base de* E

Theoreme 2.5.15 *Soit* E *un espace vectoriel de dimension finie n. Soit* F *un sous-espace vectoriel de E. Si* $dim(F) = n$, *alors* E = F.

2.5.3 Sommes directes

Définition 2.5.16 *(**Somme de plusieurs sous-espaces**) Soit* E *un espace vectoriel.*
Soit $(V_1, V_2, ..., V_n)$ *une famille de sous-espaces vectoriels de* E. *La somme* $\sum V_i$ *(ou* $V_1 + V_2 + ... + V_n$*) est l'ensemble des vecteurs x de* E *s'écrivant de la forme* $x = \sum x_i$ *où* $(x_1, ..., x_n) \in (V_1, ..., V_n$.

Proposition 2.5.17 $V_1 + ... + V_n$ *est un sous-espace vectoriel de* E. *C'est le plus petit espace vectoriel contenant* $V_1, ..., V_n$.

Définition 2.5.18 *(**Somme directe**) On dit que la somme* $V_1 + ... + V_n$ *est directe si* $\forall (x_1, ..., x_n) \in (V_1 \times ... \times V_n), \sum x_i = 0$ *implique* $\forall 1 \leq i \leq n, x_i = 0$. *Dans ce cas on écrit:*

$$V_1 + ... + V_n = V_1 \bigoplus ... \bigoplus V_n = \bigoplus_{1 \leq i \leq n} V_i$$

Proposition 2.5.19 .

- *Si les vecteurs de* E, $v_1, ..., v_n$ *forment une famille libre, alors la somme des sous-espaces* $vect(v_1), ..., vect(v_n)$ *est directe.*
- *Si* $(V_1, ..., V_n)$ *est une famille de sous-espace vectoriels de* E *dont la somme est directe et si* $(x_1, ..., x_n) \in (V_1, ..., V_n)$ *alors la famille de vecteurs de* E $(x_1, ..., x_n)$ *est libre.*
- *Si* E *et* F *sont des espaces vectoriels alors* $E \times \{0_F\}$ *et* $\{0_E\} \times F$ *sont des sous-espaces vectoriels de* $E \times F$ *et* $E \times F = (E \times \{0_F\}) \bigoplus (\{0_E\} \times F)$.

Corollaire 2.5.20 *Si* $V_1, ..., V_p$ *est une famille de sous-espaces vectoriels de* E *telle que* $E = V_1 \bigoplus ... \bigoplus V_p$, *alors* $dim(E) = \sum dim(V_i)$.
Ainsi, la somme de trois droites vectorielles du plan n'est jamais directe.

Corollaire 2.5.21 *Soient* A *et* B *deux sous-espaces vectoriels de* E. *On a alors:*

$$dim(A + B) = dim(A) + dim(B) - dim(A \bigcap B)$$

Proposition 2.5.22 *Soit* E *un espace vectoriel de dimension finie. Soit* $(V_1, ..., V_p)$ *une famille de sous-vectoriels de* E.

- *Si la somme* $V_1 + ... + V_p$ *est directe et si* $dim(E) = \sum dim(V_i)$ *alors* $E = V_1 \bigoplus ... \bigoplus V_p$.
- *Si* $E = V_1 + ... + V_p$ *et* $dim(E) = \sum dim(V_i)$ *alors la somme* $V_1 + ... + V_p$ *est directe.*

2.6 Les fonctions booléennes

Dans cette partie nous abordons les fonctions booléennes à m variables. L'étude de ces fonctions se fera sur les éléments du corps IF_2.
Considérons le corps IF_2^m où m est un entier qui sera le nombre de bit que l'on doit manipuler durant tous nos travaux.
Avant d'étudier les fonctions booléennes, il est intéressant d'évoquer quels que concepts de base qui caractèrisent les fonctions booléennes.

Définition 2.6.1 *(**Support**) Soit un vecteur $b = (b_1, ..., b_m)$ de IF_2^m. Le support de b, noté $supp(b)$, est l'ensemble des positions des bits non nuls de b. De manière plus formelle on a:*

$$supp(b) = \{i, b_i = 1\}$$

Définition 2.6.2 *(**Poids de Hamming**) Soit un vecteur $b = (b_1, ..., b_m)$ de IF_2^m. Le poids de Hamming de b est égal à son nombre de composantes non nulles, on le notera $|b|$. C'est-à-dire*

$$|b| = |supp(b)|$$

où $||$ est le cardinal du support de b.

Définition 2.6.3 *(**Ordre lexicographique inverse**)*
L'ordre lexicographique inverse sur les élements de IF_2^m est tel que $x \ll y$ (où $\ll$ est l'ordre lexicographique) si et seulement s'il existe:
$i_0 \in [1, m]$ tel que $\forall\, i \in]i_0, m]$ $x_i = y_i$ et $x_{i_0} < y_{i_0}$.
Cet ordre présente de nombreuses propriétés qui vont le rendre très utile.
En choisissant l'ordre lexicographique pour représenter les éléments de $\mathrm{IF}_2^m = \{\alpha_1, ..., \alpha_{2^m}\}$, où α_i est l'écriture en binaire sur m bits de l'entier $i-1$.
Les monomes de degrés 1 sont $\mathrm{X}_1, ..., \mathrm{X}_m$ et le vecteur associé au monome X_i une concaténation de 2^{m-i} blocs, chaque bloc formé de 2^{i-1} 0 suivi 2^{i-1} de 1.
Par exemple pour

$$\mathrm{X}_i = \left(\overbrace{\underbrace{0,...,0}_{2^{i-1}}\underbrace{1,...1}_{2^{i-1}},...,\underbrace{0,...,0}_{2^{i-1}}\underbrace{1,...,1}_{2^{i-1}}}^{2^{m-i}} \right) \qquad (1.7.1)$$

Pour les vecteurs associés aux monomes de degré supérieurs, il faut calculer le produit des vecteurs associés à ses termes.

Exemple 2.6.4

Pour m=3, on a $f = 1 + X_1 + X_2 + X_3 + X_1X_2 + X_1X_2X_3$

$$\begin{aligned} 1 &= 11111111 \\ X_1 &= 01010101 \\ X_2 &= 00110011 \\ X_3 &= 00001111 \\ X_1X_2 &= 00010001 \\ X_1X_2X_3 &= 00000001 \end{aligned}$$

Définition 2.6.5 *(**fonction booléenne**)*
Une fonction booléenne de m variables est une fonction définie par:

$$f : \mathrm{IF}_2^m \to \mathrm{IF}_2$$

où $\mathrm{IF}_2 = \{0, 1\}$.

Exemple 2.6.6 *Soit f une fonction booléenne définie par:*

$$\begin{array}{rccc} f: & \mathrm{IF}_2^3 & \to & \mathrm{IF}_2 \\ & (x_1, x_2, x_3) & \mapsto & x_1x_3 + x_2x_3 \end{array}$$

Différents opérateurs peuvent etre appliqués aux fonctions booléennes. Nous allons énumérer quelques opérateurs:
Soient f et g deux fonctions booléennes.
La multiplication:

$$(fg)(x) = 1 \qquad ssi \qquad f(x) = 1 \qquad et \qquad g(x) = 1$$

L'addition modulo 2 ou xor:
$(f + g)(x) = 1$ *ssi* $f(x) = 1$ *ou* $g(x) = 1$ *mais pas tous égaux à la fois à* 1.
$(f + g)(x) = [f(x) + g(x)] mod 2$

Définition 2.6.7 *(**Support**) Soient f et $b = (b_1, ..., b_m)$ de* IF_2^m.
Le support de f, noté sup(f), est l'ensemble des antécédents dont leurs valeurs par f sont non nuls de b.
De manière plus formelle on a:

$$sup(f) = \{b, f(b) \neq 0\}$$

x_1	x_2	x_3	f
0	*0*	*0*	*0*
0	*0*	*1*	*0*
0	*1*	*1*	*1*
1	*0*	*0*	*0*
1	*0*	*1*	*1*
1	*1*	*0*	*0*
1	*1*	*1*	*1*

Définition 2.6.8 *(**Table de vérité**) Soit f une fonction booléenne à m variables. On appelle table de vérité ou vecteur des valeurs la liste de tous les éléments de* IF_2^m *avec les valeurs prises par la fonction en chacun d'eux.*

Fonctions booléennes $\mathfrak{F}_m$ et espace vectoriel $\mathrm{IF}_2^m (n = 2^m)$:

Nous allons montrer dans cette partie la correspondance entre l'espace des fonctions booléennes $\mathfrak{F}_m$ et l'espace vectoriel IF_2^m.

Nous avons vu que toute application $f : \mathrm{IF}_2^m \to \mathrm{IF}_2$ est polynomiale.

En effet, si $y = (y_1, ..., y_m) \in \mathrm{IF}_2^m$. Soit le polynome:

$$\begin{array}{rccc} \delta_y : & \mathrm{IF}_2^m & \to & \mathrm{IF}_2 \\ & x & \mapsto & \prod_{i-1}^m (1 + x_i + y_i) \end{array}$$

qui vérifie $\delta_y = \begin{cases} 1 \ si \ x = y \\ 0 \ si \ non \end{cases}$

La fonction $f : \mathrm{IF}_2^m \to \mathrm{IF}_2$ s'écrit:

$$f(x) = \sum_{\alpha \in \mathrm{IF}_2^m} f(\alpha)\delta_\alpha(x) \qquad (1.7.2)$$

Ce qui montre que la famille $(\delta_x)_{x \in \mathrm{IF}_2^m}$ est donc génératrice.

On montre que cette famille est libre.

Ainsi on a: $dim(\mathfrak{F}_m) = 2^m$, d'où le nombre de vecteur de cette base.

D'où la famille des $(\delta_x)_{x \in \mathrm{IF}_2^m}$ est une base de $\mathfrak{F}_m$.

Remarque 2.6.9 *Ainsi on peut dire d'après* (1.7.2) *que toute fonction de $\mathfrak{F}_m$ est une fonction polynomiale.*

Définition 2.6.10 *La forme algébrique normale (FAN) d'une fonction booléenne f à m variables est donnée par l'unique vecteur de 2^m bits ($f_u, u \in \mathrm{IF}_2^m$) tel que*

$$f(x) = \sum_{u \in \mathrm{IF}_2^m} f_u x^u \qquad avec \qquad x^u = x_1^{u_1} ... x_m^{u_m}$$

On appelle degré de f, noté $deg(f)$, le degré du polynome f.

Définition 2.6.11 *(**écriture vectorielle d'une fonction**)*
Soit l'ensemble $\mathfrak{F}_m$ *des fonctions booléennes de* IF_2^m *dans* IF_2.
L'écriture vectorielle des fonctions booléennes de m variables est l'application bijective:

$$\begin{array}{rccc} ev_\alpha : & \mathfrak{F}_m & \rightarrow & \mathrm{IF}_2^m \\ & f & \mapsto & (f(\alpha_1), ..., f(\alpha_n)) \end{array} \qquad (avec \quad n = 2^m)$$

Exemple 2.6.12 *Considèrons le cas* $m = 2$.
Alors d'après ce qui précéde la base de l'espace vectoriel $\mathrm{IF}_2^n (n = 2^m)$ *existe et est:*

$$\beta = \{(0,0), (1,0), (0,1), (1,1)\}$$

Soit la fonction booléenne, construite à partir de la base de l'espace vectoriel et xor *des éléments des vecteurs de la base, définie de la manière suivante:*

$$\begin{array}{rccc} f : & \mathrm{IF}_2^m & \rightarrow & \mathrm{IF}_2 \\ & (0,0) & \mapsto & 0 \\ & (1,0) & \mapsto & 1 \\ & (0,1) & \mapsto & 1 \\ & (1,1) & \mapsto & 0 \end{array}$$

Alors l'écriture vectorielle de f *est:* $ev(f) = (0,1,1,0)$.

Remarque 2.6.13 *L'écriture vectorielle de* 1 *est :* $ev(1) = (1,1,...,1)$.

2.7 Probabilité

Dans cette partie, nous évoquerons la notion de probabilité que nous utilisons dans la suite de cette thése pour évaluer le niveau de sécurité de notre cryptosystème. La probabilité P telle que nous allons la définir ci-dessous est une fonction qui à un événement associe un nombre compris entre 0 et 1 et censé mesurer les chances de réalisation de cet événement.
Avant d'aborder la probabilité, nous allons voir quels que concept du dénombrement.

Définition 2.7.1 *(**Cardinal d'un ensemble**) Soit* E *une ensemble. On appelle cardinal de l'ensemble* E*, noté* $card(\mathrm{E})$ *ou* $|\mathrm{E}|$*, le nombre d'éléments de* E.

Définition 2.7.2 *(**Produit cartésien d'ensembles**) Soit* $n \geq 2$ *et* $\mathrm{E}_1, ..., \mathrm{E}_n$ *des ensembles. On définit le produit cartésien des* $\mathrm{E}_1, ..., \mathrm{E}_n$ *de la façon suivante:*
$\mathrm{E}_1 \times ... \times \mathrm{E}_n = \{(x_1, ..., x_n)$ *telsque* $\forall i \in \{1, ..., n\}, x_i \in \mathrm{E}_i\}$

Proposition 2.7.3 ***(produit cartésien d'ensembles finis)***
Soient $\mathrm{E}_1, ..., \mathrm{E}_n$ *des ensembles fnis. Alors:*

$$\mathrm{C}ard(\mathrm{E}_1 \times ... \times \mathrm{E}_n) = \prod_{1 \leq i \leq n} \mathrm{C}ard(\mathrm{E}_i)$$

Preuve: *Un élément de* $\mathrm{E}_1 \times ... \times \mathrm{E}_n$ *est un n-uplet* $(x_1, ..., x_n)$ *avec pour tout* i *dans* $\{1, ..., n\}$, $x_i \in \mathrm{E}_i$. *Il y a donc* $\mathrm{C}ard(\mathrm{E}_i)$ *choix pour chaque* x_i, *et donc* $\prod_{1 \leqslant i \leqslant n} \mathrm{C}ard(\mathrm{E}_i)$ *éléments dans* $\mathrm{E}_1 \times ... \times \mathrm{E}_n$.

Définition 2.7.4 ***(p-liste)***
Soient E *un ensemble de cardinal n et p un entier naturel quelconque. On appelle p-liste le nombre de p-uplet des éléments de* E.
Alors le p-liste dans E *est:*

$$card\left(\underbrace{\mathrm{E} \times \mathrm{E} \times ... \times \mathrm{E}}_{pfois} \right) = \left(\underbrace{card(\mathrm{E}) \times card(\mathrm{E}) \times ... \times card(\mathrm{E})}_{pfois} \right) = n^p$$

Définition 2.7.5 ***(Permutations d'un ensemble fini)***
Le nombre de permutrations d'un ensemble E *à n éléments est n! (n factoriel i.e.* $n! = n(n-1)(n-2)...1$*).*
Par convention, $0! = 1$.

Définition 2.7.6 *(Arrangement, p-liste sans répétition)*
Soient n et k deux entiers naturels tels que $k \leq n$. *On appelle arrangement, noté* A_n^k, *le nombre d'injections d'un ensemble à k éléments dans un ensemble à n éléments.*
Sa formule est:

$$\mathrm{A}_n^k = \frac{n!}{(n-k)!}$$

Remarque 2.7.7 *Le nombre d'arrangements de k objets parmi n est aussi le nombre de k-liste sans répétition de ces n objets.*

Définition 2.7.8 ***(Combinaison dans un ensemble fini)***
Soient E *un ensemble de cardinal n et p un entier naturel tel que* $p \leq n$. *La combinaison de p éléments dans un ensemble de n éléments, notée* $\binom{n}{p}$, *est le nombre réel defini par:*

$$\binom{n}{p} = \frac{n!}{p!(n-p)!}$$

Définition 2.7.9 *Soit* Ω *un ensemble et* $\mathfrak{F}$ *une famille d'événements observables sur* Ω. *on appelle probabilité sur* $(\Omega, \mathfrak{F})$ *toute application* P *de* $\mathfrak{F}$ *dans* $[0,1]$ *vérifiant:*

(i) $\mathrm{P}(\Omega) = 1$

(ii) *Pour toute suite* $(A_j)_{j\geq 1}$ *d'événements de* $\mathfrak{F}$ *deux disjoints (incompatibles):*

$$P(A_j) = \frac{card(A_j)}{card(\Omega)} \quad et \quad P(\bigcup_{j\in IN^*} A_j) = \sum_{j=1}^{+\infty} P(A_j)$$

Le triplet $(\Omega, \mathfrak{F}, P)$ *s'appelle espace probabilisé.*

Définir une probabilité sur $(\Omega, \mathfrak{F})$ *c'est en quelque sorte attribuer une "masse" à chaque événement observable, avec par convention une masse totale égale à 1 pour l'événement certain* Ω.

Chapitre 3

Codes correcteurs d'erreurs

" Ordinateurs, Internet, câble, numérique... Depuis la seconde moitié du XXe siècle, tous les prodiges informatiques reposent sur la Théorie des Codes. "

Jean-Jacques Walter

Sommaire

3.1 Introduction aux codes correcteurs d'erreurs

La théorie des codes s'est développée pour répondre au problème de la correction des erreurs introduites dans un système de transmission de l'information. A l'origine développée par des ingénieurs en électronique, elle constitue maintenant une branche des mathématiques discrètes.

Le support physique utilisé pour transmettre ou stocker une information (par exemple le téléphone, l'atmosphère, l'espace (penser aux communications par satellite), mais aussi la mémoire d'un ordinateur, les disques compacts, etc..) soumet cette information à des distorsions indésirables, ou bruit, qui l'altèrent. Ce bruit peut être causé par un rayonnement, une altération du support, des interférences, etc.. L'information recueillie par le receveur du canal est en général différente de celle émise par la source. L'objectif de la théorie des codes est de protéger l'information de cet éventuelle altération.

Pour ce faire, la théorie des codes s'appuie sur des notions mathématiques notamment l'algèbre discret pour la construction de certains codes que nous allons étudier en bref dans la suite.

Dans ce chapitre nous parlerons des codes correcteurs d'erreurs et certaines caractéristiques de ces codes.

3.2 Codage de l'information

3.2.1 Poids et distance de Hamming

Ces deux notions sont introduites par Hamming en 1950, ces notions nous permettent d'estimer l'efficacité d'un code dans le cadre d'un canal où les variables aléatoires définies par les coordonnées sont indépendantes et égales. Dans la suite IF_q désigne le corps à q éléments.

Définition 3.2.1 ***(Poids de Hamming)***
Soit $x = (x_1, x_2, ..., x_n) \in \mathrm{IF}_q^n$. Le poids de Hamming de x, noté $wt(x)$ est égal au nombre de coordonnées non nulles de x.

$$wt(x) = card\{i : 1 \leq i \leq n / x_i \neq 0\}$$

Définition 3.2.2 ***(distance de Hamming)***
Soient x, y deux éléments de IF_q^n.
La distance de Hamming de x et y, notée $d_{\mathrm{H}}(x, y)$ est égale au nombre d'indice i où les coordonnées de x et y diffèrent.

$$d_{\mathrm{H}}(x, y) = wt(x - y) = card\{i : 1 \leq i \leq n / x_i \neq y_i\}$$

3.2.2 Code:

Définition 3.2.3 *Un code sur* IF_q*, de longueur n, est un sous-ensemble* $\mathscr{C}$ *de* IF_q^n*. L'ensemble* IF_q *est appelé l'alphabet, n la longueur du code* $\mathscr{C}$ *et les éléments de* $\mathscr{C}$ *sont appelés les mots du code.*

Exemple 3.2.4 *Le code*

$$\mathscr{C} = \{(0,1,1,0,1,0),(1,1,1,0,1,1),(1,1,1,1,1,1),(1,1,0,0,0,0),(0,1,1,1,0,0)\}$$

est un code de longueur 6 sur l'alphabet $\mathrm{IF}_2 = \{0,1\}$*. Dans ce cas* $q = 2$*.*

3.2.3 Codage des mots:

La transmission de l'information à travers un canal de communication nécessite le codage de l'information qui traverse ce canal pour la fiabilité de cette information. Ainsi cette procédure de codage peut se faire de plusieurs manières.

Codage par vote majorité:

Supposons que Modou désire transmettre une chaîne de bits $m = (m_1, m_2, ..., m_k)$ à Aida. Comment peuvent-ils s'assurer que le message a bien été transmis à partir de cette méthode de codage?
Un moyen simple serait d'envoyer chaque bit du message plusieurs fois. Si Modou transmet chaque bit trois fois, Aida déduirait le bit en effectuant un vote majoritaire.

Exemple 3.2.5 *Par exemple, si elle envoie 000 et que Aida reçoit 010 (une erreur ayant affecte le deuxième bit), il conclurait que Modou a transmis le bit 0. Cette procédure ne fonctionnera pas si deux ou trois erreurs affectent la transmission.*

Cette méthode de codage de l'information s'appelle le codage par vote majorité.

Codage par bit parité:

Cette procédure de codage consiste ajouter un bit de parité dans l'information. C'est-à-dire ajouter un bit de tel sorte que le nombre de 1 de l'information soit pair.

Exemple 3.2.6 *Si Modou envoie l'information m=(1101) à Aida, cette dernière recevra l'information m'=(11011). Dans ce cas Aida saura que l'information envoyée est m=(1101).*

Codage à partir de la matrice génératrice:

Soient m l'information envoyée et G la matrice génératrice d'un code quelconque. Coder l'information m à partir de cette méthode consiste à trouver l'information $m' = m.\mathrm{G}$.
Cette méthode de codage nous permet d'ajouter une redondance dans le mot reçu.

Exemple 3.2.7 *si le mot envoyé est* $m = (m_1, ..., m_k)$ *alors le mot reçu sera* $m' = (m'_1, ..., m'_k, m'_{k+1}, ..., m'_n)$. *(avec* $m' = m.\mathrm{G}$ *et* $n > k$*).*

3.3 Codes linéaires

Les codes linéaires sont des codes qui ont une structure d'espaces vectoriels. En effet, les outils de l'algèbre linéaire facilitent dans ce cas les opérations d'encodage et de décodage.

Définition 3.3.1 ***(Code linèaire)***
Un code $\mathscr{C}$ *dans* IF_q^n *est dit linéaire si* $\mathscr{C}$ *est un* IF_q*-sous espace vectoriel de* IF_q^n*. Dans ce cas, on note k sa dimension.*
Si $\mathscr{C}$ *est linéaire, on peut remarquer que, si x et y sont dans* $\mathscr{C}$*, alors* $x - y$ *appartient également à* $\mathscr{C}$*. Comme* $d_{\mathrm{H}}(x, y) = wt(x - y)$*, la distance minimale de* $\mathscr{C}$ *est égale au minimum des poids des éléments non nuls de C. On a:*

$$d(\mathscr{C}) = wt(\mathscr{C}) = min(wt(x)), x \in \mathscr{C} - \{0\}$$

Définition 3.3.2 ***(matrice génératrice)***
Soit $\mathscr{C}$ *un code linèaire de longueur n et de dimension k. Une matrice génératrice de* $\mathscr{C}$ *est une matrice* $k \times n$ *dont les lignes forment une base du code* $\mathscr{C}$*.*

Définition 3.3.3 ***(Code orthogonale)***
Soit $\mathscr{C}$ *un code linéaire dans* IF_q^n *de dimension k. Le code orthogonal de* $\mathscr{C}$*, noté* $\mathscr{C}^{\perp}$*, est défini par:*

$$\mathscr{C}^{\perp} = \{x \in \mathrm{IF}_q^n / \forall y \in \mathscr{C}, < x, y >= 0\}$$

où $< x, y >= \sum_{i=0}^{n} x_i y_i$ *(produit scalaire de deux vecteurs).*

Définition 3.3.4 ***(matrice de controle)***
Une matrice de controle ou de parité de $\mathscr{C}$ *est la matrice génératrice de* $\mathscr{C}^{\perp}$*.*
Si l'ordre de la matrice génératrice de $\mathscr{C}$ *est* $k \times n$ *alors celui de la matrice de controle de* $\mathscr{C}$ *est* $(n - k) \times n$*.*

3.3.1 Codes de Reed Muller RM(r,m)

Historique des codes de Reed Muller

Etudiant l'algèbre de Boole aux circuits de communication, en 1954 D.E Muller construit une classe de codes connus sous le nom de Reed Muller car S.I Reed aura soin de développer un algorithme de décodage en vote majoritaire pour ces codes.

Définitions et propriétés des codes de Reed Muller (r,m):

Définition 3.3.5 ***(Codes de Reed Muller)***
Le code de Reed-Muller d'ordre r à m variables noté RM(r,m)*, de longueur $n=2^m$, pour $0\leq r\leq m$ est l'ensemble des images des fonctions booléennes de degré maximal r en m variables.*

Caractérisation 3.3.6 (Caractéristique d'un code Reed Muller RM(r,m))
Soit $\mathscr{C}$ un code Reed Muller d'ordre r à m variables RM(r,m).
La longueur de ce code est $n=2^m$ et sa dimension est $k=\sum_{i=0}^{r}\mathscr{C}_i^m$.
Preuve: *Soit $\mathscr{C}$ un code de Reed Muller d'ordre r à m variables.*

- *Montrons que la dimension du code $\mathscr{C}$ est $k=\sum_{i=0}^{r}\mathscr{C}_i^m$:*
  Soit f une fonction booléénne.On avait d'après la Forme Algébrique Normale (2.6.10) qu'une base des fonctions booléennes de degré au plus r est formée par les fonctions monômes de degré au plus r. Une simple énumération de ces monômes nous donne alors cette dimension.

- *Montrons que la longueur du code $\mathscr{C}$ $n=2^m$: On a dans l'ordre lexicographique défini dans 2.6.3, les monomes d'une fonction booléenne de degré 1 sont sous la forme:*

$$X_i=\left(\overbrace{\underbrace{0,\ldots,0}_{2^{i-1}},\underbrace{1,\ldots 1}_{2^{i-1}},\ldots,\underbrace{0,\ldots,0}_{2^{i-1}},\underbrace{1,\ldots,1}_{2^{i-1}}}^{2^{m-i}}\right) \tag{1.7.1}$$

  De cet ordre on peut en déduire que la longueur d'un mot de code est :

$$n=2\times 2^{m-i}\times 2^{i-1}$$

  Ce qui donne $n=2^m$.

Remarque 3.3.7 ***(Matrice génératrice d'un code*** RM(r,m)***)***
La matrice génératrice du code RM(r,m)*, noté* G(r,m)*, peut se mettre sous la forme :*

$$G(r,m)=\begin{pmatrix} G(r,m-1) & G(r,m-1)\\ 0 & G(r-1,m-1)\end{pmatrix}$$

Pour $0<r<m$.
Pour plus détails voire [30].

Exemple 3.3.8 *Le code de Reed Muller* RM(1,3)*, c'est le code de longueur* 8 *et de dimension* 4.
Donc ce code possède $2^4 = 16$ *mots.*
Ces 16 mots de codes sont définis par :

$$a_0.1 + a_1.X_1 + a_2.X_2 + a_3.X_3$$

Le tableau suivant nous donne les mots de code du code Reed Muller RM(1,3). *Ce tableau est construit à partir de l'ordre lexicographique défini dans 2.6.3.*

Comb. lin. des monomes	***Mot de code***
0	*0 0 0 0 0 0 0 0*
1	*1 1 1 1 1 1 1 1*
X_1	*0 1 0 1 0 1 0 1*
X_2	*0 0 1 1 0 0 1 1*
X_3	*0 0 0 0 1 1 1 1*
$1+X_1$	*1 0 1 0 1 0 1 0*
$1+X_2$	*1 1 0 0 1 1 0 0*
$1+X_3$	*1 1 1 1 0 0 0 0*
X_1+X_2	*0 1 1 0 0 1 1 0*
X_1+X_3	*0 1 0 1 1 0 1 0*
X_2+X_3	*0 0 1 1 1 1 0 0*
$1+X_1+X_2$	*1 0 0 1 1 0 0 1*
$1+X_1+X_3$	*1 0 1 0 0 1 0 1*
$1+X_2+X_3$	*1 1 0 0 0 0 1 1*
$X_1+X_2+X_3$	*0 1 1 0 1 0 0 1*
$1+X_1+X_2+X_3$	*1 0 0 1 0 1 1 0*

Alors la matrice génératrice de ce code, dont les lignes sont formées par la base qui définit l'ensemble des mots de ce code, est :

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

Remarque 3.3.9 *Cette matrice génératrice nous permet d'encoder un mot envoyé. Si* x *est un mot de code alors l'encodage de ce mot* y *est :* $y = x$G.
Donc l'utilisation de la matrice génératrice nous permet de faciliter l'encodage et le décodage de RM(r,m).

Propriète : 3.3.10 *Les mots de code de Reed Muller* $\mathrm{RM}(r,m)$, $\forall 0 \leq r \leq m-1$ *ont un poids pair.*

Preuve: *(voire [30])*

Définition 3.3.11 *(**Code de Reed Muller d'ordre** r)*

Le code de Reed Muller binaire d'ordre r, $0 \leq r \leq m$, *et de longueur* $n = 2^m$, *noté* $\mathrm{RM}(r,m)$, *est l'ensemble des mots resultant de toutes les combinaisons linéaires des monomes* $1, \mathrm{X}_1, ..., \mathrm{X}_m, \mathrm{X}_1\mathrm{X}_2, \mathrm{X}_1\mathrm{X}_3, ...$ *(jusqu'au degré r). Le code* $\mathrm{RM}(r,m)$ *est linéaire binaire, de dimension* $\sum_{i=0}^{r}\binom{m}{i}$ *et distance minimale* 2^{m-r}.

Exemple 3.3.12 *Pour* $m = 3$, *nous avons ci-dessous les matrices génératrices des codes de Reed Muller d'ordre 0,1,2,3 respectives* $\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2, \mathrm{G}_3$.

$$
\mathrm{G}_3 = \left(\mathrm{G}_2 = \left(\mathrm{G}_1 = \left(\begin{array}{cccccccc|c} \multicolumn{8}{c|}{\mathrm{G}_0 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}} & 1 \\ \hline 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \mathrm{X}_1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \mathrm{X}_2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \mathrm{X}_3 \end{array} \right) \right. \right.
$$

$$
\left. \left. \begin{array}{cccccccc|c} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & \mathrm{X}_1\mathrm{X}_2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & \mathrm{X}_1\mathrm{X}_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & \mathrm{X}_2\mathrm{X}_3 \end{array} \right) \right)
$$

$$
\left(\begin{array}{cccccccc|c} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \mathrm{X}_1\mathrm{X}_2\mathrm{X}_3 \end{array} \right)
$$

Proposition 3.3.13 *Les codes de Reed Muller réalisent la filtration suivante de l'espace des fonctions booléennes.*

$$\{0,1\} = \mathrm{RM}(0,m) \subset \mathrm{RM}(1,m) \subset ... \subset \mathrm{RM}(m-1,m) \subset \mathrm{RM}(m,m) = \mathrm{IF}_2^n$$

Preuve: *(voire [30])*

Proposition 3.3.14 *(Le mot de plus petit poids)*

Soit $f \in \mathrm{RM}(r,m)$ *un mot de plus petit poids. Alors il existe* $f_1, f_2, ..., f_r \in \mathrm{RM}(1,m)$, *tel que*

$$f = f_1.f_2...f_r$$

Les f_i *sont les mots de plus petits poids dans le code de Reed Muller* $\mathrm{RM}(1,m)$.

Theoreme 3.3.15 *Le dual d'un code de Reed-Muller est un code de Reed-Muller:*

$$\mathrm{RM}(r,m)^{\perp} = \mathrm{RM}(m-r-1,m) \qquad 0 \leq r \leq m-1$$

$$\mathrm{RM}(m,m)^{\perp} = 0$$

Preuve: *Les codes de Reed-Muller sont des sous espaces vectoriels de* $\mathrm{IF}_2^n, n = 2^m$.
Soient $f \in \mathrm{RM}(r,m), g \in \mathrm{RM}(m-r-1,m)$. *Alors* $deg(f) \leq r$, $deg(g) \leq m-r-1$, *et leur produit est de degré* $m-1$ *au plus,*
donc $fg \in \mathrm{RM}(m-1,m)$,*et d'après 3.3.10* fg *a un poids pair, par conséquence :*

$$< ev(f), ev(g) >= wt(fg) \quad (modulo 2) \equiv 0$$

Donc $\mathrm{RM}(m-r-1,m) \subset \mathrm{RM}(r,m)^{\perp}$.
De plus

$$\begin{aligned} dim(\mathrm{RM}(r,m)) + dim(\mathrm{RM}(m-r-1,m)) &= \sum_{i=0}^{r} \binom{m}{i} + \sum_{i=0}^{m-r-1} \binom{m}{i} \\ &= \sum_{i=0}^{r} \binom{m}{i} + \sum_{i=0}^{m-r-1} \binom{m}{m-i} \\ &= \sum_{i=0}^{r} \binom{m}{i} + \sum_{i=r+1}^{m} \binom{m}{m-i} \\ &= 2^m \\ &= n \end{aligned}$$

Ce qui implique que $\mathrm{RM}(r,m)^{\perp} = \mathrm{RM}(m-r-1,m)$
et comme $\mathrm{RM}(m,m) = \mathrm{IF}_2^n$, *donc* $\mathrm{RM}(m,m)^{\perp} = \{0\}$.

Groupe d'automorphismes

Soit $\mathrm{C}(n,k)$ un code linéaire sur IF, avec des coordonnées indiqués par les éléments de $\mathrm{I}_n = \{1,2,...,n\}$. Posons S_n le groupe symétrique de toutes les permutations de n-symboles. Soit σ une permutation de S_n, elle est définie, de manière équivalente, par sa matrice P_σ de taille $(n \times n)$, avec $\mathrm{P}_\sigma = [\mathrm{P}_{ij}]$ tel que $p_{ij} = 1$ ssi $i = \sigma(j)$.
L'action d'une permutation $\sigma \in \mathrm{S}_n$ sur un vecteur $x \in \mathrm{IF}^n$ est donnée par l'équation suivante :

$$\forall x = (x_1, x_2, ..., x_n) \in \mathrm{IF}^n, \sigma(x) = (x_{\sigma(1)}, x_{\sigma(2)}, ..., x_{\sigma(n)})$$

Définition 3.3.16 *(**Groupe d'authomorphisme de** C)*
On appelle groupe d'automorphisme de C *(noté* $Aut(\mathrm{C})$*) est l'ensemble des éléments de* $\mathrm{W}_n(\mathrm{IF})$ *qui laissent C globalement invariant, c'est à dire qui transforment un mot de* C *en un autre mot de* C.

$$(\gamma, \sigma, d) \in Aut(\mathrm{C}) <=> \forall c \in \mathrm{C}, (\gamma, \sigma, d)c = (\gamma(c_{\sigma(1)}.d_{\sigma(1)}), \gamma(c_{\sigma(2)}.d_{\sigma(2)}), ..., \gamma(c_{\sigma(n)}.d_{\sigma(n)})) \in \mathrm{C}$$

Pour plus de détails voire [30].

Groupe d'automorphismes de $\mathrm{RM}(r,m)$

Définition 3.3.17 .

- *On appelle* $\mathrm{GL}(m)$ *le Groupe Linèaire Général d'ordre* m*, i.e. le groupe de transformations linèaires inversibles de* IF_2^m*. Une fonction* τ *est dans* $\mathrm{GL}(m)$ *si elle est de la forme* $\tau(x) = \mathrm{A}x$*, où* A *est une matrice de taille* $m \times m$ *inversible à coefficient dans* IF_2*.*
- *On appelle* $\mathrm{GA}(m)$ *le Groupe Affine Général d'ordre* m*, c'est le groupe de transformations affines inversibles de* IF_2^m*. Une fonction* τ *est dans* $\mathrm{GA}(m)$ *est de la forme* $\tau(x) = \mathrm{A}x + b$*, où* A *est une matrice de taille* $m \times m$ *inversible à coefficients dans* IF_2*, et* b *est un élément de* IF_2^m*.*

Soit $\mathrm{A} = (a_{ij})$ *une* $(m,m)-$*matrice inversible, et* b *un vecteur binaire de* IF_2^m*. Définissons la transformation suivante sur l'ensemble des mots binaires de longueur* m *qui envoie 0 à* b*.*

$$\begin{array}{rccc} \sigma: & \mathrm{IF}_2^m & \to & \mathrm{IF}_2^m \\ & x^t = (x_1,...,x_m) & \mapsto & \mathrm{A}.x^t + b \end{array}$$

L'ensemble de toutes les transformations σ *forme un groupe, non commutatif pour la composition, noté* $\mathrm{GA}(m)$*, avec la composition comme opération du groupe. L'ordre de ce groupe est:*

$$|\mathrm{GA}(m)| = 2^m(2^m-1)(2^m-2)(2^m-2^2)...(2^m-2^{m-1})$$

C'est le nombre de possibilité dont on peut choisir les colonnes de la matrice A *et le vecteur* b*.* A *est une matrice inversible, donc elle est de rang* m*, et il y a* (2^m-1) *possibilités de choisir la première colonne,* (2^m-2) *possibilités pour la deuxième colonne, et ainsi de suite.*
Pour le vecteur b *nous* 2^m *possibilités. Une approximation de l'ordre es la suivante:*

$$|\mathrm{GA}(m)| = 0.29 \times 2^{m^2+m}$$

Pour chaque transformation $\sigma \in \mathrm{GA}(m)$*:*

$$\begin{array}{rccc} \sigma: & \mathrm{IF}_2^m & \to & \mathrm{IF}_2^m \\ & \alpha & \mapsto & \sigma(\alpha) \end{array}$$

Nous définissons une permutation associée π_σ *sur l'espace* IF_2^n

$$\begin{array}{rccc} \pi_\sigma: & \mathrm{IF}_2^n & \to & \mathrm{IF}_2^n \\ & x = (x_{\alpha_1},...,x_{\alpha_n}) & \mapsto & (x_{\sigma(\alpha_1)},...,x_{\sigma(\alpha_n)}) \end{array}$$

Theoreme 3.3.18 *Le groupe d'automorphismes des codes* RM *vérifie:*

$$\mathrm{Aut}(\mathrm{RM}(r,m)) \subseteq \mathrm{Aut}(\mathrm{RM}(r+1,m)), 1 \le r \le m-2$$

Preuve: *Nous allons faire une démonstration par récurrence sur* r*:*

- *Démontrons que c'est vrai pour* $r = 1$*:*
  Soit P *une permutation du groupe d'automorphismes de* $\mathrm{RM}(1,m)$*, donc pour tous les monomes de degré* 1*,* $x_{i\,i=1...m}$ *de la matrice génératrice du code, nous avons:*

$$x_i \in \mathrm{RM}(1,m) \Rightarrow x_i \mathrm{P} \in \mathrm{RM}(1,m) = \mathrm{V}ect(1, x_1, ..., x_m)$$

  Pour les éléments $\{x_{i_1 x_{i_2}}\}$*, nous avons:*

$$(x_{i_1} x_{i_2})\mathrm{P} = (x_{i_1}\mathrm{P})(x_{i_2}\mathrm{P}) \in \mathrm{RM}(2,m)$$

  Alors:

$$\forall c \in \mathrm{RM}(2,m), c\mathrm{P} \in \mathrm{RM}(2,m) \Rightarrow \mathrm{P} \in \mathrm{A}ut(\mathrm{RM}(2,m))$$

  D'où:

$$\mathrm{A}ut(\mathrm{RM}(1,m)) \subseteq \mathrm{A}ut(\mathrm{RM}(2,m))$$

- *Nous supposons que c'est vrai pour r, i.e.*

$$\mathrm{A}ut(\mathrm{RM}(r-1,m)) \subseteq \mathrm{A}ut(\mathrm{RM}(r,m))$$

- *Démontrons que c'est vrai pour* $r+1$*:*
  Soit P *une permutation du groupe d'automorphismes de* $\mathrm{RM}(r,m)$*, donc pour tous les monomes de degré* 1*,* $\{x_i\}_{i=1...m}$ *de la matrice génératrice du code, nous avons:*

$$x_i \in \mathrm{RM}(r,m) \Rightarrow x_i\mathrm{P} = f_i \in \mathrm{RM}(r,m)$$

  Pour les éléments $\{x_{j1}...x_{jl}\}, 1 < l \leq r$*, nous avons:*

$$(x_{j1}...x_{jr})\mathrm{P} = (x_{j1}\mathrm{P})...(x_{jr}\mathrm{P}) = f_{j1}...f_{jr} \in \mathrm{RM}(r,m)$$

  Supposons que P *n'est pas dans* $\mathrm{A}ut(\mathrm{RM}(r+1,m))$*,*
  donc il existe un monome v *d'ordre* $(r+1)$ *de la matrice génératrice de* $\mathrm{RM}(r+1,m)$*, tel que:* $v\mathrm{P} \notin \mathrm{RM}(r+1,m)$*.*

$$\exists \{i_1, ..., i_{r+1}\} | (x_{i_1}...x_{i_{r+1}})\mathrm{P}, d'ordre \geq r+2, pour\ r+2 \leq m$$

  Mais $\mathrm{P} \in \mathrm{A}ut(\mathrm{RM}(r,m))$*, alors pour tout* $\{j_1, ..., j_r\} \subset \{i_1, ..., i_{r+1}\}$*:*

$$(x_{j_1}...x_{j_r})\mathrm{P} = f_{j_1}...f_{j_r} \in \mathrm{RM}(r,m)$$

  Ainsi $\forall i \in \{i_1, ..., i_{r+1}\}$*,* f_i *a un monome de degré au moins deux contenant dex variables différentes des variables du monome de grand degré* r *de* $\widetilde{f_i}$*. Ceci n'est pas possible puisque dans ce cas* $f_{i_1}...f_{i_r}$ *contiendra un monome de degré* $2r$*. Ceci prouve que:*

$$\mathrm{P} \in \mathrm{A}ut(\mathrm{R}m(r+1,m)).$$

  D'où

$$\mathrm{A}ut(\mathrm{RM}(r,m)) \subseteq \mathrm{A}ut(\mathrm{RM}(r+1,m))$$

3.3.2 Code de Goppa

Soit $G(x)$ un polynôme à coefficients dans IF_{q^m} et soit

$$S_m = \frac{IF_{q^m[x]}}{< G(x) >}$$

L'anneau des polynômes à coefficients dans IF_{q^m} modulo $< G(x) >$.
Si $G(\alpha) \neq 0$, alors le polynôme $x - \alpha$ est inversible dans S_m. En effet, si on divise $G(x)$ par $x - \alpha$ on obtient

$$G(x) = q(x)(x - \alpha) + G(\alpha)$$

On en déduit alors que $q(x)(x - \alpha) \equiv -G(\alpha) \bmod < G(x) >$ et donc que

$$[-G(\alpha)^{-1} q(x)](x - \alpha) \equiv 1 \quad mod \quad < G(x) >$$

Mais comme nous avons alors que:

$$q(x) = \frac{G(x) - G(\alpha)}{x - \alpha}$$

nous obtenons finalement que :

$$\frac{1}{x - \alpha} = -\frac{G(x) - G(\alpha)}{x - \alpha} G(\alpha)^{-1}.$$

Cette définition de $(x - \alpha)^{-1}$ nous permet de définir les cods de Goppa.

Définition 3.3.19 *Soit $G(x)$ un polynome sur IF_{q^m} et soit $£ = \{\alpha_1, ..., \alpha_n\}$ un ensemmble d'éléments de IF_q tel que $G(\sigma_i) \neq 0$ avec $0 \leq i \leq n$, nous definissons*

$$\Re_a = \sum_{i=1}^{n} \frac{a_i}{x - \alpha_i} \in S_m$$

Alors le code de Goppa noté $\Gamma = \Gamma(£, G)$ est donné par:

$$\Gamma(£, G) = \{a \in IF_q^n \quad \Re_a(x) \equiv 0 \quad mod \quad < G(x) >\}$$

La matrice de controle d'un code de Goppa est alors de la forme suivante:

$$H = \begin{pmatrix} G(\alpha_1)^{-1} & G(\alpha_2)^{-1} & ... & G(\alpha_n)^{-1} \\ \alpha_1 G(\alpha_1)^{-1} & \alpha_2 G(\alpha_2)^{-1} & ... & \alpha_n G(\alpha_n)^{-1} \\ \alpha_1^2 G(\alpha_1)^{-1} & \alpha_2^2 G(\alpha_2)^{-1} & ... & \alpha_n^2 G(\alpha_n)^{-1} \\ . & . & ... & . \\ . & . & ... & . \\ . & . & ... & . \\ \alpha_1^{r-1} G(\alpha_1)^{-1} & \alpha_2^{r-1} G(\alpha_2)^{-1} & ... & \alpha_n^{r-1} G(\alpha_n)^{-1} \end{pmatrix}$$

où $r = deg(G(x))$.

Caractéristiques 3.3.20 *Soit $G(x)$ un polynome sur IF_{q^m} et soit $£ = \{\alpha_1, ..., \alpha_n\}$ un ensemmble d'éléments de IF_q. Le code de Goppa $\Gamma = \Gamma(£, G)$ est un code de longueur $n = 2^m$, de dimension $k = n - m \times r$ et de distance minimale $d \geq r + 1$ où $r = deg(G(x))$.*

3.4 Décodage de l'information

Définition 3.4.1 ***(Condition de décodage d'ordre e)***
Un code $\mathscr{C}$ de longueur n sur un alphabet IF_q vérifie la condition de décodage d'ordre e si pour tout x' de IF_q^n, il existe au plus un mot x de $\mathscr{C}$ tel que $d_{\mathrm{H}}(x,x') \leq e$. Cette condition est équivalente à ce que les fermées (pour la distance de Hamming) de rayon e, centrées sur les mots de $\mathscr{C}$ soient deux à deux disjointes.

Exemple 3.4.2 *Soient $\mathrm{IF}_2 = \{0,1\}$, $n = 5$, $e = 1$ et soit le code :*

$$\mathscr{C} = \{x = (0,1,1,1,0), y = (1,0,1,0,1), z = (1,1,0,1,1)\}$$

Lorsqu'on reçoit le mot $(1,1,1,0,1)$ en sachant qu'il y a une erreur au plus, le mot envoyé est y.

Remarque 3.4.3 *Cette méthode de décodage présente parfois des problèmes dans le cas où les boules ne sont pas disjointes.*

Définition 3.4.4 *(distance minimale d'un code)*
La distance minimale d'un code $\mathscr{C}$ est la plus petite des distances non nulles entre les mots de $\mathscr{C}$. C'est-à-dire la plus petite des distances entre les mots distincts de $\mathscr{C}$.

$$d_{min} = min\{d_{\mathrm{H}}(x,y)/(x,y) \in \mathscr{C} \times \mathscr{C}, x \neq y\}$$

Theoreme 3.4.5 *Soit d la distance minimale d'un code $\mathscr{C}$. Si $2e+1 \leq d$, alors $\mathscr{C}$ vérifie la condition de décodage d'ordre e.*
Preuve: *La démonstration de ce théorème se fait en montrant que la condition indiquée implique que les boules $\mathrm{B}(x,e)$, avec $x \in \mathscr{C}$, sont deux à deux disjointes. Soient x, y deux éléments du code $\mathscr{C}$, nous allons montrer que les boules $\mathrm{B}(x,e)$ et $\mathrm{B}(y,e)$ sont disjointes.*
Soit t un élément quelconque de $\mathrm{B}(x,e)$ Alors on a $d(x,t) \leq e$,
De plus d'après les propriétés de la distance on a : $d(x,y) \leq d(x,t) + d(y,t)$
Soit d la distance minimale du code, alors $d \leq d(x,y)$,
Ainsi $d \leq d(x,t) + d(y,t)$, or $d(x,t) \leq e$;
Ce qui donne $d \leq e + d(y,t)$, or d'après la condition du théorème on a $2e+1 \leq d$;
Ce qui implique $2e+1 \leq e + d(y,t)$,
Donc $e+1 \leq d(y,t)$,
D'où $e < d(y,t)$;
Par suite $t \notin \mathrm{B}(y,e)$;
Puis que t est quelconque, alors les boules B(x,e) et B(y,t) sont disjointes.
Ce qui entraine que Le code $\mathscr{C}$ vérifie la condition de décodage d'ordre e.

Décodage à maximum de vraisemblance (MLD):

On appelle MLD (Maximum Likelihood Decoding) l'algorithme qui permet d'obtenir le mot de code qui a la plus grande probabilité d'être transmis ayant reçu le mot erroné. Ce type de décodage est trés utilisé dans les codes de Reed Muller (pour plus de détails voire [**?**]).

Décodage à distance minimale (MDD):

On appelle MDD (Minimum Distance Decoding), l'algorithme qui permet d'obtenir le mot de code le plus proche du mot reçu, au sens de la métrique de Hamming ou euclidienne.

Chapitre 4

Cryptographie

“ En mathématiques, prouver que quelque chose est impossible n’a « aucun intérêt pratique ». Mais en cryptologie « si on peut garantir que l’adversaire est dans l’impossibilité d’accéder à des données la preuve devient utile » ”

Jacques Stern

Sommaire

4.1 Introduction à la cryptographie

La cryptologie, étymologiquement la science du secret, est considérée comme une science mathématique qui étudie le secret. Cette science englobe la cryptographie et la cryptanalyse.

La cryptographie est un ensemble de techniques visant à assurer la sécurité des communications.
De nos jours, les cryptographes ont défini des objectifs et des notions de sécurité comme la confidentialité, l'intégrité et l'authentification. La cryptographie est devenue une discipline scientifique à part entière qui utilise des concepts mathématiques et informatiques pour construire des cryptosystèmes.

La partie cryptanalyse étudie comment compromettre la sécurité des systèmes. Elle s'attaque aux divers composants : primitives, protocoles, implémentations.

Dans un cryptosystème, on distingue:

- l'espace des messages clairs M sur un alphabet A (qui peut etre l'alphabet romain, mais qui sera dans la pratique $\{0,1\}$ car tout message sera codé en binaire pour pouvoir etre traité par l'ordinateur);
- l'espace des messages chiffrés C sur un alphabet B (en général égal à A);
- l'espace des clés K
- un ensemble E de transformations de chiffrement (chaque transformation étant indexée par une clé):
$$\mathrm{E}_k : m \in \mathrm{M} \rightarrow c \in \mathrm{C}$$
- un ensemble D de transformations de déchiffrement (chaque transformation étant indexée par une clé):
$$\mathrm{D}_k : c \in \mathrm{C} \rightarrow m \in \mathrm{M}.$$

Alors l'ensemble formé $(\mathrm{M}, \mathrm{C}, \mathrm{K}, \mathrm{E}, \mathrm{D})$ est appelé un cryptosystème.
Ainsi nous distingons deux types de cryptosysémes:

- **les cryptosystèmes à clé partagé**: Jusqu'au milieu des année 70, les seuls cryptosystèmes connus étaient les crytosystèmes à clé partagée (ou symétrique). Dans ce cryptosystème, la clé de chiffrement KC est la meme que la clé de déchiffrement KD (ou du moins, la connaissance de la clé de chiffrement permettait d'en déduire

la clé de déchiffrement) ce qui obligeait à garder secrète la clé. Cependant, ce cryptosystème présente un problème crucial sur l'échange de clé surtout dans le cas d'un système développé à grande echelle.

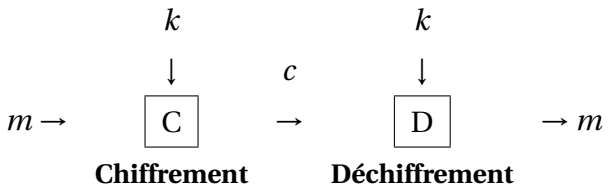
- **les cryptosystèmes à clé publique**: En 1976, W. Diffie et M. Hellman introduisirent le concept de cryptographie à clé publique (ou asymétrique)dans [12]. Dans ce type de système, la clé de chiffrement est publique, c'est à dire connue de tous. Seule la clé de déchiffrement reste secrète. Si n personnes veulent communiquer secrètement 2 à 2, il leur faut en tout $2n$ clés (chacune d'etient une clé secréte et diffuse une clé publique), alors que si elles utilisent un chiffrement symétrique, il leur faut une clé secréte pour chaque paire de correspondants, c'est à dire en tout $\frac{n}{2}$ clés.
  En 1978, le premier système de chiffrement à clé publique fut introduit par R. Rivest, A. Shamir et L. Adleman dans [28]: le système RSA. Ce système est un cryptosystème qui ait résisté à la cryptanalyse.

  Dans la suite de ce chapitre, nous présentons les cryptosystèmes symétriques et asymétriques avec une présentation des différents algorithmes. Nous ferons une description des limites de ces cryptosystèmes et enfin voir les applications de ces derniers.

4.2 Cryptographie symétrique

Le cryptosystème symétrique est très souvent utilisé bien qu'il souffre de problèmes inhérents au fait qu'émetteur et récepteur doivent partager la même clé.
Le premier argument en faveur des systèmes symétriques est qu'ils sont plus rapides que les systèmes asymétriques car ils utilisent directement les instructions câblées des processeurs du commerce.
Dans un premier temps, nous présentons quelques primitives (chiffrement par bloc, par flot et fonctions de hachage) qui permettent de construire les protocoles ou mécanismes garantissant la confidentialité, l'intégrité et l'authentification de l'origine des données (systèmes de chiffrement et codes d'authentification de message : MAC).

La clé de chiffrement est la meme que la clé de déchiffrement.



Définition 4.2.1 *Le chiffrement à clé partagée est défini par trois algorithmes:*

- ***Algorithme de génération de clés :***
  $K_G(l) = k$*: à partir d'un paramétre de sécurité* G*, il produit une clé aléatoire de* l *bit.*

- ***Algorithme de chiffrement:***
  $E_k(m) = c$ *produit le chiffré à partir de la clé* k *et du message* m*.*

- ***Algorithme de déchiffrement:***
  $D_k(c) = m$*, retrouve le message* m *à partir du chiffré* c *et de la clé* k*.*

Nous verrons dans la suite les différents algorithmes de chiffrement à clé partagée.

4.2.1 Chiffrement DES (Data Encryption Standard)

Le chiffrement DES [11] est conçu par IBM pour le NSA, à partir d'un système antérieur LUCIFER. Il est utilisé depuis 1977 avec une clef K de 56 bits, une clef secréte qui est utilisée aussi bien pour chiffrer que pour dechiffrer.
Le chiffrement et le déchiffrement de DES reposent sur des opérations facilement implantables au niveau matériel.

Description globale du système:
Un bloc de 64 bits est chiffré ainsi:

1. On fait une permutation IP (initial permutation) sur les bits de ce bloc et on le découpe en 2 moitié G_0 et D_0.

2. On répéte 16 fois (pour $i = 1, ..., 16$) l'opération suivante, appelé ronde, utilisant une sous-variante K_i de K : $(G_{i-1}, D_{i-1}) \rightarrow (G_i, D_i)$.

3. On finit en calculant $IP^{-1}(D_{16}, G_{16})$.

Le principe de ce chiffrement est de chiffrer des blocs de message de taille fixe (64, 128 ou 256 bits).

Les attaques sur DES:
Les principales attaques sur DES sont:

$\Rightarrow$ **Attaque exhaustive :** C'est une recherche exhaustive de la clé sur un ensemble.

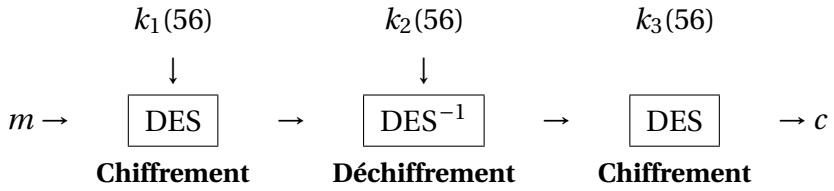
⇒ **Cryptanalyse linéaire :**La cryptanalyse linéaire consiste à faire une approximation linéaire de l'algorithme de chiffrement en le simplifiant. En augmentant le nombre de couples disponibles, on améliore la précision de l'approximation et on peut en extraire la clé.

⇒ **Cryptanalyse différentielle :** La cryptanalyse différentielle consiste en l'étude sur la manière dont les différences entre les données en entrée affectent les différences de leurs sorties.

4.2.2 Le triple DES (3DES):

La taille de l'ensemble de clés ($\{0,1\}^{56}$) constitue la plus grande menace qui pèse sur DES avec les ressources de calcul actuels. En 1999 il a suffit de 24 heures pour trouver la clé à partir d'un clair texte connu en utilisant une technique brute force massivement parallèle.

Le triple DES (3DES) nous met à l'abri de ces attaques brute force en augmentant l'espace des clés possibles à $\{0,1\}^{112}$.
Schématiquement, il fonctionne de la manière suivante:



4.2.3 AES (Advanced Encryption Standard):

L'Advanced Encryption Standard (AES) sera le nouveau standard FIPS (Federal Information Processing Standard). Il spécifiera un algorithme cryptographique à clé secrète. Cet algorithme pourra être utilisé par les organisations gouvernementales américaines pour protéger les informations sensibles (non-classifiées). Le NIST (National Institute of Standards and Technology) anticipe que l'AES sera aussi utilisé par d'autres organisations, institutions et individus en dehors du gouvernement américain et en dehors des Etats-Unis dans certains cas.

L'AES a été développé pour remplacer le DES (Data Encryption Standard). Le DES est en train d'être abandonné et est aujourd'hui uniquement utilisé dans certains systèmes légaux.

L'AES est un algorithme de chiffrement symétrique comme DES, ce qui veut dire que l'émetteur et le destinataire utilise une même clé, k, pour chiffrer et déchiffrer le message.

Structure de l'algorithme

L'algorithme est iteratif. Il peut etre decoupe en 3 blocs :

- **Initial Round.** C'est la premiere et la plus simple des etapes. Elle ne compte qu'une seule operation : Add Round Key.
- N **Rounds.** N etant le nombre d'iterations. Ce nombre varie en fonction de la taille de la cle utilisee. 128 bits → N = 9, 192 bits → N = 11, 256 bits → N = 13. Cette deuxième étape est constituée de N iterations comportant chacune les quatres operations suivantes : Sub Bytes, Shift Rows, Mix Columns, Add Round Key.
- **Final Round.** Cette étape est quasiment identique a l'une des N iterations de la deuxième étape. La seule difference est qu'elle ne comporte pas l'opération Shift Columns.

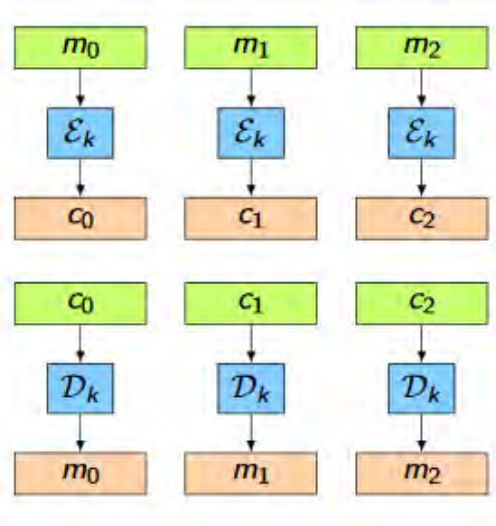


Figure 4.1 – Procédure de chiffrement ECB

Figure 4.2 – Image claire

4.2.4 Mode ECB (Electronic Code Book):

Le mode de chiffrement ECB (Electronic CodeBook) a été le premier mode inventé pour chiffrer une donnée. Le principe est très simple : on découpe les données à chiffrer en bloc de X bits, et on XOR chaque bloc avec une clef de chiffrement de X bits.

$$C_i = K \oplus m_i$$

Ce chiffrement souffre de deux inconvénients majeurs:

- Le chiffrement d'une image donne un chiffré visible.
- Vous ne devez aussi jamais réutiliser la même clef pour chiffrer 2 messages différents. En effet, si m_1 et m_2 sont deux messages chiffrés en ECB ayant comme

Figure 4.3 – Image chiffrée par ECB

chiffré respectif c_{m_1} et c_{m_2} avec la clé K.

$$c_{m_1} = \mathrm{K} \oplus m_1$$

$$c_{m_2} = \mathrm{K} \oplus m_2$$

$$c_{m_1} \oplus c_{m_2} = (\mathrm{K} \oplus m_1) \oplus (\mathrm{K} \oplus m_2) = \mathrm{K} \oplus \mathrm{K} \oplus m_1 \oplus m_2 = m_1 \oplus m_2$$

Vous obtenez donc les données en clair XORées entre elles. Avec un peu d'analyse, on peut retrouver m_1 et m_2.

4.2.5 Mode CBC (Cipher Block Chaining):

Pour corriger les problèmes de ECB, on a alors inventé le mode CBC (Cipher Block Chaining).
On voit que le soucis de ECB vient du fait qu'on réutilise la même clef pour tous les blocs, ce qui fait qu'à entrée (et donc clef) identique, la sortie sera identique. Vu qu'on ne maîtrise pas les données d'entrée, on ne peut jouer que sur la clef de chiffrement. Il faut trouver un moyen de la faire varier pour chaque bloc, pour qu'enfin à données identiques, on obtienne bien une sortie différente. La solution retenue est simplement d'utiliser la sortie du bloc précédent, de la mixer avec la clef et d'utiliser le résultat comme nouvelle clef de bloc :

$$\mathrm{C}_i = \mathrm{K} \oplus \mathrm{C}_{i-1} \oplus m_i$$

- Procédure de chiffrement CBC.
- Pour CBC, l'image chiffrée devient:

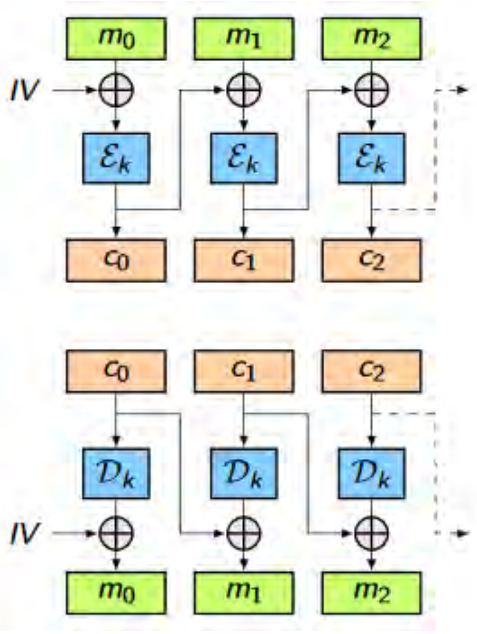


Figure 4.4 – Chiffrement CBC

Figure 4.5 – Image chiffrée par CBC

4.3 Cryptographie asymétrique

Définition 4.3.1 *La cryptographie asymétrique, ou cryptographie à clé publique, est une méthode de chiffrement qui repose sur l'utilisation d'une clé publique (qui est diffusée) et d'une clé privée (gardée secrète), dont l'une permettant de chiffrer le message et l'autre de le chiffrer. Ainsi, l'expéditeur peut utiliser la clé publique du destinataire pour coder un message que seul le destinataire (en possession de la clé privée) peut décoder, garantissant la confidentialité du contenu.*

Remarque 4.3.2 *La clé publique ne permet pas de déchiffrer et aussi de retrouver la clé privée.*

Définition 4.3.3 *Un chiffrement à clé publique se compose de trois algorithmes:*

- ***Algorithme de génération des clés:***
  $\mathrm{K_G}(l) = (p_k, s_k)$*: à partir d'un paramètre de sécurité* G, *il produit un coupe (clé publique, clé privée).*

- ***Algorithme de chiffrement:***
  $\mathscr{C}(m, p_k) = c$ *: utilise la clé publique pour chiffrer un message m.*

- ***Algorithme de déchiffrement :***
  $\mathrm{D}(c, s_k) = m$*: utilise la clé privée pour remonter à m.*

Exemple 4.3.4 ***Chiffrement de RSA:***

Définition 4.3.5 *Ce chiffrement est proposé par Rivest Shamir et Adleman en 1978. Il se base sur :*

- *un modulo* $n = p.q$ *(*p *et* q *premiers).*
- *un ordre du groupe multiplicatif* $\mathrm{Z}_n^\star = \varphi(n) = (p-1)(q-1)$

  $$\forall x \in \mathrm{Z}_n^\star, x^{\varphi(n)} = 1 \ mod(n) \qquad (\textbf{\textit{théorème d'Euler}})$$

- e *un entier premier avec* $\varphi(n)$ *et* d *sont tel que* $d = e^{-1} mod(\varphi(n))$

  $$ed + u\varphi(n) = 1 \qquad (\textbf{\textit{Bezout}})$$

Définition 4.3.6 *(**Théorème d'Euler**)*
Pour tout $m \in \mathrm{Z}_n^\star$*:*

$$(m^e)^d = m^{ed} = m^{1-u\varphi(n)} = m \times 1^{-u} = m \ mod(n)$$

Définition 4.3.7 (Algorithme de RSA)
Soient

- $n = pq$*: module* ***public***
- e*: exposant* ***public****.*
- $d = e^{-1} mod(\varphi(n))$*: exposant* ***secret****.*

Les algorithmes de RSA sont:

- ***Secret:*** $\mathrm{K_G}(l) = (n, e) \bigcup (d)$
- ***Public:*** $\mathrm{C}(m) = m^e mod(n) \rightarrow c$
- ***Secret:*** $\mathrm{D}(c) = c^d = m^{ed} = m mod(n)$

Sécurité de RSA:
La sécurité de RSA se définit suivant le niveau visé:

- *pour casser complètement le système, il faut retrouver l'exposant d ou (p et q):* ***factorisation.***
- *pour déchiffrer un message, il faut retrouver m à partir de m^e:* ***extraction de racine e-ièmes****.*

Proposition 4.3.8 (Problème de RSA)
Etant donné un module RSA, n, un exposant e premier avec $\varphi(n)$ et un élément $y \in Z_n^\star$, calculer x vérifiant:

$$y = x^e mod(n)$$

Proposition 4.3.9 (Difficulté de RSA)
Si on connait la factorisation, on casse RSA:

- *On retrouve toutes les données secrètes*
- *RSA se réduit à la factorisation.*

En pratique:
La factorisation est la seule méthode connue pour casser RSA.

Proposition 4.3.10 (Factorisation et RSA)
La connaissance d'un couple (e, d) suffit à factoriser complétement le modulo.

1. *Un tel couple* (e,d) *fournit une racine carrée modulaire de* 1
   - *Soit m quelconque. De* $m^{ed} = m$, *on déduit* $m^{ed-1} = 1 mod(n)$
   - *Donc* $m^{(ed-1)/2}$ *est une racine carrée de* 1
2. *Or, une telle racine, si elle est non triviale, donne la factorisation de n*
   - *d'une relation* $x^2 = 1 mod(n)$, *on tire:* $(x-1)(x+1) = 0 mod(n)$
   - *Donc* $x-1$ *et* $x+1$ *contiennent les facteurs de n*

Quelles que attaques sur RSA:

Attaques sur modulo commun:

Supposons qu'Alice et Bob utilisent le meme modulo, car ils n'ont pas d'exigence de confidentialité l'un vis-à-vis de l'autre.
Si un meme message est chiffré à l'intention d'Alice et de Bob, il devient possible pour tout le monde de le déchiffrer.

- *message m chiffré avec les exposants* e_A *et* e_B, *supposés premiers entre eux:* $ue_A + ve_B = 1$ ***(Bezout)***
- *soient* $c_A = m^{e_A}$ *et* $c_B = m^{e_B}$ *les chiffrés*
- *n'importe qui peut calculer* $c_A^u c_B^v = m^{ue_A + ve_B} = m$

Remarque 4.3.11 *Le niveau de sécurité de RSA devient faible si plusieurs correspondants partagent le meme modulo.*

Attaques sur un petit exposant:

Un petit exposant public permet d'accélérer les calculs.

Définition 4.3.12 *(**Problème de Logarithme discret**)*
Soit G *un groupe multiplicatif,* $g \in G$ *et* $y \in < g >$. *On définit :*

$$log_g(y) = min\{x > 0 | g^x = y\}$$

Remarque 4.3.13 *(**Limites de RSA**)*
Ces problèmes algorithmiquement difficile dans certains groupes, seront faciles à trouver une solution avec l'arrivée des ***ordinateurs quantiques****.*

4.3.1 Chiffrement Diffe-Hellman

Définition 4.3.14 *Diffie-Hellman est un algorithme d'échange de clefs : il permet à deux personnes de mettre en place un système par lequel elles pourront échanger de façon secrète. C'est le premier algorithme créé dans ce but, en 1976, par Whitfield Diffie et Martin E. Hellman dans [12].*

Le principe en est le suivant : **Echange de Clef par Diffie-Helleman:**

- B = $b^x mod(q)$ (où q un nombre premier et b un élément primitif telque pour tout b choisi sera une puissance de b).
- A = $log_b(\mathrm{B}) mod(q)$
- $\mathrm{B}_i = b^{x_i} mod(q)$ (pour chaque correspondant)
- $\mathrm{K}_{ij} = b^{x_i * x_j} mod(q)$ (clé partagée)
- $\mathrm{K}_{ij} = \mathrm{B}_i^{x_j} mod(q) = \mathrm{B}_j^{x_i} mod(q)$

Pour plus de détails, voire [**?**].

4.4 Cryptographie Hybride

Définition 4.4.1 *La cryptographie hybride est un système de cryptographie faisant appel aux deux grandes familles de systèmes cryptographiques : la cryptographie asymétrique et la cryptographie symétrique. Les logiciels comme PGP et GnuPG reposent sur ce concept qui permet de combiner les avantages des deux systèmes.*

4.4.1 Principe

La cryptographie asymétrique est intrinsèquement lente à cause des calculs complexes qui y sont associés, alors que la cryptographie symétrique brille par sa rapidité. Toutefois, cette dernière souffre d'une grave lacune, on doit transmettre les clés de manière sécurisée (sur un canal authentifié). Pour pallier ce défaut, on recourt à la cryptographie asymétrique qui travaille avec une paire de clés : la clé privée et la clé publique. La cryptographie hybride combine les deux systèmes afin de bénéficier des avantages (rapidité de la cryptographie symétrique pour le contenu du message) et utilisation de la cryptographie "lente" uniquement pour la clé.

4.4.2 Methodologie

La plupart des systèmes hybrides procèdent de la manière suivante. Une clé aléatoire, appellée clé de session, est générée pour l'algorithme symétrique (3DES, IDEA, AES et bien d'autres encore), cette clé fait généralement entre 128 et 512 bits selon les algorithmes. L'algorithme de chiffrement symétrique est ensuite utilisé pour chiffrer le message. Dans le cas d'un chiffrement par blocs, on doit utiliser un mode d'opération comme CBC (présenté 4.2.5), cela permet de chiffrer un message de taille supérieure à celle d'un bloc.
La clé de session quant à elle, se voit chiffrer grâce à la clé publique du destinataire, c'est ici qu'intervient la cryptographie asymétrique (RSA ou ElGamal). Comme la clé est courte, ce chiffrement prend peu de temps. Chiffrer l'ensemble du message avec un algorithme asymétrique serait bien plus coûteux, c'est pourquoi on préfère passer par un algorithme symétrique. Il suffit ensuite d'envoyer le message chiffré avec l'algorithme symétrique et accompagné de la clé chiffrée correspondante. Le destinataire déchiffre la clé symétrique avec sa clé privée et via un déchiffrement symétrique, retrouve le message.

Exemple 4.4.2 ***Hybride Crytographie utilisant RSA, Diffe-Hellman :***

RSA et Diffie Hellman sont largement utilisés de nos jours. Dans ce schéma hybride présenté dans [16], l'algorithme RSA est utilisé pour assurer la sécurité du message en utilisant le processus du chiffrement et du déchiffrement de RSA. L'algorithme Diffie Hellman est utilisé pour générer la clé secrète qui sera utilisée au niveau du chiffrement symétrique. Dans ce modèle proposé, le schéma symétrique utilisé est basé sur des opérations binaires XOR. Le principe est le suivant:

1. *Choisir deux grands nombres premiers* P *et* Q *utilisés dans RSA.*
2. *Générer aléatoirement les nombres* A, B, G *et* R *qui seront utilisés dans Diffe-Hellman.*
3. *Calculer* $N = P * Q$.
4. *Trouver* $phi(N) = (P-1) * (Q-1)$
5. *Choisir un entier* E *tel que* $PGCD(E, phi(N)) = 1$ *(avec* $1 < E < phi(N)$*).*
6. *Calculer* D *telque* $E * D = 1 mod(phi(N))$.
7. *En utilisant Diffe-Hellman, calculons les nombres publiques* $X = G^A mod(R)$ *et* $Y = G^B mod(R)$

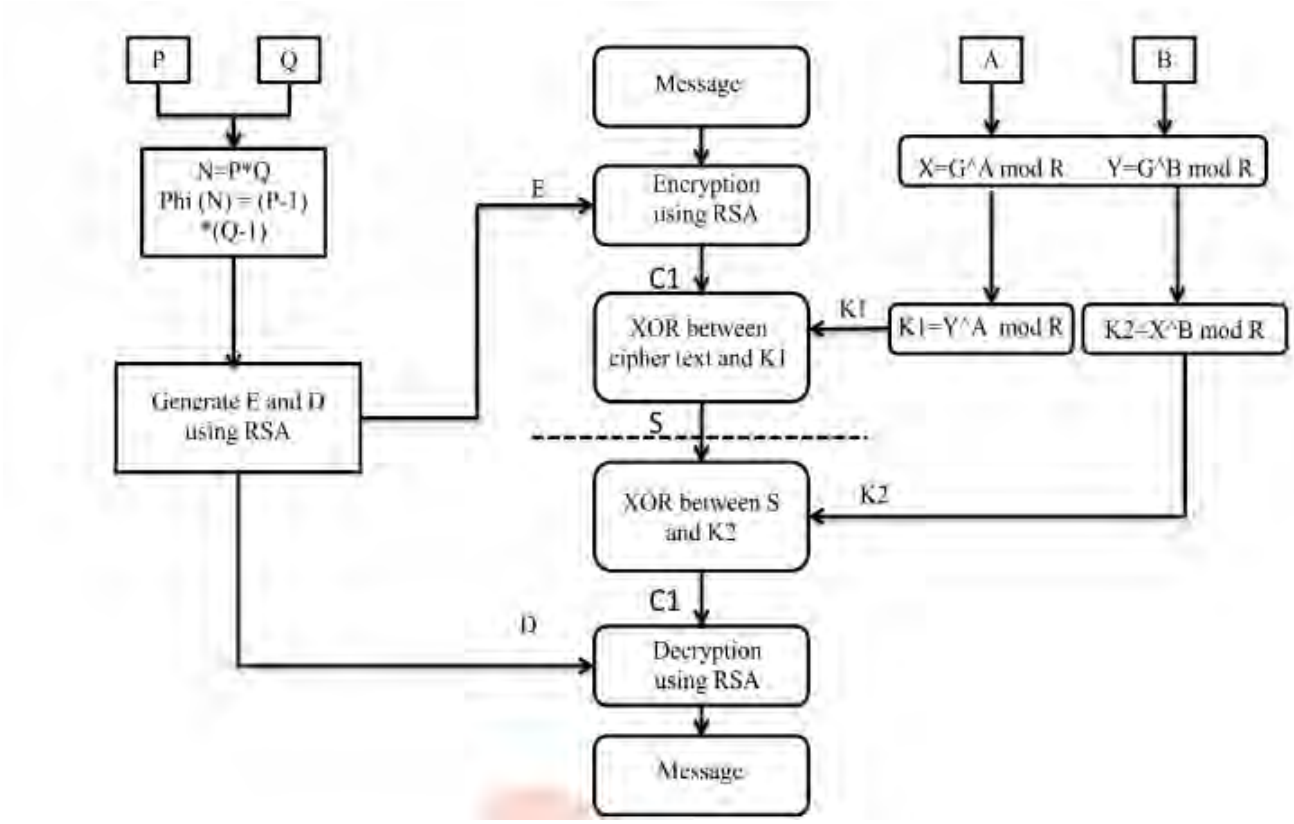


Figure 4.6 – Chiffrement Hybride utilisant RSA, Diffe-Hellman et XOR

8. *Calculons la clé secret (clé symétrique)* $K_1 = Y^A mod(R)$ *et* $K_2 = X^B mod(R)$

9. *Le chiffrement du message se fait en deux étapes:*

   - 1^{ere} *Etape: Message clair chiffré par RSA*

   $$C_1 = (M^E) mod(N)$$

   - *Utilisons les opérations binaires XOR pour chiffrer* C_1 *avec la clé* K_1

   $$S = C_1 \oplus K_1$$

10. *A la reception, le destinataire déchiffre le message à partir de la clé* K_2*,*

$$C_1 = S \oplus K_2$$

11. *En utilisant RSA, calculons*

$$M = (C_1^D) mod(N)$$

Chapitre 5

Cryptosystème basé sur les codes correcteurs d'erreurs

" Avec l'Internet et le Web, la demande en cryptologie a explosé. Et paradoxalement, la cryptologie est passée d'une science du secret à une science de confiance "

Jacques Stern

Sommaire

5.1 Introduction

Le principe du protocole des cryptosystèmes basés sur les codes consiste à faire envoyer par Alice un message contenant un grand nombre d'erreurs, erreurs que seul Bob sait détecter et corriger.
Le premier cryptosystème basé sur les codes est celui de Mc Eliece [20]. Il a été créé en 1978. C'est ainsi que d'autres cryptosystèmes ont été inventés.

Nous verrons dans la suite de ce chapitre ces systèmes de chiffrement basé sur les codes.

5.2 Cryptosystème de McEliece

Le cryptosystème de McEliece est un système de chiffrement asymétrique, inventé en 1978 par Robert McEliece. Ce système, reposant sur un problème difficile de la théorie des codes, utilise les codes de GOPPA.
Le principe du schéma de McEliece est de masquer le mot de code correspondant au message en lui ajoutant autant d'erreurs que possible tout en gardant la possibilité de corriger celles-ci. Si la méthode de correction est gardée secrète, alors seul le destinataire sera en mesure de retrouver le message original.

5.2.1 Description du schéma

Dans le cryptosystème de McEliece, la matrice génératrice G fait partir de 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 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 :

$$G_{pub} = SGP$$

5.2.2 Alogithmes du schéma

Algorithme de Génération de clé:

1. Soit G la matrie generatrice sous forme systèmatique du code de longueur n, de dimension k et t-correcteurs.

2. On choisit aléatoirement S une matrice binaire inversible de dimension (k, k).
3. On choisit aléatoire une matrice de permutation P de dimension (n, n).
4. Calculer $G_{pub} = SGP$.
5. Retourner la clé publique (G_{pub}, t) et la clé privée (S, G, P, γ) où γ est l'algorithme décodage du code utilisé et t le nombre d'erreur maximal que le code peut corriger.

Algorithme de chiffrement:

1. Bob publie la clé publique (G_{pub}, t). Si Alice veut envoyer un message m (de k bits) à Bob.
2. Il (Alice) calcule $c = mG_{pub} + e$ où e est un mot de poids t.
3. Alice envoie c le chiffré à Bob.

Algorithme de déchiffrement:

1. Bob reçoit le chiffré c et détient la clé privée (S, G, P, γ).
2. Il Calcule $c' = cP^{-1} = mSG + eP^{-1}$, c' est alors un mot du code de Goppa de matrice génératrice SG contenant t erreurs.
3. Décoder c' en calculant $c'' = \gamma(c')$
4. Calculer $m = c''S^{-1}$
5. Pour trouver le message clair m.

5.2.3 Avantage du schéma

Les avantages du schéma de McEliece sont:

- Le principal avantage de cette méthode est sa facilité d'implémentation : les seules opérations sont des opérations bit à bit.
- La sécurité croît beaucoup plus avec la taille des clés que pour le système RSA, et le chiffrement est plus rapide.
- Un autre avantage est de reposer sur un problème très différent des algorithmes asymétriques usuels. Cela signifie qu'une percée théorique dans le domaine de la factorisation, qui ruinerait RSA, n'affecterait en rien ce système.

5.2.4 Inconvénient du schéma

L'inconvénient majeur de ce système est sa taille de clé. Par exemple, la clé publique est parfois à l'odre de 2^{19} bits (64 Kio).
Avec cet inconvénient, l'implémentation de ce schéma nécessite beaucoup de place mémoire.

5.3 Cryptosystème de Niederreiter

En 1986, Harald Niederreiter [22] 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 [35].

5.3.1 Description du schéma

Le cryptosystème de Niderreiter utilise la matrice controle H comme la clé secrète. Pour construire une clé publique avec le schéma de Niederreitter, on multiplie H à gauche par une matrice binaire non singulaire d'ordre $(n-k, n-k)$. De la meme manière que Mc Eliece, on multiplie à droite par une matrice de permutation P d'ordre (n, n). Ce qui donne comme clé publique :

$$\mathrm{G}_{pub} = \mathrm{SHP}$$

5.3.2 Algorithmes

Algorithme de génération de clé :

1. Soit H la matrie de controle du code linèaire C de longueur n, de dimension k et t-correcteurs.
2. On choisit aléatoirement S une matrice binaire inversible de dimension $(n-k, n-k)$.
3. On choisit aléatoire une matrice de permutation P de dimension (n, n).
4. Calculer $\mathrm{H}_{pub} = \mathrm{SHP}$.
5. Retourner la clé publique H_{pub} et la clé privée $(\mathrm{S}, \mathrm{H}, \mathrm{P}, \gamma)$ où γ est l'algorithme décodage par syndrome de code C.

Algorithme de chiffrement :

1. Bob publie la clé publique H_{pub}. Si Alice veut envoyer un message m (de n bits) à Bob.

2. Il (Alice) calcule $c = \mathrm{H}_{pub} m^{\mathrm{T}}$

3. Retourner c

Algorithme de déchiffrement :

1. Bob détient la clé privée $(\mathrm{S}, \mathrm{H}, \mathrm{P}, \gamma)$.

2. Calculer $c' = \mathrm{S}^{-1} c = \mathrm{HP} m^{\mathrm{T}}$.

3. Trouver $\mathrm{P} m^{\mathrm{T}}$ à partir de $\mathrm{S}^{-1} c$ grace à l'algorithme de décodage γ

4. Trouver m en appliquant P^{-1} à $\mathrm{P} m^{\mathrm{T}}$

5. Retourner le message clair m.

5.4 Cryptosystème de Sidel'nikov

En 1994, une variante du cryptosystème de McEliece, appelé cryptosystème de Sidel'nikov [33], a été proposé. Cette nouvelle variante remplace les codes Goppa par les codes de Reed Muller.
De la même manière que le cryptosytème de McEliece, ce système utilise la matrice génératrice du code de Reed Muller G, une matrice inversible S et une matrice de permutation P pour construire la clé publique G_{pub} definie par:

$$\mathrm{G}_{pub} = \mathrm{SGP}$$

5.4.1 Alogithmes du schéma

Algorithme de Génération de clé:

1. Soit G la matrie generatrice d'un code de Reed Muller de longueur n, de dimension k et t-correcteurs.

2. On choisit aléatoirement S une matrice binaire inversible de dimension (k, k).

3. On choisit aléatoire une matrice de permutation P de dimension (n, n).

4. Calculer $\mathrm{G}_{pub} = \mathrm{SGP}$.

5. Retourner la clé publique (G_{pub}, t) et la clé privée $(\mathrm{S}, \mathrm{G}, \mathrm{P}, \gamma)$ où γ est l'algorithme décodage du code de Reed Muller et t le nombre d'erreurs que le code de Reed Muller peut corriger.

Algorithme de chiffrement:

1. Bob publie la clé publique (G_{pub}, t). Si Alice veut envoyer un message m (de k bits) à Bob.

2. Il (Alice) calcule $c = mG_{pub} + e$

3. Alice envoie c le chiffré à Bob.

Algorithme de déchiffrement:

1. Bob reçoit le chiffré c et détient la clé privée (S, G, P, γ).

2. Il Calcule $c' = cP^{-1} = mSG + eP^{-1}$, c' est alors un mot du code de Goppa de matrice génératrice SG contenant t erreurs.

3. Décoder c' en calculant $c'' = \gamma(c')$

4. Calculer $m = c''S^{-1}$

5. Pour trouver le message clair m.

5.4.2 Avantage du schéma

- Les codes Reed Muller disposent des algorithmes de décodage très efficients. C'est la raison pour la quelle, les schéma utilisant les codes de Reed Muller ont des algorithmes de déchiffrement beaucoup plus rapide.

- Nous remarquons aussi avec l'utilisation des codes de Reed Muller, la taille des clés du système devient plus petite.

5.4.3 Inconvénient du schéma

Les codes Reed Muller ont une propriété (3.3.13) qui rend son cryptosystème vulnérable.

5.5 Quelles que attaques connues

Dans cette section, nous décrivons les attaques connues sur les cryptosystèmes basés sur les codes.

5.5.1 Les attaques par décodage

Ces types d'attaques portent sur l'algorithme de décodage utilisé par le cryptosystème.Les auteurs ont démontré dans [3] et [15], que la probabilité pour qu'un code Goppa soit équivalent à un code de matrice génératrice G_{pub} est trop faible. Ce qui rend les attaques par décodage difficile à réaliser.

Decodage par ensemble d'information(ISD):

Dans cette attaque, l'adversaire doit décoder un code aléatoire sans connaitre le secret et aussi la structure évidente du code. En efffet, le décodage par ensemble d'information regroupe les algorithmes de cryptanalyses les plus connus qui n'utilisent aucune structure de code. Cette approche a été introduite en 1962 première fois par Prange dans [27]. L'idée est de trouver les k symboles d'informations dans le chiffré de sorte que la restriction de la matrice générateur du code à ces k positions est inversible. Ensuite, le texte clair peut être calculé en multipliant le chiffré par l'inverse de la k-matrice génératrice. Dans [27], la compléxité de cette attaque calculée en se basant sur le facteur de travail pour un code Goppa de longueur n, de dimension k et de distance minimale 2^{t+1} est:

$$\mathrm{WF} = k^3 \frac{\binom{n}{k}}{\binom{n-t}{k}}$$

Par exemple pour un code Goppa $n = 1024$, $k \leq 524$ et $t = 50$, le facteur de travail est$\mathrm{WF} = 2^{64}$.

De plus, des améliorations du décodage par ensemble d'informations ont été proposées par Lee et Brickell en EUROCRYPT88 [18], Leon en 1988 [19] et Stern en 1989 [34]. Toutes ces versions ISD utilisent les codes linéaires binaires.

Récemment au PQCrypto 2008, Bernstein, Lange et Peters [6] ont proposé plusieurs améliorations à l'ISD de Stern présenté dans [34]. Dans [6] Bernstein et al. ont analysé la compléxité de ISD et donné une nouvelle valeur du facteur de travail qui se présente sous la forme:

$$\mathrm{P}_t \times ((n-k)^3/2 + k(n-k)^2 + 2p\sigma\binom{k/2}{p} + \frac{2p(n-k-\sigma)\binom{k/2}{p}^2}{2^\sigma})$$

où

$$\mathrm{P}_t = \frac{\binom{t}{p}\binom{n-t}{k/2-p}\binom{n-k/2-t+p}{k/2-p}\binom{n-k-t+2p}{\sigma}}{\binom{n}{k/2}\binom{n-k/2}{k/2}\binom{n-k}{\sigma}}$$

De meme en 2009, Finiasz et Sendrier [14] ont présenté une amélioration supplémentaire qui va dans le meme sens que Bernstein et al.[6] sans l'analyse de l'attaque.

Dans l'optique d'une amélioration de l'ISD, Peters et Christiane en 2010 dans [26] propose une généralisation de l'algorithme de Lee-Brickell et l'algorithme de Stern.

5.5.2 Les attaques structurelles:

Ces attaques portent la structure de la clé publique. Le cryptanaliste tente de retrouver la clé privé à partir de la clé publique. Il s'agit d'étudier le comportement du cryptosystéme lorsqu'on chiffre une grande famille de messages, choisie pour que certaines composantes du systéme prennent toutes les valeurs possibles.

5.5.3 Les attaques utilisant un distingueur:

Définition 5.5.1

(Distingueur) Un distingueur pour un cryptosystéme donnée est un attaquant capable de détecter que c'est ce cryptosystéme qui est utilisé et non une fonction aléatoire. Le distingueur a éventuellement accés à un oracle de chiffrement ou de déchiffrement.

Sécurité sémantique

La sécurité sémantique annonce que le cryptogramme n'apporte aucune information que l'adversaire ne puisse calculer sans lui. Avec la sécurité sémantique, si l'adversaire sait calculer le moindre bit du message clair à partir du cryptogramme alors la sécurité sémantique n'est pas atteinte.
Un adversaire d'un schéma de chiffrement contre la sécurité sémantique est un algorithme $\mathscr{A}$, à qui on fournit le cryptogramme d'un message x, et qui doit retourner la valeur $f(x)$ d'une information arbitraire sur x.

Définition 5.5.2 ***(Sécurité sémantique)***

Un schéma de chiffrement $\mathscr{C} = (\mathscr{D}, \mathscr{R}_n, \mathscr{G}, \mathscr{E}, \mathscr{D})$ est dit sémantiquement sûr, noté SEM-sûr, si pour tout adversaire $\mathscr{A}$ en temps polynimial, pur toute fonction f sur $\mathscr{D}_n$ et toute distribution de probilité $\mathscr{X}_n$, son avantage:

$$(1) \qquad \mathrm{A}dv^{\mathscr{A}} = |\mathrm{P}rob_{e_{x\in\mathscr{X}_n}\leftarrow \mathrm{G}(1^n)}[\mathscr{A}(\mathscr{E}(e,x)) = f(x)] - \mathrm{P}rob_{e_{x,x'\in\mathscr{X}_n}\leftarrow \mathrm{G}(1^n)}[\mathscr{A}(\mathscr{E}(e,x')) = f(x)]$$

est une fonction négligeable.

*Un adversaire contre un schéma de chiffrement pour la sécurité sémantique est un algorithme qui doit retrouver une information $f(x)$ sur le clair x à partir du cryptogramme $\mathscr{E}(e,x)$.L'expression de l'avantage donné par la relation **(1)** exprime qu'il n'y a pas de différence de probabilité de succés notable de l'algorithme $\mathscr{A}$ pour trouver l'information $f(x)$ qu'on lui fournisse le chiffré du message x (premier terme) ou le chiffré d'un autre message aléatoire x' (second terme).*

La probabilité exprimée dans cette définition est pour une clé de chiffrement aléatoire e fournie par l'algorithme de géénération de clé, et pour des messages clairs aléatoires.

La sécurité sémantique est atteinte si l'avantage de tout adversaire est négligeable, pour toute distribution de probabilité sur les clairs. En particuier si deux messages clairs seulement sont également probales.

Indistinguabilité

Un chiffrement a la propriété d'indistinguabilité si un adversaire est incapable de dire si le cryptogramme qu'on lui présente provient de l'un ou de l'autre message, sachant que le cryptogramme est le chiffré de l'un de ces deux messages.

Un adversaire d'un schéma de chiffrement contre la propriété d'indistinguabilité est un algorithme $\mathscr{A}$ à qui on fournit le cryptogramme d'un message m_i choisi aléatoirement dans un ensemble de deux messages distincts $\{m_0, m_1\}$ et qui doit rendre l'indice $i \in \{0,1\}$.

Définition 5.5.3 ***(Indistinguabilité)***

On dit qu'un schéma de chiffrement $\mathscr{C} = (\mathscr{D}_n, \mathscr{R}_n, \mathscr{G}, \mathscr{E}, \mathscr{D})$ a la proporiété d'indistinguabilité si pour tout adversaire $\mathscr{A}$ en temps polynomial et pour tout couple (m_0, m_1) de message de $\mathscr{D}_n$, on a:

$$(2) \qquad Adv^{\mathscr{A}} = |Prob_{e \leftarrow G(1^n)}[\mathscr{A}(\mathscr{E}(e, m_0)) = 1] - Prob_{e \leftarrow G(1^n)}[\mathscr{A}(\mathscr{E}(e, m_1))] = 1|$$

est une fonction négligeable.

Exemple 5.5.4 *Soit $\mathscr{E}$ la fonction de chiffrement d'un schéma $\mathscr{C}$. Soit $\mathscr{C}'$ le schéma de chiffrement presque identique à $\mathscr{C}$, mais dans le quel la fonction de chiffrement a été remplcé par la fonction $\mathscr{E}'$ donnée par:*

$$\mathscr{E}'(e,m) = \begin{cases} \mathscr{E}(e,m) \ si \ m \neq 0 \\ 0^n \ si \ m = 0^n \\ 1^n \ si \ m = 1 \end{cases}$$

Le schéma de chiffrement $\mathscr{C}'$ n'est pas sémantiquement sur, car avec la distribution de probablité donnée par:

$$Prob[m] = \begin{cases} \frac{1}{2} \ si \ m = 0^n ou 1^n \\ 0 \ sinon \end{cases}$$

Il est trés facile de construire un adversaire contre la sécurité sémantique de $\mathscr{C}'$ qui donne toujours la bonne réponse pour toute fonction d'information f.
Le schéma de chiffrement $\mathscr{C}'$ n'a pas non plus la propriété d'indistinguabilité entre le chiffré de 0^n et celui de 1^n.

Attaque à clair choisi (CPA):

Au cours de son exécution, l'algorithme contre un shéma de chiffrement peut accéder à un oracle de chiffrement qui lui répondra le chiffré des requétes qui lui sont soumises. On parle alors d'attaque à clair choisi, notée **CPA (Chosen Plaintext Attack)**. On notera par exemple IND-CPA la propriété d'indistinguabilité d'un schéma de chiffrement pour tout adversaire qui dispose d'un oracle de chiffrement.

Noter que dans un chiffrement à clé publique, la clé de chiffrement étant publique, l'adversaire peut toujours chiffrer les messages de son choix. La notion d'attaque à clair choisi n'est pertinente que pour le chiffrement symétrique, où l'adversaire, ne disposant pas de la clé de chiffrement, doit avoir recours à un oracle pour obtenir des couples (clair, cryptogramme) de son choix.

Attaque à texte chiffré choisi (CCA):

Dans une attaque à texte chiffré choisi, l'ennemi choisit un texte chiffré, l'envoie à la victime et recoit en retour le texte chiffré correspondant ou une partie de celui-ci. On dit que ce type d'attaque est adaptive (CCA2) si l'ennemi choisi un texte chiffré en fonction des précédents résultats de ses attaques.
On appelera Oracle de déchiffrement d'un PKCS qui, lorsqu'il reçoit un message chiffré non conforme à PKCS, renvoie un message d'erreur.Dans notre cas, on suppose que l'ennemi a acccès à un Oracle qui, pour chaque texte chiffré reçu, renvoie un message d'erreur chaaque fois qu'un message reçu n'est ps conforme.
Exemple CCA avec RSA:
L'ennemi choisit un texte chiffré c. Son obectif est de trouver $m \equiv c^d \pmod{n}$.
L'ennemi choisit des entiers s, et calcule:

$$c' \equiv cs^e \pmod{n}$$

L'ennemi envoie c' à l'oracle. Si l'Oracle ne renvoie pas de message d'erreur, l'ennemi sait que les deux premiers bytes de ms sont $0x00$ et $0x02$. En effet :

$$(c')^d \equiv c^d s^{ed} \equiv ms \pmod{n}$$

On peut chercher à quel intervalle appartient le message clair ms pour ensuite déterminer le message m.

Part II

Résultats

Chapitre 6

Cryptosystème basé sur les codes Reed Muller modifiés

Cheikh Thiecoumba GUEYE and ElHadji Modou MBOUP. Secure cryptographic scheme based on modified Reed Muller codes. In International Journal of Security and Its Applications, pages 55–64, 2013.

> *“ La théorie, c’est quand on sait tout et que rien ne fonctionne. La pratique, c’est quand tout fonctionne et que personne ne sait pourquoi.”*
>
> Albert Einstein

Sommaire

6.1 Introduction

Beaucoup de systèmes cryptographique basés sur la théorie du codage ont été proposés. Le schéma de Sidel'nikov [33] fait partir de ces systèmes. Ce schéma asymétrique remplace les codes Goppa avec des codes de Reed-Muller. Les codes de Reed Muller possédent des algorithmes de décodage trés efficaces (cf. [30]). C'est ainsi que la cryptanalyse présentée dans [21] a montré les limites de ce schéma. Dans cette cryptanalyse, L. Minder et A. Shokrolahi exploitent la filtration (3.3.13) et le mot de plus petit poids (3.3.14) du code Reed Muller. Dans ce chapitre, nous présentons une amélioration du cryptosystème de Sidel'nikov. Pour ce faire, nous allons décrire la cryptanalyse de L. Minder et A. Shokrolahi dans la section 1, dans la section 2 nous allons proposer un code de Reed Muller modifié qui consiste à ajouter une matrice aléatoire dans la matrice génératrice du code de Reed Muller. De plus, nous utilisons ce code de Reed Muller modifié pour obtenir une version améliorée de Sidelnikov dans la section 3. Enfin dans la section 4, nous prouvons que l'amélioration de Sidel'nikov, utilisant ce code Reed Muller modifié, resiste à la cryptanalyse L. Minder et A. Shokrolahi.
A titre d'exemple, la probabilité pour casser Sidel'nikov amélioré avec un code Reed Muller RM$(3,11)$ avec $n = 2058$, $k = 232$, $n' = 10$ et $t = 400$ (l'algorithme de décodage du code Reed Muller RM$(3,11)$ peut décoder jusqu'à 400 erreurs) est $\frac{1}{2^{2320}}$. Pour le meme cas, le facteur travail (WF) de Minder et Shokrolahi est suppérieur à 2^{88}.
Cependant, cette version améliorée de Sidel'nikov a été cassée en 2015 par A.Otami et al. dans [24] en utilisant une attaque structurelle.

6.2 Cryptanalyse de Sidel'nikov

Le cryptpsystème de Sidel'nikov présente des avantages sur son algorithme de décodage rapide et sa capacité de correction interssant (d/2 : où d est sa distance minimale). Cependant ce cryptosystème dispose des vulnérabilités remarquables à cause des propriétés (3.3.13) et (3.3.14) du code Reed Muller utilisé.
Cette vulnérabilité a été exploitée par L. Minder et A. Shokrolahi afin de mener une attaque structurelle sur le schéma de Sidel'nikov.

6.2.1 Objectif

Soient RM(r,m) un code de Reed Muller d'ordre r à m variables et σ une permutation quelconque. L'objectif de cette attaque est de trouver une permutation τ tel que

$$\mathrm{RM}(r,m)^{\sigma o \tau} = \mathrm{RM}(r,m)$$

6.2.2 Methodologie

La procédure de cette attaque est présentée dans les étapes suivantes:
Soient RM(r,m) un code de Reed Muller d'ordre r à m variables et σ une permutation quelconque.

- **Etape 1:** Trouver un mot de plus petit poids. Cette opération peut se faire en utilisant l'algorithme de Canteaut et Chaubaud dans [7].
- **Etape 2:** Cette deuxième étape consiste à construire le code de Reed Muller $\mathrm{RM}(r-1,m)$ à partir du mot de plus petit poids en utilisation la proposition 3.3.14.
- **Etape 3:** Répétons les deux étapes précédentes sur le code $\mathrm{RM}(r-1,m)$ pour construire le code $\mathrm{RM}(r-2,m)$ correspondant.
- **Etape 4:** Itérer ces étapes jusqu'à obtenir le code Reed Muller $\mathrm{RM}(1,m)$ d'ordre 1 à m variables.
- **Etape 5:** Trouver une permutation τ tel que

$$\mathrm{RM}(1,m)^{\sigma o\tau} = \mathrm{RM}(1,m)$$

Cette même permutation τ peut aussi s'appliquer au code Reed Muller d'ordre r à m variables. D'où

$$\mathrm{RM}(r,m)^{\sigma o\tau} = \mathrm{RM}(r,m)$$

6.2.3 Algorithme de l'attaque

Dans cette rubrique, nous exposons les algorithmes de la cryptanalyse sur Sidel'nikov. Cet algorithme prend en entrée un code Reed Muller $\mathscr{C}$ permuté avec une permutation σ et retourne une permutation τ

1. input : $\mathscr{C} = \mathrm{RM}(r,m)^{\sigma}$: un code de Reed Muller d'ordre permuté avec σ : permutation

2. output : τ: une permutation telle que $\sigma o\tau = Id$

3. Trouver tous les mots de code de $\mathscr{C}$ appartenant au sous code $\mathrm{RM}(r-1,m)^{\sigma}$ (code de Reed Muller d'ordre $r-1$ à m variables). Trouver une base de ces mots pour construire le sous code $\mathrm{RM}(r-1,m)^{\sigma}$.

4. Itérer ces étapes précédentes r fois en décrémentant jusqu'à obtenir $\mathrm{RM}(1,m)$.

5. Déterminer la permutation τ tel que $\mathrm{RM}(1,m)^{\sigma o\tau} = \mathrm{RM}(1,m)$. Alors, $\mathrm{RM}(r,m)^{\sigma o\tau} = \mathrm{RM}(r,m)$

6. Retourner τ.

6.2.4 Détermination d'un sous-code $\mathrm{RM}(r-1,m)^{\sigma} \subset \mathrm{RM}(r,m)^{\sigma}$

Cette opération consiste à trouver le mot de plus petit. D'après la proposition (3.3.14) le mot de plus petit poids est un produit des mots de code $\mathrm{RM}(1,m)^{\sigma}$.
Donc après avoir trouvé ce mot, nous le factorisons pour ressortir tous les mots de code de $\mathrm{RM}(1,m)^{\sigma}$.
Par suite construire une base, à partir de ces facteurs, de $\mathrm{RM}(r-1,m)^{\sigma}$.
D'où le sous-code $\mathrm{RM}(r-1,m)^{\sigma}$.
Pour plus de détailles, voire [21].

6.3 Le nouveau Code

6.3.1 Définition - Exemple - Proposition

Définition 6.3.1 *(**Code de Reed Muller modifié**)*
Soit C *un code de Reed Muller de longueur n, dimension k et distance minimale d d'ordre r à m variables noté* $\mathrm{RM}(r,m)$. *Soit* A *une matrice aléatoire d'ordre* $k \times n'$ *avec* $1 \leq n' \leq 10$. *Le code Reed Muller modifié, noté* $\mathrm{RM}^+(r,m)$, *est le code défini par la matrice génératrice* $\mathrm{G}^r = [\mathrm{G}_0^r|\mathrm{A}]$, *où* G_0^r *est une matrice génératrice du code de Reed Muller* $\mathrm{RM}(r,m)$.

Exemple 6.3.2 *La matrice génératrice* G_0^1 *du code de Reed Muller RM(1,3) est :*

$$\mathrm{G}_0^1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Posons

$$\mathrm{A}^1 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

la matrice aléatoire.
Le code de Reed Muller modifié d'ordre 1 à 3 variable, noté $\mathrm{RM}^+(1,3)$, *est le code defini par la matrice de génératrice suivante:*

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

Proposition 6.3.3 *(**Réparer la filtration**)*
En utilisant les codes de Reed Muller modifiés la probabilité pour obtenir

$$\mathrm{RM}^+(0,m) \subset \mathrm{RM}^+(1,m) \subset \ldots \subset \mathrm{RM}^+(m-1,m) \subset \mathrm{RM}^+(m,m)$$

est trop faible.
Démonstration : *Supposons que* G^i, *la matrice d'ordre* $k \times \mathrm{N}$, *est la matrice génératrice du code de Reed Muller modifié* $\mathrm{RM}^+(i,m)$ *et* G^{i+1}, *d'ordre k'*$\times$N, *celle du code de Reed modifié* $\mathrm{RM}^+(i+1,m)$ *(avec* $k = \sum_{j=0}^{i} \binom{m}{j}$, $k' = \sum_{j=0}^{i+1} \binom{m}{j}$*) et* $\mathrm{N} = n + n'$ *où* $n = 2^m$ *and* $1 \leq n' \leq 10$*).*
Soit $\mathrm{G}^i = [\mathrm{G}_0^i|\mathrm{A}_i]$ *où* G_0^i *est la matrice génaratrice du code de Reed Muller* $\mathrm{RM}(i,m)$ *et* A_i *la matrice aléatoire d'ordre* $k \times n'$.
Soit $\mathrm{G}^{i+1} = [\mathrm{G}_0^{i+1}|\mathrm{A}_{i+1}]$ *où* G_0^{i+1} *est la matrice génératrice du code de Reed Muller code* $\mathrm{RM}(i+1,m)$

et A_{i+1} *la matrice aléatoire d'ordre k'*$\times n'$.

Soit H^i, *une matrice d'ordre* $(\mathrm{N}-k)\times \mathrm{N}$, *la matrice de controle du code de Reed modifié* $\mathrm{RM}^+(i,m)$ *et* H^{i+1}, *d'ordre* $(\mathrm{N}-k')\times \mathrm{N}$, *celle du code Reed Muller modifié* $\mathrm{RM}^+(i+1,m)$.

Alors

$$\mathrm{H}^i = \begin{pmatrix} \mathrm{H}_0^i | \mathrm{N}_0^i \\ \mathrm{D}_0^i | \mathrm{D}_1^i \end{pmatrix}$$

où H_0^i *est la matrice de controle du code de Reed Muller* $\mathrm{RM}(i,m)$, *Les matrices aléatoires* D_0^i *d'ordre* $(\mathrm{N}-n)\times n$, *d'ordre* $\mathrm{N}-n\times n'$, N_0^i *d'ordre* $(n-k)\times n'$ *sont liées à la matrice aléatoire* A_i,

De plus

$$\mathrm{H}^{i+1} = \begin{pmatrix} \mathrm{H}_0^{i+1} | \mathrm{N}_0^{i+1} \\ \mathrm{D}_0^{i+1} | \mathrm{D}_1^{i+1} \end{pmatrix}$$

où H_0^{i+1} *est la matrice de controle du code de Reed Muller* $\mathrm{RM}(i+1,m)$, *Les matrices aléatoires* D_0^{i+1} *d'ordre* $(\mathrm{N}-n)\times n$, D_1^{i+1} *d'ordre* $(\mathrm{N}-n)\times n'$ *et* N_0^{i+1} *d'ordre* $(n-k)\times n'$ *sont liées à la matrice aléatoire* A_{i+1}.

- Posons x un élément du code de Reed Muller modifié $\mathrm{RM}^+(i,m)$, *x n'est pas un mot du code modifié* $\mathrm{RM}^+(i+1,m)$ *si et seulement si* $[\mathrm{H}^{i+1}]^t x \neq 0$.

Déterminons $[\mathrm{H}^{i+1}]^t x$*:*

Nous avons

$$\mathrm{H}^{i+1} = \begin{pmatrix} \mathrm{H}_0^{i+1} | \mathrm{N}_0^{i+1} \\ \mathrm{D}_0^{i+1} | \mathrm{D}_1^{i+1} \end{pmatrix}$$

Alors

$$[\mathrm{H}^{i+1}]^t x = \begin{pmatrix} \mathrm{H}_0^{i+1} | \mathrm{N}_0^{i+1} \\ \mathrm{D}_0^{i+1} | \mathrm{D}_1^{i+1} \end{pmatrix} ({}^t x)$$

Soit $x = [x_0^i | x_{\mathrm{A}_i}]$ *où* x_0^i *est le mot de code du code de Reed Muller* $\mathrm{RM}(i,m)$, *alors c'est aussi un mot de code du code Reed Muller* $\mathrm{RM}(i+1,m)$, *et* x_{A_i} *est obtenu par une combinaison linèaire des lignes des la matrice aléatoire* A_i.

On obtient

$$[\mathrm{H}^{i+1}]^t x = \begin{pmatrix} \mathrm{H}_0^{i+1} | \mathrm{N}_0^{i+1} \\ \mathrm{D}_0^{i+1} | \mathrm{D}_1^{i+1} \end{pmatrix} ({}^t [x_0^i | x_{\mathrm{A}_i}])$$

$$= \begin{pmatrix} [\mathrm{H}_0^{i+1}]^t x_0^i | [\mathrm{N}_0^{i+1}]^t x_{\mathrm{A}_i} \\ [\mathrm{D}_0^{i+1}]^t x_0^i | [\mathrm{D}_1^{i+1}]^t x_{\mathrm{A}_i} \end{pmatrix}$$

avec $x_0^i \in \mathrm{RM}(i+1,m)$,

alors $[\mathrm{H}_0^{i+1}]^t x_0^i = 0$

Donc

$$[\mathrm{H}^{i+1}]^t x = \begin{pmatrix} 0 | [\mathrm{N}_0^{i+1}]^t x_{\mathrm{A}_i} \\ [\mathrm{D}_0^{i+1}]^t x_0^i | [\mathrm{D}_1^{i+1}]^t x_{\mathrm{A}_i} \end{pmatrix}$$

Ainsi $[\mathrm{H}^{i+1}]^t x \neq 0$ *si et seulement si* $[\mathrm{N}_0^{i+1}]^t x_{\mathrm{A}_i} \neq 0$, $[\mathrm{D}_0^{i+1}]^t x_0^i \neq 0$ *or* $[\mathrm{D}_1^{i+1}]^t x_{\mathrm{A}_i} \neq 0$

Déterminons la probabilité pour obtenir

$$[\mathrm{N}_0^{i+1}]^t x_{\mathrm{A}_i} \neq 0, [\mathrm{D}_0^{i+1}]^t x_0^i \neq 0 \quad or \quad [\mathrm{D}_1^{i+1}]^t x_{\mathrm{A}_i} \neq 0$$

Les matrices N_0^{i+1}, D_0^{i+1} *and* D_1^{i+1} *sont donné par la matrice alèatoire* A_{i+1}.
La filtration présentée dans la proposition 3.3.13 ne peut pas s'appliquer si et seulement si $[\mathrm{N}_0^{i+1}]^t x_{\mathrm{A}_i} \neq 0$ *ou* $[\mathrm{D}_0^{i+1}]^t x_0^i \neq 0$ *ou* $[\mathrm{D}_1^{i+1}]^t x_{\mathrm{A}_i} \neq 0$.

Nous démontrons que

$([\mathrm{N}_0^{i+1}]^t x_{\mathrm{A}_i} = 0$ *and* $[\mathrm{D}_0^{i+1}]^t x_0^i = 0$ *and* $[\mathrm{D}_1^{i+1}]^t x_{\mathrm{A}_i} = 0) \Leftrightarrow (\mathrm{A}_{i+1} = (\frac{\mathrm{A}_i}{\mathrm{D}}))$, *où* D *est une matrice d'ordre* $(n-k) \times n'$:

1. *Supposons que* $[\mathrm{N}_0^{i+1}]^t x_{\mathrm{A}_i} = 0$ *et* $[\mathrm{D}_0^{i+1}]^t x_0^i = 0$ *et* $[\mathrm{D}_1^{i+1}]^t x_{\mathrm{A}_i} = 0$:
   alors on a
   $$[\mathrm{H}^{i+1}]^t x = \begin{pmatrix} \mathrm{H}_0^{i+1} | \mathrm{N}_0^{i+1} \\ \mathrm{D}_0^{i+1} | \mathrm{D}_1^{i+1} \end{pmatrix} ({}^t[x_0^i | x_{\mathrm{A}_i}])$$
   $$= \begin{pmatrix} [\mathrm{H}_0^{i+1}]^t x_0^i | [\mathrm{N}_0^{i+1}]^t x_{\mathrm{A}_i} \\ [\mathrm{D}_0^{i+1}]^t x_0^i | [\mathrm{D}_1^{i+1}]^t x_{\mathrm{A}_i} \end{pmatrix}$$

   Donc on a
   $$[\mathrm{H}^{i+1}]^t x = 0$$

   D'où $x \in \mathrm{RM}^+(i+1, m)$ *de plus* $x \in \mathrm{RM}^+(i, m)$
   $\Rightarrow \mathrm{RM}^+(i, m) \subset \mathrm{RM}^+(i+1, m)$
   So $\mathrm{A}_{i+1} = \frac{\mathrm{A}_i}{\mathrm{D}}$

2. *Supposons que* $\mathrm{A}_{i+1} = \frac{\mathrm{A}_i}{\mathrm{D}}$,
   on a $\mathrm{G}^{i+1} = [\mathrm{G}_0^{i+1} | \mathrm{A}_{i+1}]$ *avec* $\mathrm{A}_{i+1} = \frac{\mathrm{A}_i}{\mathrm{D}}$ *et* $\mathrm{G}_0^{i+1} = [\frac{\mathrm{G}_0^i}{\mathrm{D}_0}]$ (***cf. Exemple 3.3.12***).
   Alors $\mathrm{G}^{i+1} = [\frac{\mathrm{G}_0^i}{\mathrm{D}_0} | \frac{\mathrm{A}_i}{\mathrm{D}}]$, *on a* $\mathrm{G}^i = [\mathrm{G}_0^i | \mathrm{A}_i]$.
   Posons $\mathrm{D}_1 = [\mathrm{D}_0 | \mathrm{D}]$, *alors* $\mathrm{G}^{i+1} = [\frac{\mathrm{G}^i}{\mathrm{D}_1}]$
   Donc, on a $\mathrm{RM}^+(i, m) \subset \mathrm{RM}^+(i+1, m)$, *et cela implique que* $[\mathrm{N}_0^{i+1}]^t x_{\mathrm{A}_i} = 0$ *et* $[\mathrm{D}_0^{i+1}]^t x_0^i = 0$ *et* $[\mathrm{D}_1^{i+1}]^t x_{\mathrm{A}_i} = 0$

6.3.2 Calcul de la probabilité de succès de la filtration

Posons p la probabilité pour que $\mathrm{A}_{i+1} \neq \frac{\mathrm{A}_i}{\mathrm{D}}$ et q la probabilité de son complémentaire ($q = 1 - p$).
Alors le nombre de matrice aléatoire A_i est : $2^{n' \times k}$.
Ce qui entraine que $q = \frac{1}{2^{n' \times k}}$ and $p = 1 - \frac{1}{2^{n' \times k}}$.

Nous remarquons que la probabilité p est trop grande pour des parametres du code trop grands.

6.3.3 Exemple de calcul de probabilité $\mathrm{RM}(3, m)$

Pour un code Reed Muller modifié $\mathrm{RM}^+(3, m)$, le tableau suivant décrit les valeurs de la probabilité p and q.

n	k	n'	p	q
1034 (m=10)	176	10	$1-\frac{1}{2^{1760}} \simeq 1$	$\frac{1}{2^{1760}} \simeq 0$
2058 (m=11)	232	10	$1-\frac{1}{2^{2320}} \simeq 1$	$\frac{1}{2^{2320}} \simeq 0$
1033 (m=10)	176	9	$1-\frac{1}{2^{1584}} \simeq 1$	$\frac{1}{2^{1584}} \simeq 0$
2057 (m=11)	232	9	$1-\frac{1}{2^{2088}} \simeq 1$	$\frac{1}{2^{2088}} \simeq 0$
1032 (m=10)	176	8	$1-\frac{1}{2^{1408}} \simeq 1$	$\frac{1}{2^{1408}} \simeq 0$
2056 (m=11)	232	8	$1-\frac{1}{2^{1856}} \simeq 1$	$\frac{1}{2^{1856}} \simeq 0$

n: longueur du code de Reed Muller $\mathrm{RM}(r,m)$, k: la dimension du code de Reed Muller

Nous remarquons que la probilité $p \simeq 1$.
Alors la proposition 3.3.13 n'est pas applicable pour un code Reed Muller modifié.

Proposition 6.3.4 *Soient* G_r *l'automorphisme de groupe du code Reed Muller* $\mathrm{RM}(r,m)$ *et* G_r^+ *celui du code Reed Muller modifié* $\mathrm{RM}(r,m)^+$. *Nous avons* $\mathrm{G}_r \subseteq \mathrm{G}_r^+$
Démonstration : *Un code Reed Muller* $\mathrm{RM}(r,m)$ *peut être considéré comme un code Reed Muller modifié si la matrice aléatoire A est nulle.*

Remarque 6.3.5 .

- *Contrairement aux codes Reed Muller, pour les codes de Reed Muller modifiés, il existe plusieurs codes pour une longueur et une dimension donnée. Ce qui est un advantage pour une utilisation cryptographique.*
- *Le nombre d'erreurs corrigibles par les codes Reed Muller modifiés doit être très grande, ce qui rend les algorithmes généraux de décodage linéaires inefficace. Ce qui est aussi un avantage sur la sécurité du cryptosystème.*
- *La relation* $\mathrm{G}_r \subseteq \mathrm{G}_r^+$ *rend l'agorithme de séparation des supports de Nicola Sendrier [?] impraticable pour le code de Reed Muller modifié.*

6.4 Nouveau Cryptosystème

Ce cryptosystème est une amélioration du cryptosystème de Sidelnikov [33]. Il utilise le code de Reed modifié à la place des codes de Reed Muller.
Nous présentons dans la suite les algorithmes de ce nouveau cryptosystème.

6.4.1 Algorithme de génération de clé

1. Choisissons aléatoirement une matrice génératrice G_0^r d' un code Reed Muller $\mathrm{RM}(r,m)$.

2. Posons γ_0 l'algorithme de décodage du code Reed Muller RM(r, m) et t la capacité de correction de ce code.

3. Choisissons aléatoirement une matrice A_r d'ordre $k \times n'$ (où $1 \leq n' \leq 10$).

4. Construisons une matrice génératrice du code de Reed modifié $G^r = [G_0^r | A_r]$.

5. Choisissons deux matrices : S une matrice singulière d'ordre $k \times k$ et P une matrice de permutation d'ordre N × N où N = $n + n'$.

6. Calculons $G_p = SG^rP$.

Alors les clés du cryptosystème sont:

- La clé publique $P_k = (G_p, t)$
- La clé privée $S_k = (G^r, S, P, \gamma_0)$

7. Return P_k and S_k.

6.4.2 Algorithme de chiffrement

1. input : x, e, G_p //*x texte claire, e le mot d'erreur de poids t et* G_p *la clé publique*

2. output : c //*c le texte chiffré*

3. c=xG_p+e

4. return c.

6.4.3 Algorithme de déchiffrement

1. input : c //*Texte chiffré*

2. output : x //*Le texte clair*

3. $c' = cP^{-1}$

4. Posons $c' = [c_0^{'r} | c^{'}_{A^r_{n'}}]$ //$c_0^{'r}$ *est le texte chiffré avec un cryptosystème de Sidel'Nikov.*

5. $x' = \gamma_0(c_0^{'r})$ //*Decode* $c_0^{'r}$ *par l'algorithme* γ_0

6. $x = x'S^{-1}$

7. return x.

Remarque 6.4.1 *Ainsi, nous remarquons dans l'algorithme de déchiffrement que le décodage du mot se porte uniquement sur le mot de code du code de Reed Muller (*$x' = \gamma_0(c_0^{'r})$*). Donc l'ajout d'une matrice aléatoire n'entraine pas un problème de décodage.*

6.5 Sécurité du nouveau cryptosystème

6.5.1 Taille de la clé du nouveau cryptosystème

Ce nouveau cryptosystème est une variante du cryptosystème de Sidel'nikov. Dans cette sous section, nous faisons une étude comparative de la taille du nouveau cryptosystème avec celle de Sidel'nikov.Pour ce faire, nous prenons un code Reed Muller RM(r,m) et n' = 10 pour le nouveau code. Nous présentons dans le tableau suivant l'étude comparative :

	n	k	size of the key(bit)
Sidelnicov cryptosystem	1024 (m=10)	176	10240
New cryptosystem	1034 (m=10)	176	10340
Sidelnicov cryptosystem	2048 (m=11)	232	22528
New cryptosystem	2058 (m=11)	232	22638

n la longueur du code, $k = \sum_{i=0}^{r} C_i^m$ la dimension du code Reed Muller RM$(,r)$.

6.5.2 Attaque sur le nouveau Cryptosystème

La sécurité de base d'un système de chiffrement basé sur les codes repose sur les deux attaques suivantes:

- **Les attaques structurels**: Cette attaque consiste à retrouve le secret à partir des informations publiques.
- **L'attaque par décodage** : Cette attaque consiste à retrouver le texte clair à partir du chiffré et de la clé publique. Cette attaque est difficile à réaliser à cause du problème général de décodage.

Ainsi, la difficulté de l'attaque par décodage est liée au problème général de décodage. De meme que la difficulté des attaques structurelles ne dépend pas du problème classique de la théorique des codes mais principalement de la classe des codes et la transformation secrète utilisée.

Les attaques structurelles:

Ces attaques nous permettent à partir de la clé publique de retrouver la clé privée.
L'avantage de l'utilisation des codes de Reed Muller dans un cryptosystème est le fait qu'ils sont faiblement auto-dual et de meme resistent à l'algorithme de séparation des supports [31] car cet algorithme utilise le Hull (intersection du code avec son dual) qui est inefficace dans les codes de Reed Muller. Ces avantages sont aussi vérifiés pour les codes de Reed Muller modifiés.
Dans la proposition 6.3.3, nous avons prouvé que la filtration est quasiment impraticable dans le cryptosystème basé sur les codes Reed Muller modifiés. De plus, la probabilité pour trouver le sous-code RM$^+(r-1,m)$ du code Reed Muller modifié RM$^+(r,m)$ est trop faible. Ce qui rend la cryptanalyse de Sidel'nikov [21] quasiment impossible de réduire l'ordre r du code RM$^+(r-1,m)$

afin de retrouver le code $\mathrm{RM}^+(1,m)$.

Ainsi, étant donné que la méthode d'attaque structurelle utilisée dans [21] n'est plus efficace dans ce nouveau cryptosystème, nous allons étudier une attaque structurelle exploitant la matrice A ajoutée afin de revenir au système original pour appliquer la cryptanalyse de Sidel'nikov [21].

Cryptanalyse exploitant la matrice aléatoire A**:**
Nous avons deux méthode pour trouver la matrice aléatoire afin de retrouver le code original. Ces deux methods sont décrits comme suit:

- **Trouver directement la matrice aléatoire :** Cette technique consiste à trouver la matrice aléatoire dans la matrice génératrice publique du code modifié et puis appliquer l'attaque décrite dans [21]. Nous étudions la faisabilité de cette attaque dans la suite.
  Nous savons que A est une matrice aléatoire de dimension $n' \times k$ et la dimension de la matrice génératric du code modifié est $\mathrm{N} \times k$. Donc dans une matrice génératrice, il existe $\mathscr{C}_{n'}^{\mathrm{N}}$ matrice aléatoire. Alors le facteur de travail est :

$$\mathscr{W} = \mathscr{C}_{n'}^{\mathrm{N}}$$

  Dans [**?**], le cout de la cyptanalyse sur Sidel'nikov est :

$$\mathrm{C}ost\mathrm{Sh} = 2^{-\frac{m-r+1}{m-2r+1}.\frac{m^r}{r!}.log_2(1-2^{-r})-mr+r(r-1)+(m+r^2-4r+2)}$$

  Alors, le cout de la cryptanalyse exploitant la matrice aléatoire A, noté $Cost\mathrm{MG}$, est :

$$\mathrm{C}ost\mathrm{MG} = \mathrm{C}os\mathrm{Sh} + \mathscr{C}_{n'}^{\mathrm{N}}$$

Exemple 6.5.1 *Nous calculons le cout de la cryptanalyse avec des parametres du code différents dans les tableaux suivants:*

Table. 6.5.2 *Le code Reed Muller modifié* $\mathrm{RM}^+(3,10)$

Pameters	*WF*	*CostSh*	*costMG*
N=1035 (m=10), k=176 and n'=11	2^{85}	2^{16}	$\simeq 2^{85}$
N=1034 (m=10), k=176 and n'=10	2^{78}	2^{36}	$\simeq 2^{78}$

Table. 6.5.3 *Le code Reed Muller modifié* $\mathrm{RM}^+(3,11)$

Pameters	*WF*	*CostSh*	*costMG*
N=2059 (m=11), k=232 and n'=11	2^{96}	2^{19}	$\simeq 2^{96}$
N=2058 (m=11), k=232 and n'=10	2^{88}	2^{39}	$\simeq 2^{88}$
N=2057 (m=11), k=232 and n'=9	2^{80}	2^{39}	$\simeq 2^{80}$
N=2056 (m=11), k=232 and n'=8	2^{73}	2^{39}	$\simeq 2^{73}$

- **Recherche par position ajoutée en utilisant la dimension du Hull du code modifié :**
  Nous savons que l'"ajout d'une matrice aléatoire A est synonyme d'une reduction de la dimension du Hull (intersectiondu code et son dual). Mais cette dimension n'est pas toujours égal à $k-n'$. Nous calculons dans la suite la probabilité pour que la dimension du Hull soit égal à $k-n'$.
  Soit P cette probilité :

  – Si $m > 2r+1$, alors la probilité P est :

  $$\mathrm{P} = \frac{1}{2^{n' \times (k-n')}}$$

  – Si $m < 2r+1$, alors la probilité P est :

  $$\mathrm{P} = \frac{1}{2^{n' \times (n-k-n')}}$$

  Ainsi, nous avons remarqué que la dimension du Hull ne nous permet pas d'obtenir des informations sur la matrice aléatoire ajoutée.

- **Recherche par position ajoutée en utilisant la clé public G_p :**
  La probilité P pour trouver directement les positions ajoutées (matrice aléatoire A) dans la clé public G_p est:

  $$\mathrm{P}(r,m,n') = \frac{n!}{\mathscr{A}_{\mathrm{N}}^{n}}$$

  (avec $n = 2^m$ la longueur du code Reed Muller $\mathrm{RM}(r,m)$ et $\mathrm{N} = n + n'$ la longueur du code de Reed Muller modifié $\mathrm{RM}^{+}(r,m)$.

  Exemple 6.5.4

  $$\mathrm{P}(3,11,10) = \frac{2048!}{\mathscr{A}_{2058}^{2048}} = 2^{-90}$$

  Remarque 6.5.5 *Pour resister à l'attaque de Sidel'nikov proposé dans [21], nous devons choisir un code Reed Muller* $\mathrm{RM}^{+}(r,m)$ *avec* $(n' \geq 10, m = 10 \quad et \quad r = 3)$ *ou* $(n' \geq 10, m = 11 \quad et \quad r = 3)$

- **Attaque utilisant le carré de la matrice publique G_p :**
  Cette attaque a été menée en 2015 par A.Otami et al. dans [24] sur cette version améliorée de Sidel'nikov. Cette cryptanalyse repose sur le calcul des carrés du code public. La proporiété particulière des codes de Reed-Muller, qui consiste à définir des polynômes binaires multivariés, permet de déterminer la dimension des codes carrés et ensuite de récupérer complètement en temps polynomial les positions secrètes des colonnes aléatoires ajoutées. Cette attaque rend notre amélioration obsolète.

L'attaque par décodage

Etant donné que les nouveaux codes peuvent décoder beaucoup plus d'erreurs (algorithme de décodage des codes de Reed Muller), avec une forte probabilité, que même la distance minimale du code. Ainsi l'attaque par décodage directe [[6], [4]] avec utilisation de l'agorithme [[7]] est impratricable.

Chapitre 7

Implémentation efficace du schéma hybride (KEM-DEM) basé sur les codes

“ Rien dans la nature n’est aléatoire. Le hasard n’apparaît qu’à travers l’insuffisance de nos connaissances. ”

B. Spinoza

Pierre-Louis Cayrel, Cheikh Thiecoumba Gueye, El Hadji Modou Mboup, Ousmane Ndiaye, and Edoardo Persichetti. Efficient implementation of hybrid encryption from coding theory. In Codes, Cryptology and Information Security - Second International Conference, C2SI 2017, Rabat, Morocco, April 10-12, 2017, Proceedings - In Honor of Claude Carlet, pages 254–264, 2017.

Sommaire

7.1 Introduction :

Ce chapitre est une illustration de la deuxième partie de notre thèse. Dans ce chapitre, nous avons étudié l'amélioration d'un schéma de chiffrement hyride basé sur les codes proposé dans [25] et puis implémenté en C ce schéma. En effet, un chiffrement hybride est un schéma de chiffrement utilisant à la fois un chiffrement symétrique et chiffrement asymétrique. Le principe d'un schéma hybride consiste à générer dans une prémière étape une clé aléatoire pour construire la clé symétrique. Cette clé symétrique sera utilisée pour le chiffrement des données. De plus, les deux clés (clé plubique et privée) du schéma asymétrique seront utilisées pour l'échange de la clé symétrique. Le concept de chiffrement hybride a été introduit pour la première fois par Cramer et Shoup dans [10].

Dans [8], nous proposons un schéma hybride qui utilise un cryptosystème Niederreiter comme schéma asymétrique et AES comme schéma symétrique. Ce schéma hybride est essentiellement composé de deux parties :

- **Key Encapsulation Mechanism (KEM)**, l'étape de génération du mot aléatoire et de la construction de la clé symétrique,
- **Data Encapsulation Mechanism (DEM)**, l'étape qui consiste à utiliser la clé symétrique pour chiffrer et déchiffrer les données.

Dans ce qui suit, nous orgniserons ce chapitre en cinq sections. Dans la première section, nous donnerons les rappels des outils utilisés notamment le chiffrement de Niederreiter, le chiffrement hybdride, la fonction de dérivation clé (KDF). Au niveau de la deuxième section, nous présenterons le chiffrement hybride basé sur les codes avec une description des algorithmes et aussi de la permutation aléatoire ajoutée dans l'algorithme de déchiffrement dans le cas où le décodage ne réussit pas. La description de l'implémentation du schéma hybride sera faite dans la troisième section. Ensuite, au niveau de la quatrième section nous ferons une étude de la sécurité du schéma hybride et aussi l'analyse de la performance de ce schéma en faisant une étude comparative entre ce schéma hybbride et celui de Niederreiter dans une premère étape et puis celui de l'hybride de RSA. Enfin, nous conclurons dans la cinquième section.
Comme plate-forme matérielle, nous utiliserons un ordinateur à 1.80 GHz, Intel Core i5 CPU et 4 Go de RAM et 64 bit OS. Le programme sera exécuté sur du Debian/Linux 3.5.2 et compilé avec gcc 4.7.

7.2 Préliminaires :

7.2.1 Fonction de dérivation de clé:

Définition 7.2.1 *Une fonction de dérivation de clé (KDF) est une fonction qui prend en entrée une chaîne x de longueur arbitraire et un entier $\ell > 0$ et émet une chaîne de bits de longueur fixe ℓ. Les*

fonctions de dérivation de clés sont également utilisées dans les applications de génération des clés de mots de passe ou des phrases de passe secrets.

Exemple 7.2.2 *Nous avons la KDF PBKDF2 (spécifiés dans la RFC 2,898) qui utilise la fonction de hachage SHA-2 avec un nombre élevé d'itération (souvent 1000 ou plus) pour augmenter le niveau de sécurité.*
NIST exige que la taille des clés générée par ces KDF soit au moins 128 bits.

Définition 7.2.3 ***Message Authentification Code (MAC)*** *Un MAC est un algorithme qui nous permet d'obtenir une empreinte. Cette empreinte nous permettra d'identifier le message. Un MAC est défini par une fonction noté* $\mathrm{E}v$ *qui prend en entrée une clé* K *de longueur* ℓ_{MAC} *et une chaine arbitraire* T *et retourne une chaine* τ *de longueur* ℓ_{TAG} *qui est un tag à ajouter au message.*

7.2.2 Schéma de chiffrement Hybride:

Key Encapsulation Mechanism (KEM):

Un KEM est essentiellement un système de chiffrement à clé publique, à l'exception du fait que l'algorithme de chiffrement prend aucune entrée en dehors de la clé publique et retourne une paire (K, ψ_0) où K est obtenue à partir d'un mot e généré aléatoirement et une fonction dérivation de clé appelée KDF i.e. $(\mathrm{KDF}(e) = \mathrm{K})$ et ψ_0 est le chiffré du mot e à partir d'un algorithme asymétrique $(\mathrm{D}ec_{sk}(\psi_0) = e)$. Ainsi, un KEM est constitué de trois algorithmes suivants.

Table. 7.2.4 *Key Encapsulation Mechanism.*

Key Generation	*A probabilistic key generation algorithm that takes as input a security parameter* 1^λ *and outputs a public key* pk *and a private key* sk
Encryption	*A probabilistic encryption algorithm that receives as input a public key* pk *and returns a key/ciphertext pair* (K, ψ_0).
Decryption	*A deterministic decryption algorithm that receives as input a private key* sk *and a ciphertext* ψ_0 *and outputs either a key* K *or the failure symbol* $\perp$.

Définition 7.2.5 ***(Attaque de CCA2 sur le KEM)***
Le jeu d'attaque CCA2 (Adaptative Chose-Ciphertext Attack) sur un KEM procéde de la manière suivante:

1. *Requete d'une clé publique vers un oracle de génération de clé pour obtenir* pk.
2. *Soumettons une chaine* ψ_0 *à un oracle de déchiffrement. L'oracle répondra par* $\mathrm{D}ec_{sk}^{\mathrm{KEM}}(\psi_0)$.

3. *Requete vers un oracle de chiffrement qui utilise* $\mathrm{Enc}_{pk}^{\mathrm{KEM}}$ *pour générer une paire* $(\widehat{\mathrm{K}}, \widehat{\psi_0})$. *Cet oracle choisit aléatoirement* $b \in \{0,1\}$ *et donne le challange* $(\mathrm{K}^*; \widehat{\psi_0})$ *où* $\mathrm{K}^* = \widehat{\mathrm{K}}$ *si* $b = 1$ *sinon* K^* *est une chaine aléatoire de longueur* ℓ_{K}.

4. *En mettant en mémoire les requêtes de déchiffrement, si le cryptogramme soumise est* ψ_0^*, *alors l'oracle retourne* $\perp$

5. *Retourner* $b^* \in \{0,1\}$.

L'adversaire a réussi si $b^* = b$. *Plus précisément, nous définissons un avantage noté* A *contre KEM :*

$$\mathrm{Ad}v_{\mathrm{KEM}}(\mathrm{A}, \lambda) = |\mathrm{Pr}[b^* = b] - \frac{1}{2}| \tag{1}$$

Nous savons qu'un KEM est sur si l'avantage $\mathrm{Ad}v_{\mathrm{KEM}}(\mathrm{A}, \lambda)$ *pour tout adversaire* A *dans le modèle oracle défini ci-dessus est négligeable.*

Data Encapsulation Mechanism (DEM):

Le mécanisme d'encapsulation de données (DEM) est un schéma de chiffrement symétrique. Il utilise la clé K générée dans le KEM comme clé de chiffrement et déchiffrement. Nous présentons en bref dans les deux algorithmes suivants le fonctionnement d'un DEM:

Table. 7.2.6 *Data Encapsulation Mechanism.*

Encryption	*Un algorithme de chiffrement détermiste qui prend en entrée une clé* K *et un texte clair* φ *et puis retourner un chiffré* ψ_1
Decryption	*Un algorithme de déchiffrement détermiste qui prend en entrée une clé* K *et un chiffré* ψ_1 *et puis retourner un texte claire* φ *ou un texte* $\perp$.

7.2.3 Les fonctions de hachage

Définition 7.2.7 ***(fonction de hachage)*** *Une fonction de hachage H est une application facilement calculable qui transforme une chaine binaire de taille quelconque t en une chaine binaire de taille fixe n appelée empreinte de hachage.*

Définition 7.2.8 ***(Collision)*** *On parle de collision entre* x_1 *et* x_2 *lorsque* $x_1 \neq x_2$ *et* $\mathrm{H}(x_1) = \mathrm{H}(x_2)$

Définition 7.2.9 ***(Préimage)*** *Si y est tel que* $y = \mathrm{H}(x)$*; alors x est appelé préimage de y.*

Définition 7.2.10 ***(Paradoxe des anniversaires)*** *Soient la fonction de hachage* $\mathrm{H} : \{0,1\}^t \rightarrow \{0,1\}^m$ *avec* $t > m$ *et k messages* $x_{i1 \leq i \leq k}$ *aléatoires* ($k \ll n$). *On pose:* $n = 2^m = |\{0,1\}^m|$. *L'attaque des anniversionnaires consiste à trouver la probabilité d'obtenir une collision à partir des messages* x_i.

Cette probabilité pour obtenir au moins une collision, notée p_k, est:

$$p_k = 1 - \prod_{i=1}^{k-1} (1 - \frac{i}{n}$$

Les principales fonctions de hachage:

- **Fonctions de hachage MD5:** Cette fonction de compression a été proposée par Rivest en 1991 dans [29].
- **Fonctions de hachage SHA 1:** Cette fonction a été publiée dans FIPS PUB 180-1 de NIST (National Institute of Standards and Technology) en 1995 dans [23].
- **Fonctions de hachage SHA 256:** Cette fonction a été publiée dans FIPS PUB 180-2 en 2000 [1].
- **Fonctions de hachage SHA 512** [2].

Sécurité des fonctions de hachage:

Fonction	Empreinte	Complexité requise	Résistance aux collisions	Comp. attaque
MD5	128 bits	$\mathcal{O}(2^{64})$	Cassé	$\mathcal{O}(2^{30})$
SHA-1	160 bits	$\mathcal{O}(2^{80})$	Cassé	$\mathcal{O}(2^{63})$
SHA-256	256 bits	$\mathcal{O}(2^{128})$	S$\hat{u}$r	
SHA-512	512 bits	$\mathcal{O}(2^{256})$	S$\hat{u}$r	

7.3 Schéma hybride basé sur les codes (Niederreiter):

Dans [25], l'auteur présente un schéma de chiffrement Hybride basé sur le cryptosystème de Niederreiter. Ce schème hybride est composé d'un chiffrement asymétrique pour le KEM, d'un chiffrement symétrique pour le DEM et aussi une fonction de hachage pour la construction et la protection de la clé. Dans [8], nous proposons une amélioration du schéma de E. Persichetti ([25]). Cette amélioration consiste à ajouter une permutation aléatoire dans le chiffré (ψ_0) du mot aléatoire e dans le cas où le décodage échoue. Cette permutation permet au schéma hybride de resister contre l'attaque de Bernstein et al proposée dans [5] et aussi contre les attaques critiques (CCA2, réaction attack,...) proposées dans [9]. En effet, les composantes du schéma de chiffrement hybride étudié sont:

- Le schéma de chiffrement asymétrique de Niederreiter est utilisé au niveau du KEM. Il sert aussi à protéger la clé symétrique K.
- Le schéma de chiffrement symétrique AES128-CBC est utilisé dans le DEM pour le chiffrement et le déchiffrement des données.
- La fonction de hachage SHA3 est utilisée comme fontion de dérivation de clé (KDF). De plus, elle est aussi utilisée dans la partie HMAC pour gérer l'intégrité du messagee.

Dans ce qui suit, nous présenterons la méthologie de ce schéma ainsi que les différents algorithmes de ce schéma hybride.

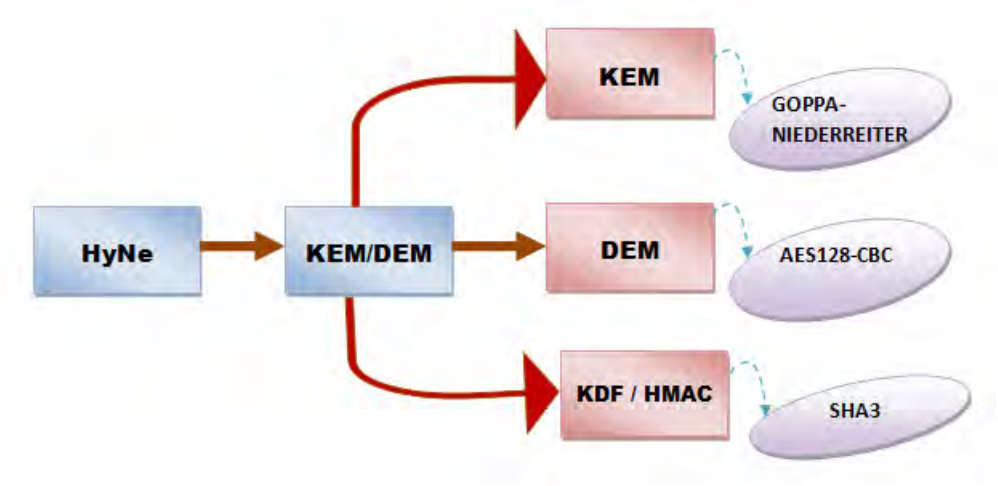


Figure 7.1 – Procédure d'un schéma Hybride de Niederreiter

7.3.1 Méthologie

Nous présentons le schéma suivant qui décrit la procédure de chiffrement hybride proposé dans [8]. Dans ce schéma, nous présentons la procédure de chiffrement hybride avec une description des différentes composantes du schéma notamment KEM, DEM, le controle d'intégrité avec une fonction HMAC et de plus la permutation du chiffré ajoutée au niveau du déchiffrement dans le cas où le décodage ne reussit pas. A cet effet, les éléments utilisés sont:

- $(\triangle, \mathrm{M})$ sont respectivement la clé privée et la clé publique du schéma asymétrique de Niederreiter. Pour construire la clé publique nous calculons $\mathrm{H} = (\mathrm{M}|\mathrm{I}_{n-k})$.
- φ est le texte claire à chiffrer et e le mot aléatoirement généré qui sert à construire la clé symétrique K avec la fonction de dérivaion de clé $(\mathrm{K} := \mathrm{KDF}(e))$. Ce mot e sera chiffré par Niederreiter avec la relation $(\psi_0 = \mathrm{H}e^{\mathrm{T}})$. Alors ψ_0 est le chiffré du mot aléatoire que nous allons envoyer au propriètaire de la clé publique (M). A la reception le mot e sera retrouvé à l'aide de la clé privée $\triangle$.
- σ est la permutation appliquée au chiffré du mot e $(\sigma(\psi_0))$ pour parer aux attaques critiques.
- $(\mathrm{K}_1||\mathrm{K}_2) := \mathrm{K}$ est la clé symétrique K scindée en deux parties K_1 et K_2 où K_1 est utilisée comme clé symétrique et K_2 dans le HMAC.
- τ et τ' sont respectivement le test d'intégrité (HMAC) calculé avant le chiffrement et après le déchiffrement.
- $\mathrm{AES}_{\mathrm{E}}nc$ et $\mathrm{AES}_{\mathrm{D}}ec$ sont les algorithmes de chiffrement et déchiffrement de l'AES utilisés dans le DEM.
- ψ' est le chiffré des données à l'aide de $\mathrm{AES}_{\mathrm{E}}nc$.

7.3.2 Algorithmes

Algorithme de génération de clé

En se basant sur la procédure de génération de clé présentée dans 5.3.2, nous choisissons en entrée un code de Goppa avec sa matrice de parité H = (M|I$_{n-k}$). En sortie, nous retournons la clé publique M et la clé privée Δ.

Algorithme de chiffrement

Input: M $\in$ K$_{pub}$: la clé publique et φ : le texte clair .
Output: ψ : le texte chiffré .

Choisir aléatoirement un mot $e \in \mathcal{W}_{n,t}$. Posons H := (M|I$_{n-k}$), alors calculer $\psi_0 := He^T$ et la clé symétrique K := KDF($e, \ell_{\text{DEM}} + \ell_{\text{MAC}}$). Scinder la clé symétrique comme K := (K$_1$||K$_2$). Calculer $\psi' := Enc^{\text{DEM}}_{K_1}(\varphi)$, et Posons T := ψ' et puis calculer $\tau := E\nu(K_2, T)$. Return $\psi := (\psi_0||\psi'||\tau)$.

Algorithme de déchiffrement

Input: Δ : la clé privée et ψ : le texte chiffré.
Output: φ : le texte clair.

Scinder le texte chiffré en trois composante pour obtenir $\psi := (\psi_0||\psi'||\tau)$, et puis calculer $e := Decode_{\Delta}(\psi_0)$: si le docodage reussit, alors calculer K := KDF($e, \ell_{\text{DEM}} + \ell_{\text{MAC}}$), sinon choisissez une permutation aléatoire σ et puis calculer K := KDF($\sigma(\psi_0), \ell_{\text{DEM}} + \ell_{\text{MAC}}$). Scinder la clé symétrique comme K := (K$_1$||K$_2$). Posons T = ψ', calculer $\tau' := E\nu(K_2, T)$: si ($\tau \neq \tau'$) echec de la vérification et return $\perp$. Sinon, calculer $\varphi := Dec^{\text{DEM}}_{K_1}(\psi')$. Return φ.

Remarque 7.3.1 *Nous avons noté l'utilisation de la permutation σ au niveau du déchiffrement. Cette permutation sert à masquer le texte chiffré ψ_0 et ensuite calculer la clé symétrique K à l'aide de cette permutation. Le but de cette permutation est de permettre à notre schéma hybride de résister contre les attaques critique ([9]). De plus, l'utilisation de cette permutation nous permet d'éviter l'attaque de Bernstein et al. dans [5]. Cette attaque sera présentée dans la suite.*

7.4 Implémentation :

7.4.1 Description :

Dans cette section, nous présentons une implémentation rapide du schéma hybride décrit dans la section précédente. Comme plateforme matèriel, nous utilisons un ordinateur de processeur Intel core i3 CPU 1.80 GHz et 4 GB RAM dans un système d'exploitation 64 bits. L'environnement d'exécution du code source se fait dans un compilateur gcc 4.7 avec un OS Debian/Linux de noyau 3.5.2.

7.4.2 Présentation des fonctions:

Nous présentons les différentes fonctions utilisées dans cette implementation.

- **Génération des clés :** L'algorithme de génération des clés utilisé dans cette étape est celui de Niederreiter. Nous avons la fonction ***keypair(***s_k, p_k***)*** de génération de clé publique et privée. Dans cette fonction, nous utilisons le polynome de Goppa pour construire les deux clés. Ces deux clés (s_k, p_k) seront stockées respectivement dans $s_k.pem$ et $p_k.pem$. La clé symétrique K sera calculée dans la procédure de chiffrement et déchiffrement (c'est la procédure KEM/-DEM).

- **Chiffrement Hybride :** Cette étape de chiffrement du schéma hybride est composée du chiffrement de Niederreiter pour le KEM, du chiffrement symétrique de AES128-CBC pour le DEM et aussi l'utilsation de la fonction de hachage SHA3 pour la fonction KDF et HMAC. Nous présentons dans la suite les fonctions utilisées dans cette partie :

  - ***Mat-from-pk()***: Cette fonction transforme la matrice public H sous forme systématique. Cette forme systématique joue un role trés important dans la compléxité de l'algorithme.

  - ***sponge()***: Cette fonction est utilisée dans la fonction de dérrivation de clé (KDF). De plus elle est aussi utilisée comme une fonction HMAC. Nous avons utilisé SHA3 pour cette fonction à cause de la sécurité et de la rapidité de SHA3.

  - ***AES128-CBC-encrypt-DEM()***: Cette fonction AES128-CBC est utilisée dans le chiffrement symétrique des données. Elle représente la partie DEM du chiffrement hybride.

  - ***encrypt-Nied()***: Cette fonction représente le chiffrement de Niederreiter. Elle est utilisée pour chiffrer le mot aléatoire e.

  - ***Encrypt-HyNe()*** Dans cette fonction, nous avons implémenté le chiffrement hybride. Cette fonction prend en entrant le texte claire et la clé publique puis retourne le texte chiffré (ψ:=($\psi_0||\psi'||\tau$):=Encrypt-HyNe(pk.pem,plainText)).

- **Déchiffrement Hybride :** Cette étape de déchiffrement du schéma hybride est composée du déchiffrement de Niederreiter pour le KEM, du déchiffrement symétrique de AES128-CBC pour le DEM et aussi l'utilsation de la fonction de hachage SHA3 pour la fonction KDF et HMAC. De la meme manière que le chiffrement hybride, le déchiffrement utilise les memes fonctions dans le sens du déchiffrement.
  D'où l'utilisation de la fonction ***decrypt-HyNe()*** comme fonction de déchiffrement du schéma hybride. Elle prend en entrée la clé privée (sk.pem) et le texte chiffré (ψ) puis retourne le texte claire (φ:=decrypt-HyNe(sk.pem,ψ)).

7.5 Sécurité et Performance

7.5.1 Sécurité du KEM/DEM

La sécurité IND-CCA2 du schéma de chiffrement hybride KEM-DEM, introduit par Cramer et Shoup dans [10], dépend de la sécurité du KEM et du DEM.Dans [25], Persichetti détaille une preuve de sécurité complète qui suit de près le paradigme; cependant, la preuve requise pour introduire une modification dans le schéma afin de garantir l'intégrité du simulateur. En fait, étant donné qu'il n'est pas possible de décider à priori si un mot donné est décodable ou non, il faut que le déchiffrement du KEM produise toujours quelque chose. Une suggestion naturelle était d'utiliser une fonction pseudo-aléatoire du texte chiffré, i.e. KDF(ψ_0). Malheureusement, ce choix rend le système vulnérable à une attaque simple, que nous décrirons ci-dessous.

Malléabilité

Malleabilité est une propriété de certains algorithmes cryptographiques. Un algorithme de chiffrement est malléable s'il est possible pour un adversaire de transformer un texte chiffré en un autre texte chiffré décrypté sur un texte en clair choisi. C'est-à-dire, étant donné un cryptage du texte en clair, il est possible de générer un autre texte chiffré comme $f(e)$, pour une fonction connue f, sans nécessairement connaître ou apprendre e.
Le schéma de Niederreiter classique est: $\psi_0 := \mathrm{H}e^{\mathrm{T}}$ avec $wt(e) := t$, un adversaire peut choisir aléatoirement i^{eme} colonne de H noté H[i] et puis soumet le chiffré $\psi_0' := \psi_0 \oplus \mathrm{H}[i]$ à l'oracle de déchiffrement. Le déchiffrement sera reussi si nous avons "1" dans la i^{eme} bit du message e. Soit e_1 un vecteur de longueur n avec seulement le i^{eme} bit différent de zero.
Alors l'oracle retourne le vecteur e' de poids $t-1$ telque :

$$\psi_0' = \mathrm{H}e'^{\mathrm{T}} = \psi_0 \oplus \mathrm{H}[i] = \mathrm{H}e^{\mathrm{T}} \oplus \mathrm{H}e_1^{\mathrm{T}} = \mathrm{H}(e \oplus e_1)^{\mathrm{T}}$$

Donc

$$e = e' \oplus e_1$$

La probilité de succée pour cette attaque est:

$$\mathrm{P}r = \mathrm{P}(e[i] = 1) = \frac{\binom{t}{1}}{\binom{n}{1}} = \frac{t}{n}$$

pour $n = 1024$ et $t = 50$, $\mathrm{P}r = 0.05$
Pour $n = 2048$ et $t = 81$, $\mathrm{P}r = 0.04$ i.e. reussi au plus 25 times.
Ainsi le schéma de Niederreiter est malléable.

Attaque de Beintein

Bernstein et al. dans [5] ont réussi à appliquer une technique d'attaque sur le schéma de chiffrement hybride. Cette attaque sera décrite ci-après.
Nous avons comme challenge chiffré texte (ψ_0, ψ', τ).
L'adversaire peut calculer le texte chiffré $(\psi_0'', \psi'', \tau'')$, où:

- $\psi_0'' = \psi_0 \oplus H[i] \oplus H[j]$, pour $i \neq j$
- ψ'' est la chaine de caractère obtenu aléatoirement
- $\tau'' = E\nu(K_2'', \psi'')$ où $K'' = (K_1'', K_2'') = KDF(\psi_0'', \ell_{DEM} + \ell_{M}AC)$

De la meme manière que précédemment, nous $\psi_0'' = H(e'')^T$ avec e'' est le vecteur e dont deux position i et j sont modifiées. Donc, si e'' a un poids inférieur ou égal à t, le décodage va réussir, et le KEM retounera $K = KDF(e'', m + \ell_{MAC})$. Ce qui entraine un probléme de déchiffrement car le hmac (controle d'intégrité) va échouer.
D'autre part, si le poid de e'' est suppérieur à t, alors le decodage va échouer et le KEM retournera exactement K''; dans ce cas le hmac (tag) va reussir. Ainsi le schema est malléable et l'adversaire pourra retrouver le secret e, et puis le text clair m. L'attaque est possible car le choix de la fonction pseudorandom rend trop prévisible pour un attaquant de calculer la chaîne de sortie par le KEM en cas de défaillance de décodage. Avec notre modification, le KEM calcule le KDF en utilisant une permutation aléatoire de ψ_0, ce qui contredit cette simple attaque.

7.5.2 Performance et comparaions

HyNE et PKCS de Niederreiter

Dans cette sous-section, nous présentons une étude comparative de l'Hybride de Niederreiter (HyNE) et l'originale du PKCS de Niederreiter. Ces deux algorithmes sont testés dans un ordinateur de processeur Intel core i3 CPU 1.80 GHz, 4 GB RAM avec un OS Debian de 64 bit. Cette comparaison est basée sur les temps d'éxecution des différents algorithmes (génération de clé, chiffrement des données et déchiffrement des données) en fonction de la taille de la clé symétrique (128, 256 et 512) et aussi des parametres du code de Goppa.

Les temps d'éxecution des algorithmes sont donnés dans le tableau **??** pour un code de Goppa de longueur $n = 2^m$ avec t la capacité de correction. Nous calculons la vitesse d'éxecution de l'algorithme de manière indépendant du processeur en utilisant la relation (nombre de cycles du processeur pour executer un byte en une seconde * Fréquence du processeur (Hz) / Taille du fichier à tester (bytes)).

HyNE et Hybride de RSA

Dans un meme environnement, nous présentons une étude comparative des temps d'éxecution de l'Hybride de Niederreiter (HyNE) et l'Hybride de RSA présenté dans [13] pour des exposant de RSA différents. Le tableau 7.1 suivant contient les résultats de cette comparaison.

7.5.3 Discussion

Dans notre compaison, nous avons évalué, en fonction de la valeur des parametres des schémas, les vitesses d'execution de l'Hybride de RSA obtenu dans [13].

Table 7.1 – Running Times of HyNe and Niederreiter PKCS

Scheme (m,t) **or** **Scheme** $(m,t,\ell_{\mathrm{DEM}})$	**Cycles**			**Security (bit ops.)**
	KeyGen	**Encrypt**	**Decrypt**	
Nied(10,28)	142060394	111014	3053326	60
Nied(11,32)	384089214	170121	3224981	88
Nied(12,48)	2613209726	322965	6095783	128
HyNe(10,28,128)	142060394	128909	3148128	60
HyNe(11,32,128)	384089214	176429	3575968	88
HyNe(12,48,128)	2613209726	349012	6196288	128
HyNe(10,28,256)	142060394	132245	3213051	60
HyNe(11,32,256)	384089214	198890	3876063	88
HyNe(12,48,256)	2613209726	434950	7969412	128
HyNe(10,28,512)	142060394	140306	3465572	60
HyNe(11,32,512)	384089214	201690	4993159	88
HyNe(12,48,512)	2613209726	492826	8860359	128

Table 7.2 – Running Times of Hybrid RSA

Exponents (bits)	**Keygen**	**Cycles (Enc)**	**Cycles (Dec)**	**Security (bit ops.)**
1024	986400	482400	463200	80
2048	4735200	2402400	2320800	112

A cet effet, nous constatons les suggestions suivantes:

- La taille des clés est plus petite chez Hybride de RSA qu'à l'Hbride de Niederreiter,
- Le chiffrement est trois fois plus rapide dans l'Hybride de Niederreiter (HyNE) qu'à l'Hybride de RSA.
- Le temps d'éxecution dans le dechiffrement est quasiment égal pour les deux algorithmes.

De plus, la comparaison faite dans le premier tableau entre l'Hybride de Niederreiter et l'original de Niederreiter (non IND-CCA2) nous donne un ratio faible de $(1.16, 1.19, 1.26)$ pour des textes respectifs de tailles $\ell_{\text{DEM}} = (128, 256, 512)$.

Chapitre 8

Protocole d'authentication forte basé sur les codes

“ L'authenticité est une pratique quotidienne qui consiste à laisser tomber celui que nous imaginons devoir être et d'embrasser celui que nous sommes vraiment. ”

Brene Brown

Sommaire

8.1 Introduction :

L'authentification est un processus important de la sécurité pour la maitrise des risques liées au controle d'accès aux systèmes d'Information. L'augmentation des besoins de connexion en position de mobilité (ordinateur portable, smart-phones, etc.) exige à traiter de manière très fine la sécurité de la connexion d'une personne à une application. Selon le moyen de connexion (son poste fixe professionnel, son ordinateur portable, son smart-phone, etc.), le risque à gérer est différent entrainant la gestion de plusieurs niveaux d'authentification pour l'accès d'une meme personne à une application. A cet effet, plusieurs types d'authentification sont créés.

Dans cette partie de notre thése, une étude théorique et pratique d'un mécanisme d'authentification basé sur les codes sera faite. En effet, ce mécanisme d'authentification utilise le principe KEM-DEM étudié dans le chapitre précédant (cf. [8]). Le principe de ce mécanisme consiste à générer dans une première étape un challenge pour un client qui demande une connexion. Le client utilise sa clé secrete pour le chiffrement de ce challenge. L'algorithme de chiffrement ainsi utilisé est un schéma symétrique. La seconde étape de l'algorithme du mécanisme d'authentification est de prouver au niveau du serveur que le chiffré du challenge est bel et bien envoyé par le client. Ce controle du paramétre d'indentification (du challenge) est fait en utilisant le schéma de chiffrement asymétrique basé sur les codes Niederreiter [22], le schéma symétrique AES128CBC et la fonction de hachage SHA3.

Dans ce qui suit, nous organiserons ce chapitre en quatre sections. Dans la première section, nous parlerons des principes d'authentification. Au niveau de la deuxième section, nous présenterons les principaux protocoles d'authentification. La description de notre protocole d'authentification sera faite dans cette troisième section avec une présentation du principe et ainsi que les différents algorithmes utilisés. Ensuite, au niveau de la quatrième section nous présenterons l'intégration de ce protocole dans une application web utilisant une base de données. Enfin, nous étudierons au niveau de la cinquième section la sécurité de ce mécanisme d'authentification.

8.2 Principes :

8.2.1 Définitions:

Identification

L'identification est un principe permettant de connaître l'identité d'une entité. Le vérificateur vérifie l'information révélée par l'entité contre celles de toutes les entités connus. L'identification doit etre unique.

Authentification

L'authentification est un mécanisme qui permet de prouver l'identité dont se réclame une entité (utilisateur, application, équipement,...) ayant à interagir avec les autres objects du Système d'Information. Selon le mecanisme mis en oeuvre, la preuve d'identité est plus ou moins forte.

8.2.2 Principes

L'accès aux ressources d'un système d'information par une entité, se décompose en trois sous-processus:

- l'authentification,
- l'identification (identité numérique (Internet)),
- le contrôle d'accès (contrôle d'accès logique).

La mise en place d'un mmécanisme d'authentification dans un système d'information permet:

- la protection du patrimoine informatique des entreprises,
- la protection des intérêts commerciaux et supérieurs des institutions publiques et prrivées,
- la réduction du coût d'attaques, de la perte de temps et de l'information.

A ce titre, le principe d'authentification pour un système informatique repose sur un processus permettant au système de s'assurer de la légitimité de la demande d'accès faite par une entité (être humain ou un autre système...) afin d'autoriser l'accès à des ressources du système (systèmes, réseaux, applications...) conformément au paramétrage du contrôle d'accès. L'authentification permet donc, pour un système, de valider la légitimité de l'accès aux ressources du système, ensuite le système attribue à cette entité les données d'identité pour cette session (ces attributs sont détenus par le système ou peuvent être fournis par l'entité lors du processus d'authentification). C'est à partir des éléments issus de ces deux processus que l'accès aux ressources du système pourra être paramétré (contrôle d'accès).
L'authentification contribue à la facturation des services, à la confiance dans l'économie numérique, condition indispensable du développement économique et ainsi à la protection de la vie privée. Les données personnelles véhiculées dans les systèmes d'information sont des données sensibles à protéger.
La notion d'authentification s'oppose à celle de l'identification d'une personne physique ou morale (dirigeant et toute personne autorisée). Cette distinction est importante puisque par abus de langage, on parle d'authentification alors qu'il s'agit d'identification. Lorsqu'une personne présente sa pièce d'identité lors d'un contrôle, elle est identifiée grâce à un document officiel, mais n'est pas authentifiée, car le lien entre la pièce d'identité et la personne n'est pas établie de façon indiscutable, irrévocable et reconnue par les tribunaux en cas de litige. Par opposition, lorsqu'une personne est authentifiée, cette authentification doit-être apportée par un tiers de confiance et par une preuve au sens juridique reconnue devant les tribunaux (ex: la signature électronique de

Figure 8.1 – Principe d'authentification forte

la carte bancaire).

Les systèmes d'authentification courants utilisent un seul facteur (en général un mot de passe). Le principe de l'authentification forte est d'utiliser plusieurs facteurs de nature distincte afin de rendre la tâche plus compliquée à un éventuel attaquant. Les facteurs d'authentification sont classiquement présentés comme suit :

- Ce que l'entité connaît (un mot de passe, un code NIP, une phrase secrète, etc.)
- Ce que l'entité détient (une carte magnétique, RFID, une clé USB, un PDA, une carte à puce, un smartphone, etc.). Soit un élément physique appelé jeton d'authentification, authentifieur ou token.
- Ce que l'entité est, soit une personne physique (empreinte digitale, empreinte rétinienne, structure de la main, structure osseuse du visage ou tout autre élément biométrique)
- Ce que l'entité sait faire ou fait, soit une personne physique (biométrie comportementale tel que signature manuscrite, reconnaissance de la voix, un type de calcul connu de lui seul, un comportement, etc.).

8.3 Les protocoles d'authentificationn :

L'authentification des utilisateurs intervient :

- sur un réseau local,
- sur un réseau étendu local client-serveur de type Intranet,
- sur un réseau internet pour accèder des parties confidentielles ou payantes des sites web,
- sur un réseau local client-serveur classique : Windows, Linnux,...
- sur un réseau étendu avec accès distant (VPN): Windows RAS (Remote Access Service),...

- accès à des serveurs confidentiels ou payants, téléservice bancaire, ordre de bourse, commerce électronique, etc.

A ce sens, plusieurs protocoles d'authentification sont créés. Ces protocoles sont utilisés suivants des besoins spécifiques par rapport au type d'authentification de l'utilisateur. Nous trouvons plusieurs protocoles d'authentification, les principaux sont:

8.3.1 Protocole d'authentification d'origine

Principe

Soit A et B deux entités qui souhaitent échanger des informations en utilisant protocole d'authentification d'origine. Dans le cas où A desire comminiquer à B, les différentes méthodes utilisées pour que B soit assuré que le message a été créé par A sont:

❶ Méthode utilisant un algorithme asymétrique de signature :
A->B : a.{m}SK : A chiffre le message m à partir de sa clé privée (signature)

❷ Méthode utilisant un MAC :
A->B : a.m|H(m) : A envoie le message plus son haché H(m)

Vulnérabilité

Ce protcole d'authentification est vulnérable contre l'attaque de rejeu. Cette attaque est une forme d'attaque réseau dans la quelle une transmission est malicieusement répétée par un attaquant qui a intercepté la transmission. Il s'agit d'un type d'usurpation d'identité.
Les attaques par rejeu peuvent être contrées en utilisant un identifiant de session en plus du mot de passe pour identifier une personne.

8.3.2 Protocoles d'authentification d'entité

Principe

Soit A une entité qui veut s'authentifier auprès d B. Dans ce protocole, les types d'identification utilisés sont:

❶ **Authentification faible** : L'utilisation des mots de passe fixes ou mots de passe à sens unique.

❷ **Authentification forte** : Ce type d'authentification utilise le protocole défi-réponse. Il est basés sur la cryptographie symétrique ou asymétrique

Vulnérabilité

- Possibilité d'intercepter le mot de passe
- Mots de passes stockés dans un fichier protégé en lecture et écriture, ou

- Utilisation de fonction à sens unique avant le stockage (exemple: login sous unix)
- Attaque par dictionnaire

8.3.3 Authentification forte par défi-réponse basée sur une clé partagée

Principe

Variante 1 :

❶ A->B : A envoie son identifiant.

❷ B->A : eb : B génére le challenge eb.

❸ A->B : {eb}Kab : A envoie à B le chiffre du challenge. Le challenge est chiffré à partir de la clé partagée par A et B (Kab).

Variante 2 :

❶ A->B : A envoie son identifiant.

❷ B->A : {eb}Kab : B déchiffre le chiffré grace sa clé partagée Kab.

❸ A->B : eb : Le challlenge

Variante 2 :

❶ A->B : {ta}Kab

Nécessite que A et B aient des horloges synchronisées. Bob déchiffre le message et s'assure que ta est dans un intervalle de temps raisonnable.

Vulnérabilité

Si la base de donnée du côté serveur (B) est corrompue, Alice peut être usurpée par un intrus.

Solution

L'utilisation de la clé publique est une solution.

8.3.4 Authentification forte par défi-réponse à base de clés publiques

Principe

Variante 1 :

❶ A->B : A envoie son identifiant.

❷ B->A : eb : B génére le challenge eb.

❸ A->B : {eb}SKa : A signe le du challenge à partir de sa clé privée.

Variante 2 :

❶ A->B : A envoie son identifiant.

❷ B->A : {eb}PKa : B vérifie la signature du challenge avec la clé publique de A.

❸ A->B : eb : Le challlenge

Vulnérabilité

L'authentification est unilatérale car un intrus peut faire signer à A un message, ou intercepter un message qui lui est destiné et le lui faire déchiffrer !!!

Solution

L'utilisation de l'authentification basée sur KEM/DEM. Ce protocole d'authentification sera présenté dans la section 8.4.

8.3.5 Kerberos

Le protocole Kerberos est un service distribué d'authentification qui permet à un procédé (un client) de prouver son identité à un verificateur (un serveur d'application, ou serveur simplement) sans envoyer des données à travers le réseau qui pourrait permettre à un agresseur à les imiter postérieurement. Il garantit l'intégrité et la confidentialité des données envoyées entre le client et serveur.

8.3.6 SecurWar ID

Le protcole SecurWar ID permet une authentification forte des utilisateurs par carte à puce. Il permet aussi précisément d´authentifier sans aucun doute les utilisateurs distants. Il se base sur les principes suivants :

- les informations échangées pour l´authentification ne sont jamais deux fois identiques et ne présentent aucun élément exploitable par les observateurs externes.
- la conservation des clés secrètes sur carte à microprocesseur offre une garantie d´inviolabilité maximale.
- chaque carte utilisateur contient une clé différente, et son utilisation est protégée par la frappe d´un code personnel qui bloque la carte après trois tentatives erronées.
- une session de sécurité permet un contrôle permanent de l´origine des connexions applicatives, et des ré-authentifications régulières et automatiques en cours de dialogue (transparente pour l´utilisateur) permet de prévenir toute tentatives d´usurpation d´identité

8.3.7 HTTP auth

Le protocole HTTP Auth permet d'offrir un service d'authentification continue. Il utilise des techniques cryptographie forte, avec des protocoles et algorithmes de hash et signature éprouvés (MD5, SHA/DSA, RSA) combinés avec des clés longues et fortes tels que :

- Algorithmes symétriques : 128 bits.
- Algorithmes asymétriques : 1024 ou 2048 bits.

Le protocole HTTP Auth est déployé au niveau du serveur et ne nécessite aucun programme, module ou objet sur les postes des clients. HTTP Auth crée dynamiquement les signatures numériques de chaque utilisateur, puis les envoie (après chiffrement fort) sur le poste de l'utilisateur à chaque connexion HTTP.

8.4 Protocole d'authentification KEM/DEM :

Ce protocole est une authentification forte utilisée dans un serveur web. Il utilise des algorithmes cryptographiques robuste capable de resister contre les attaques critiques et l'attaque des anniversaires et autres. Dans ce protocole, nous avons utilisé la fonction de hachage SHA512 (SHA3), réputée sûre (cf. 7.2.3), pour la construction de la clé session, le cryptosystème de Niederreiter décrit dans 5.3 pour la protection du challenge et de plus cette clé de session sera utilisée pour assurer la non-répudiation et l'intégrité du message. Ce protocole est une version améliorée http Auth dans la mesure où il utilise des algorithmes cryptographiques post-quantums. Le noyau de base de ces différents algorithmes (SHA3, Niederreiter) est développé en langage C. A partir des programmes C quatre fichiers executables sont générés et puis exportés dans le serveur d'application apache. Dans ce qui suit nous présenterons en détails la description du protocole d'authentification avec les différentes phases de la conception du programme classique C jusqu'à son utilisation dans une application web.

8.4.1 Description :

Nous présentons le protocole d'authentification forte qui utilise la fonction hachage SHA3 comme KDF et le cryptosystème asymétrique de Niederreiter. Le programme C contenant les différentes fonctions de SHA3, de et Niederreiter sera compilé pour donner quatre fichiers executables suivants:

- **keygen** est le fichier executable utilisé pour la génération de la clé public et privée de Niederreiter. En effet cette partie consiste à générer une paire de clé (clé publique et privée) pour le serveur et aussi une paire de clé pour un client. La confiance numérique du protocole d'authentification exige que chaque client du système posséde la clé publique du serveur et sa propre clé privée. Le serveur stocke dans son magazin de clé sa propre clé privée et la clé publique de chaque client.

- **encrypt** est le fichier utilisé pour le chiffrement du challenge e. Ce chiffrement sera fait dans les deux sens i.e.
  - d'une part le serveur chiffre le challenge avec la clé publique du client qui demande une connexion pour ainsi envoyer ce chiffré,
  - d'autre part le client, après avoir déchiffré avec sa clé privée, chiffre encore le challenge avec la clé publique du serveur et puis l'envoie au serveur.
- **decrypt** est celui utilisé pour le déchiffrement avec la clé privée du client et celle du serveur.
- **secretKey** le fichier exécutable utilisé pour construire la clé session K. Cette clé de session est construite en utilisant le challenge e et la fonction KDF. Elle permet de garantir la non-répudiation et l'intégrité des échanges entre le serveur et le client.

Génération des clés

- $(\triangle, \mathrm{M})$ sont respectivement la clé privée et la clé publique du schéma asymétrique de Niederreiter. Pour construire la clé publique nous calculons $\mathrm{H} = (\mathrm{M}|\mathrm{I}_{n-k})$. Dans ce protocole, nous générons les paires de clés suivantes:
  - $(\triangle_{\mathrm{C}}, \mathrm{M}_{\mathrm{C}})$ respectivement la clé privée et la clé publique du client.
  - $(\triangle_{\mathrm{S}}, \mathrm{M}_{\mathrm{S}})$ respectivement la clé privée et la clé publique du serveur.
- $e \in \mathscr{W}_{n,t}$ un mot aléatoire généré de taille n (où n est la longueur du code de Goppa choisi). Ce mot e sera le challenge du protocole d'authentification.
- Une clé de session K sera calculé avec la relation $\mathrm{K} = \mathrm{KDF}(e, \ell_{\mathrm{DEM}} + \ell_{\mathrm{MAC}})$ où $\mathrm{K} \in \mathrm{IF}_{2^{\ell_{\mathrm{DEM}} + \ell_{\mathrm{MAC}}}}$.
- Enfin,
  - Le client garde sa clé privée $\triangle_{\mathrm{C}}$ et la clé publique M_{S} du serveur pour chiffrer toutes informations envoyées aux serveur.
  - Le serveur stocke sa clé privée $\triangle_{\mathrm{S}}$ et la clé publique M_{C} du client pour chiffrer toutes informations envoyées au client.

Processus d'authentification

1. **CLIENT ==> SERVEUR :** Le client demande une connexion au serveur en envoyant son identifiant de connexion (id).
2. **SERVEUR ==> CLIENT :**Le serveur fait les étapes suivantes :
   - ❶ Génére un challenge e,
   - ❷ Chiffre le challange avec la clé publique du client demandeur M_{C} ($\psi_0 := \mathrm{M}_{\mathrm{C}}(e)$),
   - ❸ Calcule une clé de session avec $\mathrm{K} := \mathrm{KDF}(e)$,

❹ Séparer la clé de session en deux parties $K := (K_1||K_2)$ où $K_1 \in IF_{2^{\ell_{DEM}}}$ et $K_2 \in IF_{2^{\ell_{MAC}}}$,

❺ Calcul τ le paramatre de test d'intégrité (le HMAC) avec $\tau = E\nu(K_2)$.

❻ Après, il envoie au client $(\psi_0||\tau)$.

3. **CLIENT ==> SERVEUR :** Le client reçoit le cryptogramme formé du chiffré du challenge et τ. Il suit les étapes suivantes :

❶ Dechiffre ψ_0 avec la clé privée $\triangle_C$ pour obtenir e,

❷ Si le décodage réussi, il calcule la clé de session en utilisant le challenge $K := KDF(e)$,

❸ Sinon, il calcule la clé de session en utilisant la permutée du chiffré du challenge $K := KDF(\sigma(\psi_0))$,

❹ Séparer la clé de session en deux parties $K := (K_1||K_2)$ où $K_1 \in IF_{2^{\ell_{DEM}}}$ et $K_2 \in IF_{2^{\ell_{MAC}}}$,

❺ Calcul τ'_2 le paramatre de test d'intégrité (le HMAC) avec $\tau'_2 = E\nu(K_2)$.

❻ Chiffre le challange avec la clé publique du serveur M_S $(\psi_{2_0} := M_S(e))$,

❼ Après, il envoie au serveur $(\psi_{2_0}||\tau'_2)$.

4. **SERVEUR ==> CLIENT :** Le serveur fait la vérification :

❶ Dechiffre ψ_{2_0} avec sa clé privée $\triangle_S$ pour obtenir le challenge e,

❷ Si le décodage réussi, il calcule la clé de session en utilisant le challenge $K := KDF(e)$,

❸ Sinon, il calcule la clé de session en utilisant la permutée du chiffré du challenge $K := KDF(\sigma(\psi_{2_0}))$,

❹ Séparer la clé de session en deux parties $K := (K_1||K_2)$ où $K_1 \in IF_{2^{\ell_{DEM}}}$ et $K_2 \in IF_{2^{\ell_{MAC}}}$,

❺ Calcul τ''_2 le paramatre de test d'intégrité (le HMAC) avec $\tau''_2 = E\nu(K_2)$.

❻ Si$(\tau''_2 == \tau'_2)$, alors l'authentification reussie,

❼ Sinon l'authentification échoue.

Ce processus d'authentification est décrit dans le schéma de la figure **8.2** (PROCESSUS D'AUTHENTIFICATION).

8.4.2 Algorithmes:

Algorithme de création des clés

❶ Générer une paire de clé du schéma asymétrique Niederreiter pour le client :

- $M_C \in K_{pub}$: la clé publique du client.,
- Δ_C : la clé privée du client

❷ Générer une paire de clé du schéma asymétrique Niederreiter pour le serveur :

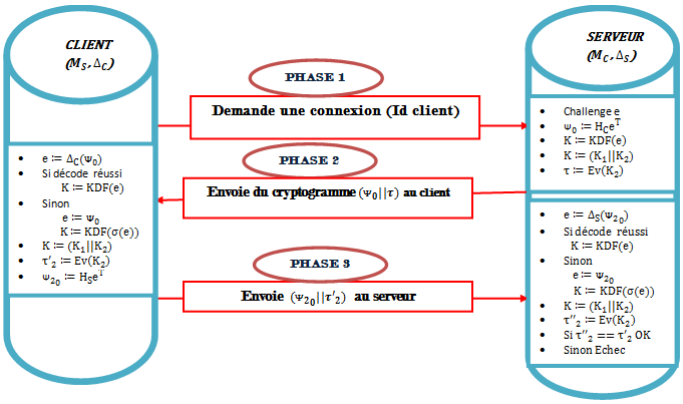


Figure 8.2 – PROCESSUS D'AUTHENTIFICATION

- $M_S \in K_{pub}$: la clé publique du serveur.,
- Δ_S : la clé privée du serveur

❸ Calculer une clé de session K échangée entre le client et le serveur.

- Choisir aléatoirement un mot $e \in \mathcal{W}_{n,t}$
- Calculer $K := SHA3(e, \ell_{DEM} + \ell_{MAC})$: la clé session,

Algorithme du processus d'authentification

PASE 1 : CLIENT

❶ Le client envoie son Id au serveur.

PASE 2 : SERVEUR

- **INPUT**: e, Id
- **OUTPUT**:$(\psi_0||\tau)$

❶ $\psi_0 := H_C e^T$

❷ $K := SHA3(e, \ell_{DEM} + \ell_{MAC})$

❸ $(K_1||K_2) := K$

❹ $\tau := E\nu(K2)$

❺ return $(\psi_0||\tau)$

PHASE 2: CLIENT

- **INPUT**: $(\psi_0||\tau)$
- **OUTPUT**:$(\psi_{2_0}||\tau'_2)$

❶ $e := \mathrm{D}ecode_{\Delta_{\mathrm{C}}}(\psi_0)$

❷ Si(Decode reussi) $\mathrm{K} := \mathrm{SHA3}(e, \ell_{\mathrm{DEM}} + \ell_{\mathrm{MAC}})$

❸ Sinon $\mathrm{K} := \mathrm{SHA3}(\sigma(\psi_0), \ell_{\mathrm{DEM}} + \ell_{\mathrm{MAC}})$

❹ $(\mathrm{K}_1 || \mathrm{K}_2) := \mathrm{K}$

❺ $\tau_2' := \mathrm{E}\nu(\mathrm{K2})$

❻ return $(\psi_{2_0} || \tau_2')$

PHASE 3: SERVEUR

- **INPUT**: $(\psi_{2_0} || \tau_2')$
- **OUTPUT**:b

❶ $e := \mathrm{D}ecode_{\Delta_{\mathrm{C}}}(\psi_{2_0})$

❷ Si(Decode reussi) $\mathrm{K} := \mathrm{SHA3}(e, \ell_{\mathrm{DEM}} + \ell_{\mathrm{MAC}})$

❸ Sinon $\mathrm{K} := \mathrm{SHA3}(\sigma(\psi_{2_0}), \ell_{\mathrm{DEM}} + \ell_{\mathrm{MAC}})$

❹ $(\mathrm{K}_1 || \mathrm{K}_2) := \mathrm{K}$

❺ $\tau_2'' := \mathrm{E}\nu(\mathrm{K2})$

❻ Si$(\tau_2'' == \tau_2')$ $b = 1$

❼ Sinon $b = 0$

❽ Return b

8.5 Sécurité du protocole d'authentification KEM/DEM

8.5.1 Sécurité contre l'attaque Man in the Middle

Toutes les informations confidentielles échangées entre le client et le serveur dans ce protocole sont chiffrées. De plus, la fonction de hachage SHA3 est utilisée pour assurer l'intégrité des informations qui transitent dans le canal. Ainsi , l'utilisation des algorithmes de chiffrement et de hachage prouve que ce protocole peut resister contre les attaques de type Man in the Middle.

8.5.2 Sécurité contre l'attaque de rejeu

La particularité de ce protocole par rapport aux autres est que chaque entité dispose sa clé privée qui lui permet de vérifier l'origine des informations. Ce controle bidirectionnel permet à ce protocole de resister contre l'attaque de rejeu. De plus, l'utilisation de la clé de session peut aussi éviter l'attaque de rejeu.

8.5.3 Sécurité contre les attaques critique

De la meme manière que KEM/DEM du chapitre précedent, ce protocole peut resister contre les attaques critique. L'utilisation de la permutation dans le cas où le décodage échoue peut éviter l'attaque de Bernstein et al. dans [5].

8.6 Application

Le protocole sera déployé dans une application PHP utilisant une base de donées Mysql. Pour chaque utilisateur créé le protocole lui associe une paire de clé publique et privée et aussi un mot aléatoirement généré. Ainsi, le mot aléatoire sera utilisé pour construire la clé secréte. Cette dernière sera copiée directement chez le client pour servir de clé d'authentification. Enfin, la clé publique, la clé privée et le chiffré du mot aléatoire seront sauvegardés dans le serveur et vont servir de verification du paramétre d'identification.

8.6.1 Prérequis

Les outils nécessaires pour le déploiement de ce protocole d'authentification sont :

- une plateforme Linux (Ubuntu, Debian, Red Hat,...) pour gérer l'environnement serveur.
- un compilateur gcc pour la compilation et l'execution des codes C
- un serveur d'application apache pour l'execution des codes PHP. Dans le cas où nous utilisons d'autres technologies, il n'est pas exclus de choisir un autre serveur d'application.
- un serveur de base de données pour le stockage des données. Dans notre cas, la base de données MySql est choisie.

8.6.2 Fonctionalités

Le protocole d'authentification basé sur les codes déployés dispose des fonctionalités suivantes:

- Protection des serveurs : le protocole permet de vérifier à chaque appel d'un programme transactionnel du serveur Web que l'utilisateur est autorisé à l'exécuter. Il s'agit d'un service d'authentification forte.
- Session continue dans un contexte asynchrone : le produit authentifie de façon « continue » les utilisateurs; le système affecte à chaque utilisateur une session qui est signée.
- Paramètrage des durées des sessions : Il est possible de définir des durées maximales de session pour chaque utilisateur, avec déconnexion du service à expiration.
- Logging complet des actions utilisateurs : chaque appel de script provoque un logging complet et détaillé : paramètres d'identification de l'utilisateur (adresse IP, OS du poste client, marque/modèle de navigateur, date/heure, etc.).

Une description de l'application sera présentée dans l'annexe de la thése.

Conclusion et perspectives

"Comment la fin justifierait-elle les moyens? Il n'y a pas de fin, seulement des moyens à perpétuité."

René Char

La cryptographie asymétrique s'est répandue depuis la fin des années 1976 à la suite des travaux de Diffie et Hellman [12], de la diffusion du protocole RSA (Rivest, Shamir et Adleman) [28] qui sont basés sur la théorie des nombres et de plus des travaux R.J. McEliece [20] pour la cryptographie basé sur la théorie des codes. La sécurité de ces protocoles basés sur la théorie des nombres repose sur la difficulté de problèmes mathématiques sous-jacents tels que le problème de la factorisation des grands entiers, le problème du logarithme discret. Aujourd'hui, le problème majeur de certains protocoles basés sur la théorie des nombres (ex. RSA) est l'arrivée de la cryptographie quantique. Face à cette problèmatique la cryptographie basé sur les codes est considérée comme une solution efficace. C'est ainsi que nous assistons à une prolifération des cryptosystèmes basés sur les codes. Ces schémas de chiffrement sont des alternatifs de McEliece avec une subtutition de code ou une amélioration de McEliece.

Au cours de ce travail de thèse, nous avons étudié dans une première étape l'amélioration d'un schéma de chiffrement de Sidel'nikov [17] et dans une seconde étape une amélioration et implémentation d'un schéma hybride basé sur les codes [8]. A cet effet, nous avons effectué des controles de preuve de sécurité du schéma de chiffrement de Sidel'nikov amélioré en fonction des attaques structurelles et par décodage. De plus, nous avons procédé à une implantation logicielle efficace du schéma hybride présenté dans la seconde partie et avons vérifié leurs éventuels bénéfices en terme de performance. Enfin, dans la dernière partie nous avons présenté un protocole d'authentification forte basée sur les codes. Ce protocole a été expérimenté dans une application web.

Dans une première partie, nous avons présenté les préliminaires avec des rappels mathématiques, cryptographies, théorie des codes ainsi que les protocoles cryptographies basés sur les codes et les principales attaques susceptibles de menacer les opérations précédentes.
Dans le chapitre 2, nous avons rappelé les fondements de la mathématiques sur les anneaux et corps finis. Nous avons présenté les principales opérations sur les éléments de ces ensem-

bles : l'addition, la soustraction, la multiplication et l'élévation au carré, la réductionmodulaire, l'inversion modulaire et le calcul de racine carrée dans le cas de la caractéristique deux.
Au niveau du troisième chapitre, nous avons présenté les codes correcteurs d'erreurs dans sa généralité. A cet effet, nous avons décrit les éléments fondemmentaux des codes correcteurs d'erreurs (distance hamming, poids, codage des mots,...), les codes linéaires et ainsi les décodages. Nous avons rappelé dans ce chapitre les proprités des codes de Reed Muller ainsi que les caractèristiques de ces codes.
Au chapitre 4, nous avons évoqué les notions fodemmentales de la cryptographie par une description de la cryptographie symétrique, asymétrique et aussi hybride ainsi que les différents algorithmes correspondants.
Dans le chapitre 5, nous avons rappelé les cryptosystèmes basés sur les codes correcteurs d'erreurs avec une présentation du cryptosystème de McEliece [20], Niederreiter [22] et aussi de Sidelnikov [33] ainsi que les algorithmes correspondants. C'est dans ce chapitre que nous avons décrit les attaques qui peuvent menacer ces cryptosystèmes.

La seconde partie de la thése, nous avons proposé nos différentes contributions. Au niveau du chapitre 6, nous avons exposé un nouveau schéma de chiffrement basé sur les codes de Reed Muller modifiés. Ce schéma de chiffrement est une amélioration du schéma de Sidelnikov. Enfin, nous avons décrit au niveau du chapitre 7 une implantation logicielle efficace du schéma hybride basé sur les codes. Dans ce chapitre, nous avons amélioré le schéma hybride basé sur le cryptosystème asymétrique de Niederreiter. Dans l'implantation de ce schéma, nous avons utilisé le schéma de chiffrement symétrique AES pour le chiffrement et déchiffrement des données et la fonction de hachage Keccack (SHA512) pour le hachage de la clé et la gestion de l'intégrité (HMAC). Enfin, dans le chapite 8 nous avons décrit en détails une implémentation d'un protocole d'authentification forte basé sur les codes.

Perspectives

Nous présentons comme perspective de recherche:

- **Implémentation KEM-DEM EN VHDL**
- **Implémentation sur carte à puce du schéma de chiffrement hybride en utilisant Basic Card**
- **Implémentation sur carte à puce du générateur pseudo aléatoire Xsynd en utilisant Basic Card**
- **Implémentation du protocole d'authentification forte basé sur les codes dans un matériel (VHDL, Carte à puce, token)**

Bibliographie

[1] *Secure hash standard*. National Institute of Standards and Technology, Washington, 2002. Note: Federal Information Processing Standard 180-2. 79

[2] Secure hash standard. March 2012. Note: Federal Information Processing Standards Publication 180-4. 79

[3] C. Adams and H. Meije. Security-related comments regarding mceliece public key cryptosystem. pages 454–455, 1989. 57

[4] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{(n/20)}$: How 1+1=0 improves information set decoding. In *Eurocrypt 2012, Lecture Notes Computer Science, Springer-Verlag, 2012.*, 2012. 73

[5] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. Ntru prime. 2016. http://eprint.iacr.org/2016/461. 4, 79, 82, 84, 101

[6] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the McEliece cryptosystem. In *Johannes Buchmann and Jintai Ding (editors). Post-Quantum Cryptography, Second international workshop, PQCrypto 2008, Cincinnati, OH, USA, October 17-19, 2008, proceedings, Lecture Notes Computer Science 5299, Springer*, pages 31–46, 2008. 57, 73

[7] Anne Canteaut and Florent 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. pages 367–378, 1998. 65, 73

[8] Pierre-Louis Cayrel, Cheikh Thiecoumba Gueye, El Hadji Modou Mboup, Ousmane Ndiaye, and Edoardo Persichetti. *Efficient Implementation of Hybrid Encryption from Coding Theory*, pages 254–264. S. El Hajji et al. (Eds): C2SI 2017, LNCS 10194, Springer International Publishing AG 2017, Cham, 2017. 3, 5, 6, 76, 79, 81, 90, 102

[9] Pierre-Louis Cayrel, Cheikh Thiecoumba Gueye, Ousmane Ndiaye, and Robert Niebuhr. Critical attacks in code-based cryptography. *IJICoT*, 3(2):158–176, 2015. 79, 82

[10] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. volume 33, pages 167–226, Philadelphia, PA, USA, January 2004. Society for Industrial and Applied Mathematics. 3, 4, 76, 84

[11] Des. Data encryption standard. In *In FIPS PUB 46, Federal Information Processing Standards Publication*, pages 46–2, 1977. 38

[12] W. Diffie and M. Hellman. New directions in cryptography. pages 644–654, 1976. 37, 47, 102

[13] Maen T. Alrashdan Faraz F. Moghaddam and Omidreza Karimi. A hybrid encryption algorithm based on rsa small-e and efficient-rsa for cloud computing environments. 1 No 3:238–241, 2013. 85

[14] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In *In Advances Cryptology, Asiacrypt 2009, Lecture Notes Computer Science 5912, Springer*, pages 88–105, 2009. 57

[15] P. Gaborit and M. Girau. Lightweight code-based identification and signature. 2007. 57

[16] Assistant professor LJIET Ahmedabad. Gaurav R. Patel, Prof. Krunal Panchal PG Scholar. Hybrid encryption algorithm. pages 2321–9939, 2014. 48

[17] Cheikh Thiecoumba Gueye and El Hadji Modou Mboup. Secure cryptographic scheme based on modified reed muller codes. In *International Journal of Security and Its Applications*, pages 55–64, 2013. 4, 6, 102

[18] Pil Joong Lee and Ernest F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In *Christoph G. Guenther (editor). Advances cryptology-EUROCRYPT '88. Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques held Davos, May 25-27, Lecture Notes Computer Science 330, Springer, Berlin*, pages 275–280, 1988. 57

[19] Jeffrey S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. pages 1354–1359, 1988. 57

[20] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. In *Jet Propulsion Laboratory DSN Progress Report 42-44*, pages 114–116, 1978. 4, 52, 102, 103

[21] Lorenz Minder and Amin Shokrollahi. Cryptanalysis of the Sidel'nikov cryptosystem. In Moni Naor, editor, *Advances Cryptology-EUROCRYPT 26th annual international conference on the theory and applications of cryptographic techniques, Lecture Notes Computer Science 4515. Springer*, pages 347–360, 2007. 4, 6, 64, 65, 71, 72, 73

[22] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. In *Problems of Control and Information Theory 15*, pages 159–166, 1986. 54, 90, 103

[23] U.S. Department of Commerce, National Institute of Standards, and Technology. *Secure hash standard*. National Institute of Standards and Technology, Washington, 1995. Note: Federal Information Processing Standard 180-1. 79

[24] Ayoub Otmani and Hervé Talé Kalachi. Square code attack on a modified sidelnikov cryptosystem. In *Codes, Cryptology, and Information Security - First International Conference, C2SI 2015, Rabat, Morocco, May 26-28, 2015, Proceedings - In Honor of Thierry Berger*, pages 173–183, 2015. 4, 64, 73

[25] Edoardo Persichetti. Secure and anonymous hybrid encryption from coding theory. In Philippe Gaborit, editor, *Post-Quantum Cryptography: 5th International Workshop, PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings*, pages 174–187, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 4, 76, 79, 84

[26] Christiane Peters. *Information-Set Decoding for Linear Codes over F q*, pages 81–94. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. 57

[27] Eugene Prange. Re transactions on information theory, 8(5). page 5–9, September 1962. 57

[28] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978. 37, 102

[29] Ronald L. Rivest. The MD5 Message-Digest Algorithm (RFC 1321). http://www.ietf.org/rfc/rfc1321.txt?number=1321. 79

[30] B. Sakkour. Etude et amélioration du décodage des codes de reed muller d'ordre 2. In *These*, 2007. 26, 28, 29, 64

[31] Nicolas Sendrier. Finding the permutation between equivalent linear codes : the support splitting algorithm. pages 1193–1203, 2000. 71

[32] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, 1994. 3

[33] Vladimir M. Sidel'nikov. Open coding based on reed-muller binary codes. In *Russian. Diskretnaya Matematika 6, 3-20. English : A public-key cryptosystem based on binary Reed-Muller codes*, 1994. 4, 55, 64, 69, 103

[34] Jacques Stern. A method for finding codewords of small weight. In Gerard D. Cohen and Jacques Wolfmann, editors, *Coding theory and applications. Proceedings of the Third International Colloquium on Coding Theory,Lecture Notes Computer Science 388, Springer*, pages 106–113, 1989. 57

[35] R. Deng Y. Li and X. Wang. On the equivalence of mceliece's and niederreiter's cryptosystems. pages 271–273, 1994. 54

Résumé

Nom et prénoms du Candidat : **El Hadji Modou MBOUP**

Chiffrement hybride et Protocole d'authentification forte basé sur les codes correcteurs d'erreurs

Les cryptosystèmes basés sur la théorie des codes sont considérés comme des schémas très prometteurs pour la cryptographie post-quantique.
C'est dans cet optique que nous avons étudié dans la première partie de notre thése une amélioration du cryptosystème de Sidel'nikov afin de réparer la cryptanalyse de L. Minder et A. Shokrollahi. A cet effet, nous avons proposé une nouvelle variante de ce Cryptosystème avec une modification du code ReedMuller. Le but de cette modification des codes Reed Muller est de rendre quasiment impraticable l'attaque présentée par L. Minder et A. Shokrollahi.
De plus, dans la deuxième partie de la thése, nous avons étudié une amélioration d'un chiffrement hybride basé sur le PKCS de Niederreiter et ainsi l'implémenter en C. Dans ces travaux, nous avons réparé ce schéma hybride, afin qu'il resiste à l'attaque de Bernstein.
En inspirant du principe de ce chiffrement hybride, nous sommes intérerssés dans la troisième partie de cette thése à une étude d'un protocole d'authentification forte basé sur les codes. Le but de ces travaux est de proposer un mécanisme d'authentification forte pour les objets connectés sur internet.