HABILITATION À DIRIGER DES RECHERCHES

présentée devant

L'Université de Rennes 1
Institut de Formation Supérieure
en Informatique et en Communication

par

Eric Filiol [1]

Modèles booléens en virologie et en cryptologie

soutenue le 17 janvier 2007 devant le jury composé de

MM.	KIRCHNER Claude	Rapporteur
Mme	MAKNAVICIUS-LAURENT Maryline	Rapporteur
MM.	QUISQUATER Jean-Jacques	Rapporteur
MM.	COUSIN Bernard	Examinateur
MM.	MÉ Ludovic	Examinateur
MM.	STEYAERT Jean-Marc	Président

1. ESAT, Laboratoire de virologie et de cryptologie

Remerciements

Ce mémoire et le travail de recherche qui l'a rendu possible n'auraient pas été possibles sans les encouragements permanents, compréhensifs et quelquelfois très patients de mon épouse et de mon fils. Ils ont été les inspirateurs constants de ce travail de recherche, activité dévoreuse de temps souvent perçue comme une maitresse exigeante par ceux qui vous entourent. C'est la raison pour laquelle je tiens non seulement à leur rendre hommage mais également à leur dédier ce mémoire.

Les travaux de recherche que je présente ici, ainsi que ma passion pour cette activité, doivent énormément au général (R) Max Mayneris, qui m'a non seulement permis de m'épanouir dans ce domaine mais a également œuvré directement, alors que j'étais enseignant-chercheur militaire au Centre de recherche des École de Coëtquidan, pour que je puisse la placer dans le contexte opérationnel auquel j'ai toujours été très attaché. Je tiens à lui exprimer ici ma très respectueuse gratitude.

Je souhaiterais tout d'abord remercier Bernard Cousin pour tous les conseils et toute l'aide qu'il m'a prodigué avant et pendant la préparation de cette habilitation. Cette aide et sa disponibilité ont été particulièrement précieuses.

Je tiens à remercier Maryline Maknavicius-Laurent, Claude Kirchner et Jean-Jacques Quisquater de l'insigne honneur qu'ils m'ont fait en acceptant d'être les rapporteurs de ce mémoire, pourtant dense à la croisée de plusieurs disciplines. Je les remercie pour cela, pour leurs conseils amicaux et pour leur gentillesse qui m'ont accompagné dans la préparation et l'amélioration de ce document.

Merci également à Jean-Marc Steyaert qui m'a fait le plaisir et l'honneur de participer au jury final et de le présider de main de maître, avec cette touche d'humour et cet esprit de légère impertinence qui le caractèrise et le rend si attachant. Je le sais particulièrement occupé,

et ce en permanence, aussi l'occasion de le revoir sur Rennes pour cette soutenance est un réel plaisir.

Merci à Ludovic Mé, qui outre d'avoir également accepté de participer au jury, a dès le début joué le rôle d'ancien et m'a guidé, conseillé et aidé dans la préparation de cette habilitation. Le chemin maintenant pratiquement achevé, je mesure combien ses conseils avisés ont été précieux. Je tiens également à témoigner de sa gentillesse et de sa bonne humeur qui le caractérisent.

L'activité de recherche est stimulante, en autres choses, parce qu'elle permet de rencontrer d'autres chercheurs, de confronter ses propres idées aux leurs et plus généralement de rencontrer des gens passionnants et passionnés et de travailler ensembles. Pour ma part, ces personnes ont été nombreuses et je tiens à les remercier pour les différents échanges et collaborations que j'ai pu avoir avec eux, durant toutes ces années : les membres du (célèbre) cours SSIC de l'ESAT et en particulier le chef de bataillon Evrard, les chercheurs du projet Codes de l'INRIA, ceux du laboratoire d'informatique fondamentale de Lille, ceux du département de mathématiques de la faculté des sciences de Limoges, les membres du comité de programme de SSTIC, Arnaud Metzler, Fred Raynal (aka Pappy), Caroline Fontaine, Claude Petit, Vlasti Broucek et Paul Turner de l'université de Tasmanie, Rainer Fahs, Sebastian Rohr, Urs Gattiker, Bimal Roy, Aleksander Kolosha, les personnels des centres de l'Alliance Française de Saint Petersbourg, de Moscou, de Samara, de Toliatti et de Nijni Novgorod qui m'ont donné l'occasion de faire connaître la science française en Russie, Jean-Yves Marion et Guillaume Bonfante, Radu State, Mathieu Kaczmarek du LORIA, Markkus O. Saarinen – merci à lui pour son soutien face à un certain manque de « courtoisie » –, Guillaume Arcas, Cédric Tavernier, les membres du comité de lecture du Journal in Computer Virology*, Robert Erra et Benjamin Caillat de l'ESIEA, Mohammed Lambarki de l'EMIAE à Casablanca... et tous ceux que la fatigue du à un manque de sommeil chronique – et non l'ingratitude – me fait oublier temporairement.*

Je tiens également à remercier tous mes stagiaires et thésards qui ont accepté de me suivre, de faire des mathématiques et de la programmation plus qu'ils ne l'auraient souhaité pour certains, de se mettre à LaTeX– ils n'avaient pas le choix de toute manière –, et d'oublier

à cause de moi ce que sont soirées, nuits, week ends *et vacances. Ils ont très largement contribué à faire avancer bien des projets du laboratoire : Johann Barbier, Sébastien Josse, le capitaine Mayoura, Erwan Attié, Philippe Beaucamps, Jean-Paul Fizaine, Grégoire Jacob, Alexis La Goutte, Mickaël Le Liard, Jean-Marie Borello, Aurélien Derock, Alessandro Gubbioli de l'université de Milan, les lieutenants de vaisseau Ratier, Turcat, Hansma, de Drézigué, les lieutenants Azatassou, Tanakwang, Smithsomboon, Plan, Helo,...*

Merci également à Nathalie Huilleret, Brigitte Jülg, Méline Berthelot, Nicolas Puech, Guido Zozimo Landolfo des éditions Springer qui ont soutenu plusieurs projets éditoriaux dans le domaine de la virologie informatique et ainsi ont très largement contribué à redonner à cette science le statut noble qu'elle mérite.

Que vive la science et, grâce à elle, toute la noblesse de l'esprit et du cœur qui en un seul mélange se fondent. Que tous ceux qui croient en ses principes et les défendent, pour le bonheur et la liberté des générations futures, en soient remerciés. Je leur dois beaucoup.

Table des matières

1 Introduction **1**
- 1.1 Les fonctions booléennes 1
- 1.2 Guide de lecture 4

2 Fonctions booléennes et détection virale **7**
- 2.1 La modélisation de la détection virale 8
- 2.2 Modèle mathématique de l'analyse de forme 10
  - 2.2.1 Définition d'un schéma de détection 10
  - 2.2.2 Propriétés des schémas de détection 14
- 2.3 Le problème de l'extraction 20
  - 2.3.1 L'extraction de schémas de détection 20
  - 2.3.2 Approche naïve : Algorithm E-1 21
  - 2.3.3 Approche par apprentissage de DNF - Algorithme E-2 23
  - 2.3.4 Exemples de fonctions de détection 28
- 2.4 Schéma de détection sécurisé 31
  - 2.4.1 Constructions combinatoires et probabilistes . . . 32
  - 2.4.2 Analyse mathématique 37
  - 2.4.3 Implémentations et performances 40
- 2.5 Problèmes ouverts et axes de recherche futurs 40

3 Fonctions booléennes, diffusion et confusion **45**
- 3.1 Introduction 45
- 3.2 Caractérisation des fonctions booléennes et résultats . . . 47
  - 3.2.1 Structure d'une forme algébrique normale 47
  - 3.2.2 Caractérisation des coefficients de Walsh 51
- 3.3 Un nouveau test statistique 53
  - 3.3.1 Le test de la constante affine 55

3.3.2 Le test des d-monômes 55
3.3.3 Tests des d-monômes sur un sous-ensemble fixé de sorties 56
3.4 Résultats de simulation 57
3.4.1 Chiffrement par flot 57
3.4.2 Systèmes de chiffrement par bloc 59
3.4.3 Fonctions de hachage 60
3.5 Problèmes ouverts et axes de recherche futurs 62

4 L'attaque par codes à répétition 65
4.1 Introduction 65
4.2 Notations et rappels théoriques 68
4.2.1 Codes à répétition 68
4.2.2 Systèmes de chiffrement par blocs et cryptanalyse linéaire 70
4.3 Cryptanalyse de systèmes de chiffrement par blocs par codes à répétition 71
4.3.1 Chiffrement par blocs et codes à répétition 71
4.3.2 Description de l'attaque PDRC 73
4.3.3 Critère de résistance contre l'attaque PDRC . . . 77
4.4 Implémentation pratique de l'attaque PDRC 79
4.5 Généralisation : la cryptanalyse par code à répétition . . 82
4.6 Conclusion 85

5 Preuve de cryptanalyse à apport de connaissance nulle 87
5.1 Introduction 87
5.2 Concepts et notations 90
5.3 Le chiffrement Bluetooth 91
5.3.1 L'algorithme de chiffrement E0 93
5.3.2 Etat de l'art de la cryptanalyse de E0 95
5.4 Preuve de cryptanalyse de type Zero-knowledge 97
5.4.1 La propriété du poids de Hamming 100
5.4.2 La propriétés des sous-suites constantes (*runs*) . . 101
5.4.3 Cumul des propriétés du poids de Hamming et des run 102
5.5 Preuve de cryptanalyse de type Zero-knowledge pour E0 104
5.6 Travaux futurs, perspectives et conclusion 106

6 Conclusion **109**

Bibliographie **112**

Annexes **123**
.1 Résultats d'analyse d'antivirus 123
.2 Motif de détection et fonction de non détection de Norton pour *W32.Bagle.E* 130
.3 Implémentation de référence de l'algorithme E0 132
.3.1 Header File « include.h » 132
.3.2 Fichiers d'entête « e0light.h » 132
.3.3 Procédure de chiffrement 133
.3.4 Programme principal 134
.4 Valeurs de preuve 0-K 134

Table des figures

4.1 Canal binaire symétrique et système de chiffrement par blocs 72
4.2 Codes concaténés 75

5.1 Description fonctionnelle du chiffrement Bluetooth 92
5.2 Description fonctionnelle de la procédure de chiffrement E0 94

Liste des tableaux

2.1 Algorithme d'extraction de schéma de détection E-1 . . . 22
2.2 Algorithme d'extraction de schéma de détection E-2 . . . 25
2.3 Fonctions booléennes modélisant la détection comportementale . . . 43

3.1 Probabilités exacte et approchée pour une fonction d'être équilibrée . . . 53
3.2 Systèmes par flot : résultats de tests (niveau de signification $\alpha = 0.001$) . . . 57
3.3 Lili128 : résultats expérimentaux pour les tests T_1^d et T_2^d. 58
3.4 DES : valeurs des estimateurs D^2 . . . 59
3.5 AES(128, 128) : Valeurs des estimateurs D^2 . . . 61
3.6 Résultats expérimentaux pour les tests T_1^1 et T_2^1 ($\alpha = 0.05$). 61
3.7 Résultats expérimentaux pour les tests T_1^2 et T_2^2 ($\alpha = 0.05$). 62

4.1 Fluctuation du ε observé . . . 67
4.2 Algorithme A.1 de cryptanalyse par code á répétition . . 74
4.3 Algorithme A.2 de cryptanalyse par code á répétition concaténés . . . 76
4.4 Comparaison de la cryptanalyse linéaire de Matsui et de la cryptanalyse RC pour le DES . . . 84

5.1 Les polynomes de rétro-action de E0 . . . 95
5.2 Bijection linéaire de E0 . . . 95
5.3 Complexités comparées des attaques à suite longue sur E0 96
5.4 Complexités comparées des attaques à suite courte sur E0 97
5.5 Complexité pour la réalisation de la propriété du poids de Hamming (approche aléatoire ; $n = 128$) . . . 101

5.6 Comparaison des complexités pour les propriétés du poids de Hamming et des runs (recherche aléatoire ; $n = 128$) . 103
5.7 Nombre de clefs retrouvées pour quelques valeurs de (k, r) significatives ($n = 128$) 105

1 Antivirus testés (versions & bases de signatures) 123
2 Motifs de détection pour *W32/Bagle.A* 125
3 Motifs de détection pour *W32/Bagle.E* 126
4 Motifs de détection pour *W32/Bagle.J* 127
5 Motifs de détection pour *W32/Bagle.N* 128
6 Motifs de détection pour *W32/Bagle.P* 129

Chapitre 1

Introduction

1.1 Les fonctions booléennes

Les fonctions booléennes sont un outil puissant de modélisation mathématique, utilisé dans de nombreux domaines : informatique, cryptologie, électronique, télécommunications, recherche opérationnelle... Bien que cet outil soit très vite inutilisable en pratique, du fait de la complexité des ensembles de données mis en jeu, il se révèle extrêmement puissant par sa capacité à décrire simplement et efficacement de nombreux concepts, par ailleurs difficiles à appréhender.

Mon activité de recherche depuis une dizaine d'années est centrée sur l'étude et l'utilisation des fonctions booléennes en cryptologie et plus récemment dans celui de la virologie, où cette approche s'est révélée particulièrement puissante.

Dans le domaine de la cryptologie les fonctions booléennes sont des objets connus mais malheureusement sous-employés pour ne pas dire sous-considérés. La meilleure preuve réside dans la constatation que très peu d'articles sont retenus dans les grandes conférences considérées comme réputées dans le domaine de la cryptologie, alors que parmi ceux soumis certains sont de très bonne qualité. La raison tient peut être à la complexité de ces objets, difficiles ne serait-ce qu'à représenter et encore plus à étudier du fait de leur extrême richesse combinatoire. En cryptologie, une bonne fonction, comme beaucoup de primitives utilisées, ne doit présenter aucune structure exploitable. Or, définir ce qu'est une structure exploitable est déjà, en cryptologie, un problème en soi –

comment savoir ce que les cryptanalystes utilisent ou pourraient utiliser en réalité. Prouver que de telles structures ne sont pas réalisées dans une fonction booléenne devient vite impossible lorsque le nombre de variables dépasse la dizaine. Les fonctions booléennes n'attirent malheureusement que peu de chercheurs en cryptologie, la plupart d'entre eux préférant les élégantes structures mathématiques considérées par la cryptologie asymétrique.

Dans le domaine de la virologie, les fonctions booléennes étaient quasiment ignorées jusqu'aux travaux présentés dans ce mémoire. Le problème SAT a été considéré par D. Spinellis en 2003 [Spin03] pour démontrer que le problème de détection des virus polymorphes de taille bornée est NP-complet. Plus récemment en 2005, Kinder et al. [Kinder05] ont utilisé des techniques de preuve de modèles (*model checking*) pour la spécification de motifs de détection de codes malveillants. La preuve de modèles, par l'utilisation et le traitement de formules logiques partage certains aspects et techniques utilisées pour les fonctions booléennes (notamment dans le domaine des graphes). Mais cette approche ne permet de définir que des spécifications de détection – débouchant toutefois sur des techniques de détection plus puissantes que les techniques statiques d'analyse de forme généralement utilisées dans les produits antivirus –, autrement dits des ensembles d'instances particulières du problème de détection des codes malveillants. L'approche par *model checking* ne permet pas d'envisager, à ce jour, ce problème sous un angle général, comme le permettent les fonctions booléennes. L'analyse bibliographique n'a pas permis d'identifier d'autres cas, études ou résultats en virologie informatique ayant considéré directement ou indirectement les fonctions booléennes.

Une fonction booléenne est une fonction définie sur $\mathbb{F}_2^n$ où $\mathbb{F}_2$ représente le corps fini à deux éléments 0 et 1. Cette fonction est à valeur dans $\mathbb{F}_2$. Dans certains contextes, notamment la cryptologie, il est d'usage de considérer des fonctions booléennes généralisées, encore appelées fonctions booléennes vectorielles. Ces fonctions sont à valeur dans $\mathbb{F}_2^m$. Ainsi une fonction de hachage peut être vue comme une telle fonction avec $n > m$ alors que dans un système de chiffrement par bloc, nous aurons soit $m = n$ (cas de l'AES [AES]) soit $m < n$ (S-boxes du DES).

L'intérêt particulier des fonctions booléennes réside essentiellement dans les propriétés et caractéristiques suivantes :

- la considération de variables à deux valeurs, bien qu'en apparence assez pauvre, permet en réalité de décrire des objets infiniment plus complexes. Lorsque l'on doit appréhender des ensembles comportant un nombre très important d'objets, une variable booléenne décrit, selon sa valeur 0 ou 1, si l'objet représenté par cette variable fait partie ou non d'un sous-ensemble d'étude donné. Une fonction booléenne décrit alors une ou plusieurs propriétés sur ce sous-ensemble. Cet aspect s'est révélé particulièrement puissant dans mes travaux en virologie. La notion de motif de détection virale, dont l'élément de base est généralement l'octet, a pu ainsi être décrite de manière plus intéressante en considérant des variables booléennes. Cette approche a débouché sur une formalisation plus puissante de détection ;
- une fonction possède plusieurs formes de représentation dont les plus intéressantes sont les formes dites normales (algébrique, disjonctive, conjonctive...). Chaque forme permet de privilégier une ou plusieurs propriétés particulières que les autres formes ne permettront pas de considérer. Bien que le problème soit de complexité générale NP, le passage d'une forme à une autre ou la modélisation d'un système par une forme donnée plutôt qu'une autre permet d'obtenir des résultats fondamentaux très intéressants. Cette approche a été utilisée avec succès dans mes deux domaines de recherche tant en virologie qu'en cryptologie ;
- une fonction booléenne permet une représentation compacte de propriétés concernant des ensembles de taille démesurée qu'une approche exploratoire rend impossible à étudier. En particulier, il est toujours possible de considérer une approche analytique des objets que l'on est amené à étudier, plus particulièrement en cryptologie. Les fonctions booléennes sont les briques élémentaires permettant de décrire tous les autres objets. Ainsi, une fonction booléenne vectorielle $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ peut être vue comme une famille de m fonctions booléennes $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$. Cette vision m'a permis de définir une nouvelle méthode d'évaluation de sécurité des systèmes de chiffrement par bloc et des fonctions de hachage.

1.2 Guide de lecture

Ce mémoire résume une partie de mes recherches consacrées aux fonctions booléennes et à leurs applications essentiellement en cryptologie mais également en virologie. L'essentiel de ces recherches a été mené au sein du laboratoire de virologie et de cryptologie de l'École Supérieure et d'Applications des Transmissions. Le présent mémoire est organisé comme suit.

Le chapitre 2 présente une modélisation de la détection virale à l'aide de fonctions booléennes. Ces résultats ont permis d'unifier la vision traditionnelle opposant la détection par analyse de forme (dont l'une des techniques les plus connues est la recherche de signature) à celle d'analyse comportementale. Avec ce modèle, la détection se décrit simplement par la notion de schéma de détection, défini comme un couple constitué d'un ensemble de référence (une base de signatures ou une base de comportements) et une fonction (booléenne) de recherche. L'intérêt de cette modélisation est de pouvoir conceptualiser le problème de la détection sans considérer la nature des propriétés (forme ou fonctions) utilisées. Cette approche a permis notamment d'identifier de graves faiblesses dans les techniques de détection utilisées actuellement – notamment l'étude en boîte noire des antivirus par un attaquant – et de proposer un schéma de détection sécurisé supprimant ces faiblesses.

Le chapitre 3 traite de certains aspects en matière de cryptanalyse dans lesquels les fonctions booléennes jouent un rôle important. En effet, un système de chiffrement peut être décrit comme un ensemble de fonctions booléennes. Dans ce chapitre 3, je présenterai comment cette vision peut être utilisée pour caractériser et évaluer la sécurité de ces systèmes. Je présenterai ensuite un nouveau test statistique qui permet une vision plus structurelle de ces systèmes et qui a permis de détecter de nombreuses faiblesses, indétectables avec les tests classiques, pour des systèmes récents. Cette nouvelle modélisation permet également de définir rigoureusement les notions, fondamentales en cryptologie, de *diffusion* et de *confusion* [Shan48]. Jusqu'à présent ces propriétés n'avaient pas fait l'objet d'une caractérisation formelle.

Le chapitre 4 présente des travaux récents et en cours concernant la cryptanalyse des systèmes de chiffrement par blocs. L'utilisation d'une classe particulière de codes correcteurs d'erreurs, les codes à répétition, permet d'offrir une nouvelle vision de ces systèmes, vision dans

laquelle certaines propriétés des fonctions booléennes peuvent être déterminantes.

Enfin, le chapitre 5 présentera des résultats récents dans le domaine de la cryptanalyse opérationnelle. La publication de telles techniques efficaces posent d'importants problèmes de responsabilité, notamment vis-à-vis de la loi. Cependant, une autre responsabilité, elle morale et scientifique, impose dans certains cas de les révéler. Je montrerai comment résoudre ce problème en conciliant ces deux aspects antinomiques. En considérant des propriétés booléennes remarquables sur les suites de sortie d'un algorithme de chiffrement – lesquelles ne peuvent en pratique être obtenues avec les cryptanalyses connues – et en exhibant les clefs ayant permis de les produire, je prouve que cela résulte d'une faille conceptuelle du système. La preuve, vérifiable en temps polynomial, ne permet cependant pas de reconduire l'attaque. J'ai appelé ce type de preuve, *preuve de cryptanalyse de type Zero-knowledge*.

Afin de privilégier une présentation logique et harmonieuse de mes travaux, je traiterai en parallèle, des problèmes ouverts identifiés et qui constitueront quelques uns de mes axes de recherche futurs.

Chapitre 2

Fonctions booléennes et détection virale

L'utilisation de fonctions booléennes en virologie informatique n'est pas intuitive. Cette science a une image purement technique dans laquelle la part théorique représente une proportion extrêmement faible. Pour illustrer cela, signalons qu'en vingt ans, seules cinq thèses de doctorat ont été soutenues dans le monde. Seules les thèses de Fred Cohen [Cohen86] et de Ferenc Leitold [LEIT94] traitent des aspects fondamentaux et formels, et dans une moindre mesure, il est possible de retenir celle Markus Schmall [SCHMALL02]. Cette situation se retrouve en ce qui concerne les publications scientifiques : moins d'une dizaine de papiers théoriques sont répertoriés [Adleman88, BKM05, Chess00, FHZ05, Forrest97, Spin03, ZZ04, ZZ05]. Les quelques outils utilisés sont très éloignés des fonctions booléennes et en général des mathématiques discrètes : machines de Turing et fonctions récursives représentent l'essentiel des outils considérés par les quelques auteurs ayant travaillé dans le champ théorique de la virologie informatique. La principale motivation de ces quelques travaux a été dans un premier temps de modéliser les différents codes malveillants connus et d'en dresser une classification. Dans une seconde étape, l'objectif a été de déterminer la complexité de certaines instances du problème de détection, problème qui a été prouvé comme généralement indécidable par Fred Cohen [Cohen86]. L'essentiel des résultats a permis d'établir une vision claire du problème de la détection antivirale, et ce pour quelques familles de codes malveillants connus.

Ces quelques travaux théoriques ont ouvert de nombreuses voies de recherche et identifié de très nombreux problèmes ouverts [FHZ05]. Ces derniers concernent tant l'aspect considéré comme relevant de l'« attaque » (identification et définition de nouvelles classes de codes malveillants) que celui relevant classiquement de la « défense » (d'une manière générale le problème de la détection virale). L'expérience montre régulièrement que les modèles formels actuels ont, pour ces deux aspects, prouvé leurs limites. L'évolution des codes malveillants ainsi que la méthodologie des attaquants rend peu à peu obsolètes tous les modèles théoriques connus. Ainsi, à titre d'illustration, le concept de codes k-aires [1] n'est pas appréhendable par les modèles de Cohen [Cohen86] ou d'Adleman [Adleman88]. En particulier, la notion de synchronisme de codes ou celle d'inter-dépendance de code ne peut être prise en compte.

L'évolution des modèles en virologie informatique passe par l'utilisation de nouveaux outils plus adaptés. Parmi les nombreux outils des mathématiques discrètes susceptibles d'enrichir la panoplie déjà disponible pour le théoricien, les fonctions booléennes représentent un outil de choix qui a prouvé dans bien des domaines son intérêt et sa puissance [Fil01] (le meilleur exemple est très certainement celui de la cryptologie ; voir chapitre 3). La première – et unique à ce jour – utilisation des fonctions booléennes en virologie concerne le problème général de la détection virale, et ce indépendamment de la technique utilisée [Fil06a] qui sera présentée dans la section 2.1. Cette approche permet à présent de tester de manière rigoureuse et surtout reproductible, l'offre logicielle actuelle dans le domaine de la détection virale.

2.1 La modélisation de la détection virale

Le terme de détection virale concerne toutes les infections informatiques connues, qu'elles soient auto-reproductrices, comme les virus ou les vers, ou simple, comme les chevaux de Troie ou les bombes logiques. En 1986, Fred Cohen [Cohen86] a démontré que le problème de la détection virale était généralement indécidable. La preuve utilise

1. Les codes k-aires sont constitués d'un groupe de k codes, lesquels collaborent en vue d'une action offensive donnée et ce selon plusieurs modes possibles. Cette notion a été formalisée pour la première fois dans [Fil03].

l'indécidabilité du problème de l'arrêt [Tur36] et montre que ce dernier est réductible à celui de la détection virale.

Des résultats antérieurs ont permis d'établir des resultats de complexité pour certaines instances de ce problème, dans un contexte de temps/mémoire limités. La plupart de ces résultats ont montré que dans la plupart des cas, ces instances sont NP-complètes [Adleman88, BKM05, Spin03, ZZ04] voir au-delà [Adleman88, ZZ04].

Deux grandes approches dans les techniques de détection virale ont été identifiées :

- l'analyse de forme qui considère une code comme une suite d'octets, hors de tout contexte d'exécution. L'objectif est de rechercher des éléments caractéristiques d'une infection (les *signatures*), lesquels sont constitués de motifs plus ou moins complexes et qui doivent être rassemblés dans une base continuellement mise à jour. Cette analyse de forme peut également considérer des répartitions statistiques de certaines instructions (techniques d'*analyse spectale*) ou des codes d'intégrité locaux. La philosophie générale est de comparer, sur sa forme uniquement, un code en comparant des caractéristiques structurelles observées, à des valeurs de références rassemblées dans des bases ;
- l'analyse comportementale qui considère cette fois les actions menées, déclenchées par le code dans un contexte d'exécution directe (analyse en temps réel) ou indirecte par simulation en mémoire (techniques d'émulation). Dans ce cadre comportemental, les actions d'un code donné sont comparées à un modèle de référence et décrit en pratique à l'aide de bases de comportements qu'il est nécessaire de mettre également à jour, au fur et à mesure de l'évolution des comportements des codes malveillants.

Ces deux approches sont toujours présentées comme différentes, l'analyse comportementale étant considérée comme une évolution majeure par rapport à l'analyse de forme. La réalité est toute autre [Fil06a]. L'utilisation des fonctions booléennes permet d'unifier ces deux visions et de n'en considérer plus qu'une. De plus, cette modélisation permet de décrire de manière puissante et unifiée, le mode opératoire de l'attaquant qui souhaite analyser un système de détection antivirale pour mieux le contourner. Dans les deux cas, les fonctions booléennes permettent d'identifier les propriétés générales que tout schéma de détec-

tion devrait avoir et évaluer l'effort que doit déployer un attaquant pour contourner les techniques de détection. Ce dernier aspect correspond à un problème d'apprentissage de formules booléennes.

2.2 Modèle mathématique de l'analyse de forme

Le but est ici de définir précisément ce qu'est un schéma de détection ou « signature ». La modélisation aura notamment pour objectif d'appréhender les deux aspects fondamentaux du problème : la défense (une détection efficace) et l'attaque (les auteurs de codes malveillants) ainsi que leur interaction.

2.2.1 Définition d'un schéma de détection

Considérons un fichier quelconque $\mathcal{F}$ de taille n. Il représentera potentiellement un fichier infecté ou directement le code d'un programme malveillant. $\mathcal{F}$ est une séquence d'octets, autrement dit une séquence de n symboles pris dans l'alphabet $\Sigma = \mathbb{N}_{255} = \{0, 1, \ldots, 255\}$. $\mathcal{F}$ est donc un élément de Σ^n.

Nous noterons $\mathcal{S}_{\mathcal{M}}$ un motif de détection de taille s relativement à un code malveillant $\mathcal{M}$. C'est un élément de Σ^s. Ainsi

$$\mathcal{S}_{\mathcal{M}} = \{b_1, b_2, \ldots, b_s\}.$$

Le i-ème octet de $\mathcal{F}$ (respectivement de $\mathcal{S}_{\mathcal{M}}$) sera noté $\mathcal{F}(i)$ ((resp. $\mathcal{S}_{\mathcal{M}}(i)$).

On dit que $\mathcal{F}$ est infecté par $\mathcal{M}$ s'il existe s indices dans $\mathcal{F}$, notés $\{i_1, i_2, \ldots, i_s\}$ tels que

$$\mathcal{F}(i_j) = b_{\sigma(j)} \qquad 1 \leq j \leq s,$$

où σ désigne une permutation des octets de $\mathcal{S}_{\mathcal{M}}$. Cette permutation permet de décrire toutes les modifications ou transformations éventuelles opérées par un auteur de codes malveillants sur $\mathcal{M}$. En effet, des techniques d'obfuscation de code [Fil06b, Chap. 7], de polymorphisme [Fil03, Szor05] peuvent être utilisées pour modifier la structure de $\mathcal{S}_{\mathcal{M}}$ par réarrangement de ses octets. Plusieurs cas sont à considérer :

– la modification de l'indexation des octets de $\mathcal{S}_\mathcal{M}$ (par exemple par insertion de code inutile ou mort (*dummy code*)). Alors nous avons $\sigma = Id_{\mathbb{N}_s^*}$;
– la modification de l'ordre des octets de $\mathcal{S}_\mathcal{M}$ (par l'usage d'obfuscation ou de chiffrement par transposition [2]). Alors $\sigma \neq Id_{\mathbb{N}_s^*}$. Toutefois, le flux d'exécution ré-ordonne les octets de $\mathcal{S}_\mathcal{M}$ pendant l'exécution de $\mathcal{F}$. Les techniques de scan de seconde génération, les métaheuristiques visent à retrouver ou contourner la permutation σ tandis que les techniques de scan plus basiques (dite de première génération) assurent l'identification et la détection proprement dite lors d'une seconde étape.

Il est important de signaler qu'il est plus puissant de considérer des indices dans $\mathcal{F}$ où sont localisés effectivement les octets de $\mathcal{S}_\mathcal{M}$, à une permutation près de ces octets, plutôt que les octets de $\mathcal{S}_\mathcal{M}$ eux-mêmes. Cette approche permet de mieux appréhender celle de tout plagiaire qui souhaiterait produire une variante non détectée à partir d'un code détecté, et ce sans trop d'effort. Il s'intéresse essentiellement aux positions i qu'il lui faudra modifier plutôt qu'à la valeur $\mathcal{F}(i)$ correspondante. Dans un contexte d'analyse en boîte noire, nous noterons $\mathcal{S}_{\mathcal{F},\mathcal{M}}$ l'ensemble des indices $\{i_1, i_2, \ldots, i_s\}$.

Décrivons maintenant formellement l'action d'un détecteur par analyse de forme $\mathcal{D}$. Pour cela définissons les s variables binaires X_j ($1 \leq j \leq s$) comme suit :

$$X_j = \begin{cases} 1 & \text{si } \mathcal{F}(i_j) = b_{\sigma(j)} \\ 0 & \text{autrement.} \end{cases}$$

Cette notation permet de décrire avec précision les modification éventuelles des octets de $\mathcal{S}_\mathcal{M}$ par tout plagiaire tentant de contourner un détecteur donné $\mathcal{D}$. Ainsi, $X_j = 0$ signifie que le plagiaire a modifié $\mathcal{S}_\mathcal{M}(j)$. L'association de tout octet de $\mathcal{S}_\mathcal{M}$ à une variable booléenne permet de considérer l'ensemble booléen $\mathbb{F}_2^{|\mathcal{S}_\mathcal{M}|}$.

Considérons à présent une fonction booléenne $f_\mathcal{M} : \mathbb{F}_2^s \to \mathbb{F}_2$ où $\mathbb{F}_2$ est le corps de Galois à deux éléments. Nous dirons que $\mathcal{D}$ décide que

2. Il existe deux techniques de chiffrement : les procédés de substitution dans lesquels les caractères sont remplacés par d'autres selon une convention secrète et les procédés de transposition, dans lesquels l'ordre des caractères est modifié, également selon une convention secrète. Les deux procédés peuvent être combinés dans le cadre du surchiffrement.

$\mathcal{F}$ est infecté par le code malveillant $\mathcal{M}$, relativement à la *fonction de détection* $f_{\mathcal{M}}$ et au motif $\mathcal{S}_{\mathcal{M}}$ si et seulement si $f_{\mathcal{M}}(X_1, X_2, \ldots, X_s) = 1$. En d'autres termes,

$$f_{\mathcal{M}}(X_1, X_2, \ldots, X_s) = \begin{cases} 1 & \mathcal{F} \text{ est infecté par } \mathcal{M} \\ 0 & \mathcal{F} \text{ n'est pas infecté par } \mathcal{M}. \end{cases}$$

Notons qu'implicitement le recours à la fonction booléenne de détection considère, par un codage approprié, l'ensemble $\mathcal{F}_2^{|\mathcal{S}_{\mathcal{M}}|}$. Nous conserverons cette transformation implicite pour ne pas alourdir les notations, dans la mesure où elle reste intuitive.

Les fonctions de détection seront considérées sous leur forme normale disjonctive (DNF), c'est-à-dire sous la forme d'une union logique (notée $\vee$) de M mintermes constitués de l'intersection logique (notée $\wedge$ ou simplement omis lorsqu'il n'y a pas de risque de confusion) de variables booléennes X_i :

$$f(X_1, X_2, \ldots, X_s) = \bigvee_{j=0}^{M} \Big(\bigwedge_{l \in S_j \subseteq \{1,2,\ldots,s\}} X_{j_l} \Big).$$

Par construction, les variables booléennes n'apparaissent jamais sous la forme $\overline{X_i}$ (négation de X_i). Les fonctions de détection sont par conséquent représentées par des DNF monotones. Nous verrons dans la section 2.3.3, que toute détection par analyse de forme peut être modélisée par une fonction de détection $f_{\mathcal{M}}$ et un ensemble $\mathcal{S}_{\mathcal{M}}$. Cela nous conduit à définir la notion générale *schéma de détection.*

Définition 1 *(Schéma de détection)*
Soit un code $\mathcal{M}$. On appelle schéma de détection de $\mathcal{M}$, la donnée d'une paire $\mathcal{SD} = \{\mathcal{S}_{\mathcal{M}}, f_{\mathcal{M}}\}$. La fonction $f_{\mathcal{M}}$ est appelée fonction de détection *relativement à $\mathcal{M}$. Dans la détection par analyse de forme, $\mathcal{S}_{\mathcal{M}}$ est un ensemble d'octets. Dans l'analyse fonctionnelle (analyse comportementale), $\mathcal{S}_{\mathcal{M}}$ sera un ensemble de fonctions de programmes.*

Remarque. Une base de signatures peut être définie comme un ensemble de schéma de détection. En particulier, les motifs de détection, et par là, la taille des fonctions de détection, peuvent varier d'un schéma de la base à un autre.

Les auteurs de variantes non détectées à partir de codes détectés vont analyser leur schéma de détection pour un ou plusieurs détecteurs donnés. Afin de disposer d'un modèle plus adapté pour la description de l'approche adoptée par le plagiaire, nous considérerons la notion de schéma de non-détection ou *schéma de contournement*.

Définition 2 *(Schéma de contournement)*
Soit un code $\mathcal{M}$. On appelle schéma de contournement de $\mathcal{M}$, la donnée d'une paire $\mathcal{SC} = \{\mathcal{S}_\mathcal{M}, \overline{f_\mathcal{M}}\}$. La fonction $\overline{f_\mathcal{M}}$ est appelée fonction de non détection *ou* fonction de contournement *relativement à $\mathcal{M}$. Dans la détection par analyse de forme, $\mathcal{S}_\mathcal{M}$ est un ensemble d'octets. Dans l'analyse fonctionnelle (analyse comportementale), $\mathcal{S}_\mathcal{M}$ sera un ensemble de fonctions de programmes.*

La fonction de contournement $\overline{f_\mathcal{M}}$ est en fait la négation de la fonction $f_\mathcal{M}$ soit $1 \oplus \overline{f_\mathcal{M}}$. Cette fonction décrit les différentes possibilités de modifications pouvant être effectuées dans les octets composant $\mathcal{S}_\mathcal{M}$ pour contourner un schéma de détection donné. Ces possibilités correspondent aux s-uplets $(x_1, x_2, \ldots, x_s) \in \mathbb{F}_2^s$ pour lesquels la fonction vaut 1. Pour un s-uplet donné, la modification à faire alors est la suivante :

$$\begin{cases} \text{si } x_i = 0 & \text{l'octet } i \text{ de } \mathcal{S}_\mathcal{M} \text{ doit être modifié} \\ \text{si } x_i = 1 & \text{l'octet } i \text{ de } \mathcal{S}_\mathcal{M} \text{ peut être laissé non modifié.} \end{cases}$$

Remarque.- Le code malveillant $\mathcal{M}$ est caractérisé à la fois par le motif de détection $\mathcal{S}_\mathcal{M}$ mais également par la fonction de détection, qui, en quelque sorte, représente le mode de recherche et de validation de $\mathcal{S}_\mathcal{M}$. Ainsi, il est imaginable qu'un fichier donné contienne le motif $\mathcal{S}_\mathcal{M}$ mais que malgré tout, il ne soit pas détecté comme malveillant. C'est là tout l'intérêt de la notion de schéma de détection dans la mesure où la fonction de détection associée, lorsqu'elle n'est pas triviale (voir section 2.3), va limiter les possibilités de fausses alarmes.

L'utilisation d'une fonction booléenne permet d'enrichir la notion de détection virale. Cela permet de caractériser le mode de recherche d'un motif. D'autre part, cet outil permet de considérer une vision unique là où la vision traditionnelle oppose analyse de forme et analyse comportementale. Avec notre modèle unifié, la détection virale considère la notion générale de variables (des octets ou des comportements) qui sont des entrées d'une fonction booléenne, elle-même décrivant la richesse plus ou moins grande de la recherche.

2.2.2 Propriétés des schémas de détection

La question naturelle qui se pose est celle concernant les propriétés que doit avoir tout schéma de détection efficace. La qualité finale d'un antivirus donné en dépend. Ces propriétés permettent également d'évaluer l'offre logicielle dans ce domaine, sur une base que l'on souhaite rigoureuse et reproductible. Actuellement, cette évaluation est un processus subjectif, non reproductible par un tiers extérieur et souvent les résultats d'une évaluation à une autre peuvent grandement varier, pour un même ensemble de produits. Nous allons présenter les propriétés essentielles qu'un schéma de détection de qualité devrait posséder. Ces propriétés concernent d'une part les motifs de détection et d'autre part la fonction de détection.

Entropie et transinformation

Cette propriété décrit l'incertitude qu'un analyste doit affronter lors d'une analyse en boîte noire relativement à un détecteur $\mathcal{D}$ donné, pour identifier (extraire) le schéma de détection $\{\mathcal{S}_\mathcal{M}, f_\mathcal{M}\}$.

Dans ce but, nous utiliserons la fonction *entropie*, définie par C. E. Shannon [Shan48], en théorie de l'information. Soit une variable X pouvant prendre un ensemble fini de valeurs x_i, chacune avec une probabilité p_i. L'incertitude liée à X, encore appelée entropie de X est définie par :

$$H(X) = -\sum_k p_k \log_2(p_k).$$

Plus l'entropie est faible, moins l'incertitude est importante. Ainsi, à l'extrême, quand nous avons $p_i = 1$ pour un i donné (les autres p_j valent 0, pour $j \neq i$) alors $H(X) = 0$. Il n'y a aucune incertitude car on a toujours $X = x_i$. Dans le contexte du problème de l'extraction d'un schéma de détection, la variable X représente un schéma [3] $\{\mathcal{S}_\mathcal{M}, f_\mathcal{M}\}$ et

3. Il s'agit là d'un abus de notation, destiné à simplifier l'approche. En toute rigueur, la variable X représente un ensemble de variables décrivant s, $\mathcal{S}_\mathcal{M}$ et $f_\mathcal{M}$. Autrement dit, on peut écrire $X = (X_1, X_2, \ldots, X_n)$. Alors, soit par un codage approprié, on réunit toutes ces données en un ensemble de valeurs prises par la variable X (type codage de Gödel ; voir [Fil03, Chap. 2]) soit on considère la fonction entropie conjointe $H(X_1, X_2, \ldots, X_n) = -\sum_{(x_1,x_2,\ldots,x_n)} p[X_1 = x_1, \ldots, X_n = x_n] \log_2(p[X_1 = x_1, \ldots, X_n = x_n])$.

tous ses paramètres. En effet, l'analyste n'a *a priori* aucune information sur un quelconque des paramètres du schéma (taille s, les octets de $\mathcal{S}_\mathcal{M}$, le nombre de termes et les termes de $f_\mathcal{M}$).

La difficulté à extraire le schéma de détection complet, dans un processus d'analyse en boîte noire augmente avec $H(\mathcal{S}_{\mathcal{F},\mathcal{M}})$, et si $H(\mathcal{S}_{\mathcal{F},\mathcal{M}}) = 0$, l'analyste n'a à affronter aucune incertitude.

Le fait est que dans notre cas, le calcul de $H(X)$ est très complexe à calculer. C'est la raison pour laquelle nous utiliserons plutôt le concept d'*information mutuelle*, encore dénommée *transinformation*. Si nous considérons que $\mathcal{S}_\mathcal{M}$ et $f_\mathcal{M}$ peuvent être décrits par un ensemble de variables (pour l'analyste), alors nous définissons

$$I(\mathcal{S}_{\mathcal{F},\mathcal{M}}; f_\mathcal{M}, \mathcal{S}_\mathcal{M}) = H(\mathcal{S}_{\mathcal{F},\mathcal{M}}) - H(\mathcal{S}_{\mathcal{F},\mathcal{M}} | f_\mathcal{M}, \mathcal{S}_\mathcal{M}),$$

comme la quantité d'information que $f_\mathcal{M}$ et $\mathcal{S}_\mathcal{M}$ révèlent ensembles au sujet de $\mathcal{S}_{\mathcal{F},\mathcal{M}}$. En d'autres termes, la transinformation décrit et quantifie ce qu'apporte à l'analyste, le fait de disposer de $\mathcal{D}$. La question est alors de déterminer si $\mathcal{D}$ fournit une information à l'analyste et si oui, quelle est la quantité d'information fournie.

Par construction, il est évident que l'on peut de manière symétrique adopter le même formalisme pour un schéma de contournement. Ainsi :

$$I(\mathcal{S}_{\mathcal{F},\mathcal{M}}; f_\mathcal{M}, \mathcal{S}_\mathcal{M}) = I(\mathcal{S}_{\mathcal{F},\mathcal{M}}; \overline{f_\mathcal{M}}, \mathcal{S}_\mathcal{M}).$$

Rigidité

Une fois que l'analyste est parvenu à extraire l'ensemble d'un schéma $\mathcal{R}(\mathcal{S}_\mathcal{M}, f_\mathcal{M})$, pour produire une variante non détectée, relativement à ce schéma, il devra le contourner. Nous définirons la *rigidité* d'un schéma, que nous noterons $\mathcal{R}(\mathcal{S}_\mathcal{M}, f_\mathcal{M})$, comme la difficulté, plus ou moins grande, qu'aura un plagiaire, à le contourner. Cette donnée dépend de manière évidente non seulement de la taille de $\mathcal{S}_\mathcal{M}$ mais également du poids de la fonction de détection $f_\mathcal{M}$:

- plus la taille de $\mathcal{S}_\mathcal{M}$ est importante, plus le nombre de possibilités de modifications, offertes au plagiaire, augmente ;
- moins la fonction de détection possède d'entrées pour laquelle elle vaut 1 (le poids de la fonction), plus le mode de recherche de $\mathcal{S}_\mathcal{M}$ est facile à leurrer.

Ces constatations amènent donc à définir la rigidité par

$$\mathcal{R}(\mathcal{S}_{\mathcal{M}}, f_{\mathcal{M}}) = \frac{\{X = (X_1, X_2, \ldots, X_s) | f_{\mathcal{M}}(X) = 1\}}{s2^s} \quad (2.1)$$

$$= \frac{\{X = (X_1, X_2, \ldots, X_s) | \overline{f_{\mathcal{M}}}(X) = 0\}}{s2^s}. \quad (2.2)$$

Ainsi, nous avons

$$0 \leq \mathcal{R}(\mathcal{S}_{\mathcal{M}}, f_{\mathcal{M}}) < 1,$$

Plus $\mathcal{R}(\mathcal{S}_{\mathcal{M}}, f_{\mathcal{M}})$ est faible, plus le schéma de détection est facile à contourner. À l'extrême, si $\mathcal{R}(\mathcal{S}_{\mathcal{M}}, f_{\mathcal{M}}) = 0$, le code n'est pas référencé dans la base.

Il est intéressant de remarquer que $\mathcal{R}(\mathcal{S}_{\mathcal{M}}, f_{\mathcal{M}})$ est lié uniquement au nombre de configurations d'octets qui peuvent être modifiés (comme $X_i \in \{0, 1\}$). Un plagiaire peut changer chacun des octets présents dans ces configurations et le remplacer par une des quelconques 255 autres valeurs que celle effectivement présente dans le code. Par conséquent, le nombre total de modifications possibles devient extrêmement important. Il croit inversement avec $\mathcal{R}(\mathcal{S}_{\mathcal{M}}, f_{\mathcal{M}})$. Plus précisément, si on note $X = (X_1, X_2, \ldots, X_s) \in \mathbb{F}_2^s$ et si $\mathrm{wt}(X)$ désigne le poids (nombre de bits égaux à 1 dans l'écriture binaire de X), alors le nombre total Δ de modifications possibles, que le plagiaire peut effectuer est donné par

$$\Delta = \sum_{X \in \mathbb{F}_2^s} \overline{f_{\mathcal{M}}}(X).(256^{s-\mathrm{wt}(X)} - 1). \quad (2.3)$$

Efficacité

L'efficacité d'un schéma de détection concerne sa capacité à détecter et à identifier de manière univoque un code malveillant donné connu. Il doit également le faire sans erreur, c'est-à-dire ne pas incriminer à tort un programme sain. Cette dernière contrainte est en fait liée à la probabilité de fausse alarme pour le test statistique associé [Fil06b, Chap. 2]. C'est également lié à des aspects phylogénétiques des codes malveillants tels que définis dans [GGPS98, KWL05].

Des études [KA94] ont montré qu'en règle générale, déterminer la valeur de cette probabilité de fausse alarme est plus difficile qu'il n'y parait. Typiquement, le paramètre $s = |\mathcal{S}_{\mathcal{M}}|$ a une valeur comprise entre

12 et 36 [KA94]. La probabilité pour une séquence de s octets d'apparaitre au hasard est de $\frac{1}{256^s}$. Donc, dès que s est suffisamment grand, cette probabilité théorique tend vers 0 et en théorie toujours, aucun fichier ne peut être détecté par erreur comme infecté. Malheureusement la réalité ne colle pas à la théorie. Comme souligné dans [KA94], la probabilité de trouver une séquence aléatoire de $s = 24$ octets dans un corpus de 500 Mo est de 0.34 alors que la théorie donne une probabilité de 0.85×10^{-49}.

Cet écart vient du fait que les octets dans un fichier ne sont pas des variables aléatoires indépendantes, identiquement distribuées (selon une probabilité $p = \frac{1}{256}$). À titre d'exemple, nous avons $P[b_0 =' M'] = 1$ et $P[b_1 =' Z'|b_0 =' M'] = 1$ dans un exécutable Win32. Il existe de fortes dépendances entre les octets dans un fichier. Un meilleur modèle consiterait à utiliser des processus markovien [4], pour décrire les dépendances fortes des octets entre eux.

Ainsi qu'il est mentionné dans [KA94], "*l'expérience, qu'elle soit humaine ou algorithmique, est un ingrédient essentiel dans le choix d'une bonne signature virale.*" La plupart des antivirus du commerce utilisent des paramètres de taille s relativement efficaces.

Remarque.- L'efficacité d'un schéma de détection dépend principalement de la taille s. C'est la raison pour laquelle, certains antivirus sont plus sujets que d'autres à provoquer des fausses alarmes, du fait qu'ils utilisent des valeurs de s trop petites. Mais la fonction de détection peut avoir, dans le cas de fonctions de détection non triviales, un effet positif sur le taux de fausses alarmes. L'exploration et l'étude des fonctions de détection relativement à cet aspect est un problème ouvert.

Propriétés des fonctions de détection

La fonction de détection joue un rôle important dans un schéma de détection. Elle constitue le mode de recherche du motif proprement dit. Une propriété comme la rigidité montre que le poids de cette fonction (le nombre de ses entrées pour lesquelles la fonction vaut 1) est un

4. Un *processus markovien* est un processus dans lequel, à chaque instant la probabilité d'un état quelconque du système dans le futur dépend seulement de l'état du système à l'instant actuel (t_0) sans dépendre de la manière dont le système a été amené dans cet état.

paramètre essentiel. Alors qu'un poids d'une unité indique une seule possibilité de réalisation, un poids plus important augmente d'autant les possibilités (ou de configurations) de détection. Ce poids détermine également celui de la fonction de non détection, en vertu de l'égalité bien connue, pour une fonction $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$:

$$\mathrm{wt}(f) = 2^n - \mathrm{wt}(\overline{f}).$$

En d'autres termes, si la fonction de détection possède un poids faible (nombre limité de possibilités de détection) la fonction de non détection aura un poids élevé (nombre élevé de possibilités de contournement). Cela permet de donner la définition suivante.

Proposition 1 *Soit un schéma de détection* $\mathcal{SD} = \{\mathcal{S}_\mathcal{M}, f_\mathcal{M}\}$ *et le schéma de contournement* $\mathcal{SC} = \{\mathcal{S}_\mathcal{M}, \overline{f_\mathcal{M}}\}$ *et soit* $s = |\mathcal{S}_\mathcal{M}|$. *Le schéma* $\mathcal{SD}$ *est dit* plus fort *que le schéma* $\mathcal{SC}$ *si et seulement si*

$$2^{s-1} \leq wt(f_\mathcal{M}) \leq 2^s - 1$$

Cette définition permet d'établir un premier critère pour une « bonne » fonction de détection.

Une autre propriété intéressante concerne l'importance relative des variables X_i en entrée de la fonction de détection. Il est essentiel qu'elles aient toutes la même importance sur la valeur de la fonction $f_\mathcal{M}$ (ou de manière équivalente, sur celles de $\overline{f_\mathcal{M}}$). Dans le cas contraire, si une variable était prépondérante (respectivement moins prépondérante) sur les autres, le plagiaire ne manquera pas de tirer parti de cette propriété pour optimiser ses chances de contournement de la détection : il modifiera prioritairement (respectivement en dernier) cette variable. Cela se généralise à un ensemble quelconque de t variables. Nous adopterons la définition suivante :

Définition 3 *Une fonction de détection sera dite* faiblement contournable à l'ordre t *si et seulement si la sortie de la fonction de détection ne dépend statistiquement d'aucun sous-ensemble de taille au plus* t *de variables d'entrées. Une fonction de détection sera dite* fortement contournable à l'ordre t *si et seulement si la fonction de détection dépend statistiquement et identiquement de tout sous-ensemble de taille au plus* t *de variables d'entrées.*

En d'autres termes, aucun sous-ensemble d'au plus t variables d'entrée ne sera plus intéressant à considérer qu'un autre pour modifier le code, en vue de produire une variante non détectée. La différente entre « faiblement » et « fortement » tient au fait que dans le premier cas, il n'existe aucune dépendance vis-à-vis d'un quelconque sous-ensemble de variables de taille au plus t alors que dans le second cas il existe une dépendance mais elle est la même pour tous ces sous-ensembles. Il est assez intuitif de supposer que le premier cas est plus difficilement réalisable que le second. Nous le démontrerons dans la section 2.4.

Notons au passage que si $f_{\mathcal{M}}$ est faiblement contournable à l'ordre t (respectivement fortement contournable à l'ordre t), il en est de même de $\overline{f_{\mathcal{M}}}$.

Cette propriété amène tout naturellement à considérer une classe particulière de fonctions booléennes, très importantes en cryptologie à clef secrète : les fonctions immunes aux corrélations à l'ordre t. Afin d'utiliser cette propriété, nous allons d'abord définir l'outil mathématique de base pour l'étude des fonctions booléennes. Le lecteur pourra consulter [Beau84, Chap. 2], [MvOV97, pp. 207] et [Fil01] pour plus de détails sur cet outil.

Définition 4 *Soit une fonction booléenne sur $\mathbb{F}_2^n$. La* transformée de Walsh-Hadamard *de f est la transformée de Fourier de la fonction signe correspondante, $x \mapsto (-1)^{f(x)}$:*

$$\forall u \in \mathbb{F}_2^n,\ \widehat{\chi_f}(u) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)} (-1)^{<u,x>}$$

où $< ., . >$ désigne le produit scalaire usuel.

La transformée de Walsh permet de caractériser les dépendances statistiques existant entre des sous-ensembles de variables en entrée et la sortie de la fonction. Précisons cela.

Définition 5 *Une fonction booléenne à n variables est dite* sans corrélation à l'ordre t*, ou immune aux corrélations à l'ordre t, si sa distribution de valeurs ne change pas lorsque l'on fixe au plus t entrées.*

Autrement dit, la sortie de la fonction est statistiquement indépendante de tout vecteur $(X_{i_1}, X_{i_2}, \ldots, X_{i_t})$. Par exemple, la fonction

$$f(X_1, X_2, \ldots, X_n) = X_1 \oplus X_2 \oplus \ldots X_n,$$

est sans corrélation à l'ordre $(n-1)$.

En 1988 [XioMass88], un résultat important a permis de caractériser cette propriété l'aide de la transformée de Walsh.

Proposition 2 *[XioMass88] La fonction booléenne f à n variables est sans corrélation à l'ordre t si et seulement si elle vérifie*

$$\widehat{\chi_f(u)} = 0 \quad \forall u \in \mathbb{F}_2^n,\ 1 \leq wt\,^5(u) \leq t.$$

Nous pouvons maintenant établir la proposition suivante.

Proposition 3 *Une fonction $f_{\mathcal{M}}$ est faiblement contournable à l'ordre t si et seulement si elle est sans corrélation à l'ordre t. Une fonction $f_{\mathcal{M}}$ est fortement contournable à l'ordre t si et seulement si*

$$\forall u \in \mathbb{F}_2^n \text{ tel que } 1 \leq wt(u) \leq t \qquad \widehat{\chi_f(u)} \text{ est une constante.}$$

Preuve.
Evidente par définition de l'immunité aux corrélations. Notons que cette propriété est également valable pour la fonction de non détection $\overline{f_{\mathcal{M}}}$. En effet, nous avons

$$\widehat{\chi_f(u)} = -\widehat{\chi_{\overline{f}}(u)}.$$

■

Nous verrons dans la section 2.4.1 un exemple de fonction pour chacune de ces deux propriétés.

2.3 Le problème de l'extraction

2.3.1 L'extraction de schémas de détection

Le problème de l'extraction de motifs de détection a été étudié par Christodorescu et Jha [CJ04]. L'algorithme qu'ils ont proposé ne considère qu'un cas trivial de fonctions de détection et par conséquent, le problème général de l'extraction de schéma n'est pas résolu. Nous allons étudier deux algorithmes d'extraction, le second permettant de résoudre

5. $wt(u)$ désigne le poids de u, c'est-à-dire le nombre de bits valant 1 dans son écriture binaire.

de manière générale ce problème, et ce quelle que soit la fonction de détection utilisée.

Soit un détecteur $\mathcal{D}$ donné, l'objectif est de retrouver le schéma de détection complet, pour un code malveillant $\mathcal{M}$, relativement à $\mathcal{D}$. Cette reconstruction est réalisée par une analyse en boîte noire. Aucune technique de rétro-ingéniérie n'est utilisée. Avec les notations précédentes, il s'agit donc de retrouver la fonction de non détection $\overline{f_{\mathcal{M}}}$ et les indices des octets impliqués dans le motif lui-même. Comme le fichier examiné $\mathcal{F}$ est le code malveillant lui-même, nous utiliserons la notation $\mathcal{S}_{\mathcal{M},\mathcal{M}}$ au lieu de $\mathcal{S}_{\mathcal{F},\mathcal{M}}$.

Nous considérerons la fonction de non détection plutôt que la fonction de détection pour les raisons suivantes :

- le point de vue de l'attaquant est plus intéressant pour évaluer la résistance d'un antivirus, à l'analyse en boîte noire ;
- extraire la fonction de détection aurait nécessité de calculer ensuite sa négation pour obtenir la fonction de non détection. Malheureusement, le calcul de la négation d'une forme disjonctive normale a une complexité en pire cas, exponentielle[6]. En conséquence, l'algorithme E-2 extraira directement $\overline{f_{\mathcal{M}}}$.

2.3.2 Approche naïve : Algorithm E-1

Un premier algorithme a été conçu pour traiter l'instance la plus fréquente du problème de l'extraction. Si on le compare à l'algorithme proposé par Christodorescu et Jha [CJ04], il peut sembler, à première vue moins efficace et plutôt naïf. Toutefois, l'expérience pratique montre que ce n'est pas le cas. Cet algorithme « naïf », bien au contraire, est parvenu systématiquement à résoudre le problème de l'extraction, pour l'instance concernée, là où celui de Christodorescu et Jha avait échoué pour certains cas dans lesquels, le motif de détection est très dispersé dans le code.

Considérons le code $\mathcal{M}$ de taille n. Le pseudo-code de ce premier algorithme est donné en figure 2.1. L'algorithme modifie successivement chaque octet de $\mathcal{M}$ (remplacé par un octet nul) et soumet le code ainsi

6. Ce calcul revient à transformer une forme disjonctive normale en sa forme conjonctive normale. Cette opération a une complexité exponentielle [Papadim95, Theorem 4.1].

Entrée : Un code malveillant $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$, un détecteur $\mathcal{D}$ et une identification σ pour $\mathcal{M}$.
Sortie : Le motif $\mathcal{S}_{\mathcal{M},\mathcal{M}}$ (indices).

$\mathcal{S}_{\mathcal{M},\mathcal{M}} \leftarrow \{\}$
Pour i = 1 à n faire
    modifier seulement l'octet m_i in $\mathcal{M}$
    Si $\mathcal{D}(\mathcal{M}) \neq \sigma$ alors
        $\mathcal{S}_{\mathcal{M},\mathcal{M}} \leftarrow \mathcal{S}_{\mathcal{M},\mathcal{M}} \cup \{i\}$
    Fin si
Fin pour
retourner $\mathcal{S}_{\mathcal{M},\mathcal{M}}$

TABLE 2.1 – Algorithme d'extraction de schéma de détection E-1

modifié au détecteur $\mathcal{D}$. Si ce dernier détecte toujours $\mathcal{M}$ (c'est-à-dire $\mathcal{D}(\mathcal{M}) = \sigma$), cela implique que l'octet modifié n'est pas impliqué dans le schéma de détection. Dans le cas contraire, l'indice de l'octet appartient à $\mathcal{S}_{\mathcal{M},\mathcal{M}}$.

La complexité de l'algorithme 2.1 est linéaire en la taille de $\mathcal{M}$ soit $\mathcal{O}(n)$. Le principal intérêt de cet algorithme est sa capacité à extraire le motif $\mathcal{S}_{\mathcal{M},\mathcal{M}}$, quelle que soit sa structure (octets dispersés, faible nombre d'octets...). En revanche, son efficacité est limitée à une unique fonction de détection – certes la plus fréquemment utilisée – dont la DNF est donnée par :

$$f_{\mathcal{M}}(X_1, X_2, X_3, \ldots, X_s) = X_1 \wedge X_2 \wedge X_3 \wedge \ldots \wedge X_s.$$

Cette fonction de détection correspond à la technique de détection la plus fréquemment utilisée : recherche de « signatures » simples (technique basique). C'est la fonction ET. Contrairement à ce que prétendent la plupart des éditeurs d'antivirus, les résultats prouvent que cette technique est encore très largement utilisée. Des résultats détaillés pour certaines variantes de la famille *I-Worm.Bagle* sont présentés dans [Fil06a].

Remarque.- La fonction de non détection correspondant à la fonction ET est très simple à calculer, en utilisant les règles élémentaires du calcul booléen. Nous avons :

$$\overline{f_{\mathcal{M}}}(X_1, X_2, X_3, \ldots, X_s) = X_1 \vee X_2 \vee X_3 \vee \ldots \vee X_s.$$

C'est la fonction OU. Elle exprime qu'en modifiant un quelconque des octets situés aux indices de $\mathcal{S}_{\mathcal{M},\mathcal{M}}$, cela suffit à rendre le code indétectable.

2.3.3 Approche par apprentissage de DNF - Algorithme E-2

L'extraction de certains schémas de détection n'est pas réalisable avec l'algorithme E-1. Ce sont les schémas pour lesquels, la fonction de détection est différente de la fonction ET. Nous allons donc considérer un algorithme général permettant de résoudre le problème de l'extraction d'un schéma de détection (ou de manière équivalente d'un schéma de contournement). Cet algorithme utilise des techniques d'apprentissage de formules booléennes. Rappelons tout d'abord quelques concepts de la théorie de l'apprentissage. Le lecteur intéressé pourra consulter [KV94] pour une présentation détaillée de cette théorie.

Apprentissage de concepts booléens

Nous nous intéresserons à l'apprentissage de *concepts booléens* dans lequel celui qui « apprend » – en d'autres termes un analyste en boîte noire dans notre contexte – a pour objectif d'inférer comment une fonction cible inconnue – la fonction de non détection dans notre cas – classe les éléments d'un espace donné selon la valeur (0 ou 1) que prend cette fonction pour un nombre donné d'éléments de cet espace.

Le *domaine d'instances* $\mathcal{X}$ est l'ensemble de tous les éléments (ou *instances*) possibles qu'il faut classer. Dans le contexte de la détection virale, nous considérerons l'hypercube booléen $\{0,1\}^n$, comme domaine de référence. Cela correspond aux différentes entrées possibles pour la fonction de non détection, avec les notations de la section 2.2.1. Un *concept booléen* ou*concept* est une fonction booléenne définie sur un domaine $\mathcal{X}$. Une *classe de concept* $\mathcal{C}$ est une famille de sous-ensembles de

$\mathcal{X}$. En d'autres termes, $\mathcal{X} \subseteq \mathcal{P}(\mathcal{X})$. Dans notre contexte, $\mathcal{C}$ décrit l'ensemble de toutes les formes normales disjonctives de fonctions de non détection [7]. Maintenant, tout $x \in \mathcal{X}$ est classé selon son appartenance à un *concept cible* $f \in \mathcal{C}$ – la fonction de non détection $f_{\mathcal{M}}$. Un élément $x \in \mathcal{X}$ est un exemple *positif* de f si $f(x) = 1$ et un exemple *négatif* dans le cas contraire. Cette méthode d'apprentissage est dénommée *modèle d'apprentissage par requêtes* (*Query Learning Model*).

Algorithme général d'extraction

Dans notre contexte, la formule booléenne à apprendre [8] est la fonction de non détection $f_{\mathcal{M}}$ sous sa forme disjonctive normale. Chaque variable X_i désigne un octet du code malveillant $\mathcal{M}$ avec $i = 1, 2, \ldots, n$ ($n = |\mathcal{M}|$). La forme disjonctive normale est alors la réunion de conjonctions de variables booléennes pouvant être sous la forme « vraie » (notée X_i) ou « fausse » (notée $\overline{X_i}$).

Il est intéressant de préciser un point important. Le problème de l'apprentissage de la formule DNF est depuis longtemps un problème ouvert même si des techniques efficaces d'apprentissage ont été imaginées pour certaines classes particulières de formules DNF. Toutefois, la complexité en mémoire et en temps de tout algorithme d'apprentissage dépend de manière évidente de celle du concept cible sous-jacent. Dans le cas de formules DNF, elle dépend du nombre de termes, compris entre 0 (la fonction constante nulle) à 2^n (la fonction constante $f(x) = 1$). La fonction de détection ET que nous avons considérée dans la section 2.3.2 contient un seul terme. Cela explique pourquoi l'algorithme E-1 est bien mieux adapté que l'algorithme E-2 (dont la compléxité n'est plus linéaire) pour ce cas très particulier mais néanmoins le plus répandu, de fonction. L'algorithme général d'extraction E-2 est présenté en figure 2.2. Par souci de clarté, nous donnons ici une version non récursive. La notation $X = 1_I(X_1, X_2, \ldots, X_n)$ la construction d'une terme logique de DNF, en utilisant la fonction caractéristique relativement à un intervalle I d'indices. Chaque variable X_i ou sa négation apparait

7. Rappelons qu'une fonction peut avoir plusieurs formes disjonctives normales équivalentes. Ainsi, les DNFs $x_1 \vee x_2$ et $x_1 x_3 \vee x_1 \overline{x_3} \vee x_2$ décrivent la même fonction. La première est la forme simplifiée dite minimale.

8. Le terme « apprendre » signifie que l'on cherche à trouver les termes de cette formule.

Entrée : Un code malveillant $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$, un détecteur $\mathcal{D}$ et une identification σ pour $\mathcal{M}$.
Sortie : Le motif $\mathcal{S}_{\mathcal{M},\mathcal{M}}$ (indices) et la fonction de non détection $\overline{f_{\mathcal{M}}}$ (DNF).

$\mathcal{S}_{\mathcal{M},\mathcal{M}} \leftarrow \{\}$; $\text{DNF}_{f_{\mathcal{M}}} \leftarrow \{\}$
$S \leftarrow 2^{\lfloor \log_2(n) \rfloor + 1}$; $S' \leftarrow 2^{\lfloor \log_2(n) \rfloor + 1}$
Tant que $S > 0$ faire
    $S \leftarrow \frac{S}{2}$; $q \leftarrow \frac{S'}{S}$
    Pour i = 0 à q -1 faire
        binf $\leftarrow i \times S$; bsup $\leftarrow$ binf $+ S$
        modifier les octets dans $I = [\text{binf}, \text{bsup}[$ in $\mathcal{M}$
        Si $\mathcal{D}(\mathcal{M}) \neq \sigma$ alors
            $\mathcal{S}_{\mathcal{M},\mathcal{M}} \leftarrow \mathcal{S}_{\mathcal{M},\mathcal{M}} + I$
            $X = 1_I(X_1, X_2, \ldots, X_n)$
            $\text{DNF}_{f_{\mathcal{M}}} \leftarrow \text{DNF}_{f_{\mathcal{M}}} \cup X$
        Fin si
    Fin pour
Fin tant que
$\mathcal{S}_{\mathcal{M},\mathcal{M}} \leftarrow$ CombinatorialMinimize($\mathcal{S}_{\mathcal{M},\mathcal{M}}$)
$\text{DNF}_{f_{\mathcal{M}}} \leftarrow$ LogicalMinimize($\text{DNF}_{f_{\mathcal{M}}}, \mathcal{S}_{\mathcal{M},\mathcal{M}}$)

TABLE 2.2 – Algorithme d'extraction de schéma de détection E-2

dans le terme en vertu de la règle suivante :

$$X_i = \begin{cases} \overline{X_i} & \text{si } i \in I \\ X_i & \text{si } i \notin I \end{cases}$$

L'algorithme E-2 se compose de trois parties :

1. La première phase est la phase initiale d'apprentissage. Elle consiste à modifier des portions d'octets contigüs du code $\mathcal{M}$. Cette modification se fait par approche dichotomique, en $\log_2(n)$ pas. Les portions sont de taille décroissante, égale à une puissance de deux. Comme dans l'algorithme de Christodorescu et Jha [CJ04], le but

premier est d'identifier les octets impliqués dans le motif de détection $\mathcal{S}_{\mathcal{M},\mathcal{M}}$. Cette phase permet de trouver quelques termes de la DNF de la fonction de non détection. Cette phase a une complexité de $\mathcal{O}(2n)$ (boucle WHILE).

2. Une phase de simplication combinatoire, appelée COMBINATORIALMINIMIZE, va éliminer les redondances entre les termes initiaux de la DNF. L'ensemble $\mathcal{S}_{\mathcal{M},\mathcal{M}}$ produit par la phase précédente contient en général quelques termes dont les intervalles supports (voir notation précédente) correspondants sont inclus dans d'autres intervalles. A titre d'exemple, le terme $X_1X_2X_3X_4$ et le terme X_1X_2 ont pour intervalle support respectif $[1,4]$ et $[1,2]$. Le second est inclu dans le premier. Cela signifie que les variables X_3 et X_4 ne jouent aucun rôle dans le motif de détection. Seul le terme X_1X_2 est conservé. Cette étape va donc simplifier d'un point de vue combinatoire, l'ensemble $\mathcal{S}_{\mathcal{M},\mathcal{M}}$. Il s'agit de rechercher les éléments minimaux dans un ensemble partiellement ordonné par la relation d'inclusion entre intervalles. Cette étape possède une complexité en pire cas en $\mathcal{O}(t\log(t)+tn)$ (tri puis comparaison de termes consécutifs) avec $t=|\mathcal{S}_{\mathcal{M},\mathcal{M}}|$ en entrée de la procédure COMBINATORIALMINIMIZE. L'ensemble produit est de taille s.

3. Une phase LOGICALMINIMIZE dont le rôle est
   - de finaliser l'apprentissage par une recherche exhaustive sur tous les s-uplets de variables. Avec les notations précédentes, à chaque s-uplets de $\mathbb{F}_2^s$ correspond une configuration de modifications des octets de $\mathcal{S}_{\mathcal{M},\mathcal{M}}$. Le code $\mathcal{M}$ est alors modifié selon chacune de ces configurations, puis testé vis-à-vis du détecteur $\mathcal{D}$. Si le code, pour une configuration donnée, n'est plus détecté par $\mathcal{D}$, le terme logique correspondant est ajouté à la formule DNF ;
   - sur la DNF finale, une étape de minimisation logique est effectuée. En effet, comme à l'issue de la phase finale d'apprentissage, la DNF contient des termes redondants, il est nécessaire d'éliminer ces redondances d'un point de vue logique en utilisant les règles du calcul booléen [DKF63, Chap. 9] et [Mich00]. L'algorithme utilise la méthode de Quine-McCluskey [McClus56, Quine52, Quine53]. Le problème de la minimisation logique est

un problème NP-dur [Mich00, Chap. 5.8.3]. Sa complexité est en $\mathcal{O}(s2^s)$ [Weg87].

La complexité de la procédure LogicalMinimize est finalement est en $\mathcal{O}(2^s + s2^s)$ si $s = |\mathcal{S}_{\mathcal{M},\mathcal{M}}|$ en entrée de cette procédure.

Nous avons donc le résultat suivant.

Théorème 1 *L'algorithme d'extraction d'un schéma de contournement d'un code $\mathcal{M}$ de taille n dont le motif de détection est de taille s, donné en figure 2.2 a une complexité en $\mathcal{O}(sn+s2^s)$ et une complexité mémoire en $\mathcal{O}(wt(f_{\mathcal{M}}))$.*

Preuve.
Avec la discussion et les notations précédentes, la complexité générale en pire cas est en $\mathcal{O}(2n + t\log(t) + tn + s^2 2^s)$. Les termes $2n$ et t sont négligeables par rapport à 2^s et nous avons également $t \cong s$ (ce qui est confirmé par l'expérience). En ce qui concerne la complexité en mémoire, elle dépend de manière évidente du nombre de termes dans la fonction de détection. D'où le résultat. ■

Il est intéressant de noter que la complexité mémoire est également un paramètre à considérer. La boucle While produit un ensemble $\mathrm{DNF}_{f_{\mathcal{M}}}$ dont la taille est bornée par le nombre maximal possible de termes, soit $2n$. La procédure LogicalMinimize produit, elle, un ensemble $\mathrm{DNF}_{f_{\mathcal{M}}}$ d'une taille qui peut être plus importante selon la fonction de non détection $f_{\mathcal{M}}$. Les résultats expérimentaux obtenus sur la famille *I-Worm.Bagle* n'a pas montré d'augmentation significative de cette taille.

Remarques.-

1. L'algorithme E-1 est un cas particulier de l'algorithme E-2, quand la fonction de détection est la fonction logique et. Ce cas étant le plus fréquent, comme l'ont montré les nombreux tests effectués, l'utilisation de l'algorithme E-1 est un meilleur choix préalable, dans la mesure où cela permet de faire l'économie des étapes de minimisations combinatoire et logique.
2. L'apprentissage d'une formule DNF monotone (cas de la fonction de détection) possède un complexité bien meilleure lorsque l'on utilise l'algorithme d'Angluin [Angluin88]. En effet, en utilisant la méthode *membership and equivalence queries*, cet algorithme

apprend une formule monotone, sur le domaine $\{0,1\}^n$, composée de m termes avec seulement $\mathcal{O}(nm)$ requêtes. Mais pour obtenir la fonction de non détection correspondante, la négation de la formule obtenue doit être calculée. Ce calcul possède une complexité en pire cas, exponentielle.

3. L'algorithme E-2 trouve la DNF exacte de la fonction de non detection alors que généralement les méthodes d'apprentissage peuvent ne fournir qu'une DNF non simplifiée.

2.3.4 Exemples de fonctions de détection

A toute technique de détection par analyse de forme, correspond une fonction booléenne de détection (ou de manière équivalente une fonction de non détection). Pour illustrer cela, et afin de montrer qu'il existe bien d'autres fonctions de détection que celle triviale (la fonction ET), nous allons donner quelques exemples simples.

Détection par caractères génériques (*Wildcards*)

Cette forme de détection considère généralement des motifs dans lesquels les indices d'octets peuvent en partie être variables. Considérons le motif de détection suivant emprunté à [Szor05, Section 11.1.2].

```
..... 07BB ??02 %3 33C9 ......
```

Le symbole `??` indique de négliger l'octet placé à cet endroit tandis que la chaîne `%3 33` demande de rechercher le caractère `33` dans l'un quelconque des trois octets suivants. Ce mode de recherche correspond en fait à la fonction de détection décrit par la DNF suivante (nous ne donnons que l'extrait pertinent de la DNF) :

$$\begin{aligned}
\ldots \quad & (X_i = 07) \wedge (X_{i+1} = BB) \wedge (X_{i+3} = 02) \wedge (X_{i+4} = 33) \wedge (X_{i+5} = C9) \\
\vee \quad & (X_i = 07) \wedge (X_{i+1} = BB) \wedge (X_{i+3} = 02) \wedge (X_{i+5} = 33) \wedge (X_{i+6} = C9) \\
\vee \quad & (X_i = 07) \wedge (X_{i+1} = BB) \wedge (X_{i+3} = 02) \wedge (X_{i+6} = 33) \wedge (X_{i+7} = C9) \\
\ldots &
\end{aligned}$$

Techniques de non-coïncidences partielles

Cette technique a été développée par IBM dans le cadre de ses recherches sur les scanners antiviraux. La techique dite de non-coïncidences partielles (ou technique *mismatch*) autorise, dans un motif de détection de taille s, que μ octets prennent une valeur quelconque. Par exemple, considérons le motif de détection `01 02 03 04` avec une valeur de non-coïncidences partielles de $\mu = 2$ (cet exemple a été pris dans [Szor05, Section 11.1.3]). Alors la fonction de détection correspondante (extrait de la DNF) est donnée par :

$$\begin{aligned} \ldots \quad & (X_i = 01) \wedge (X_{i+1} = 02) \vee (X_i = 01) \wedge (X_{i+2} = 03) \\ \vee \quad & (X_{i+1} = 02) \wedge (X_{i+2} = 03) \vee (X_i = 01) \wedge (X_{i+3} = 04) \\ \vee \quad & (X_{i+1} = 02) \wedge (X_{i+3} = 04) \vee (X_{i+2} = 03) \wedge (X_{i+3} = 04) \end{aligned}$$

D'une manière générale, pour un motif de taille s et de valeur de non-coïncidences partielles μ, la formule DNF contient $\binom{s}{\mu}$ termes, chacun d'entre eux ayant $s-\mu$ variables (il s'agit d'une $(s-\mu)$-DNF monotone).

Identification quasi-exacte

Cette méthode est utilisée pour augmenter la qualité de la détection. Plutôt que de considérer un seul schéma de détection, on en considère deux (voire éventuellement plus) [Szor05, Section 11.2.3]. Il est alors facile à démontrer que si on utilise les schémas $(\mathcal{S}^1_{\mathcal{M},\mathcal{M}}, f^1_{\mathcal{M}})$ et $(\mathcal{S}^2_{\mathcal{M},\mathcal{M}}, f^2_{\mathcal{M}})$, il est possible de le décrire par un schéma unique $(\mathcal{S}_{\mathcal{M},\mathcal{M}}, f_{\mathcal{M}})$ tel que $\mathcal{S}_{\mathcal{M},\mathcal{M}} = \mathcal{S}^1_{\mathcal{M}} \cup \mathcal{S}^2_{\mathcal{M},\mathcal{M}}$ et $f_{\mathcal{M}} = f^1_{\mathcal{M}} \vee f^2_{\mathcal{M}}$ ou $f_{\mathcal{M}} = f^1_{\mathcal{M}} \wedge f^2_{\mathcal{M}}$ selon les cas (à une minimisation logique près car l'union logique ou l'intersection logique peuvent produire des DNF non minimales).

Notons que ce cas couvre également celui où plusieurs moteurs d'analyse de forme sont utilisés. Plusieurs antivirus combinent en effet un moteur heuristique à un moteur classique.

Autres techniques

Les autres techniques par analyse de forme comme la technique *des signets* (*bookmarks techniques*), le scan de motifs utiles (*smart scan-*

ning), la technique du squelette (*skeleton detection*), la méthode du décrypteur statique (*static decryptor techniques*), les heuristiques reposant sur la forme..., parmi de nombreuses autres (voir [Szor05, Chap. 11]) peuvent être modélisées par des schémas de détection similaires aux précédents. Mais pour ces techniques, en général plus élaborées, quelques remarques doivent êtres faites :

- la taille s du motif de détection est plus importante. Le motif s'étend plus largement sur tout le code ;
- la fonction de détection est en général plus complexe (le nombre de termes est proche de 2^s). Cependant, il n'est pas sûr que la fonction de non détection soit tout aussi complexe. C'est là également un problème ouvert ;
- des considérations de nature combinatoire sur la structure de la DNF de la fonction de détection peuvent également intervenir, en plus du poids de cette dernière.

En ce qui concerne les techniques de contrôle de parité (*checksum*) ou de hachage partiel (code de redondance cyclique, fonction de hachage... voir [Szor05, Chap. 11]), notre approche reste la même mais plutôt que de considérer les octets du code, il est préférable de voir ce dernier sous une forme binaire. Alors, toute valeur de parité (*checksum*) ou tout haché partiel peut être décrit par un ensemble de fonctions booléennes [Fil02]. La fonction de détection résultante est alors la conjonction logique de ces fonctions booléennes (à une étape de minimisation logique près). L'exemple (trivial) suivant illustre notre propos.

Exemple 1 *Soient deux octets, vus comme des suites binaires :*

$$O_i = (b_7^i, b_6^i, \ldots, b_0^i) \text{ et } O_j = (b_7^j, b_6^j, \ldots, b_0^j).$$

Considérons le contrôle de parité sur deux bits (p_1, p_0)*, définis comme suit :*

$$p_1 = \bigoplus_{k=0}^{7} b_k^i \text{ et } p_0 = \bigoplus_{k=0}^{7} b_k^j.$$

Un élément de détection (conjointement avec d'autres) sera que (p_1, p_0) *diffèrent de valeurs attendues (la fonction de détection correspondante vaut alors 0). La DNF de la fonction de détection sera alors définie par*

$$f = p_1 \wedge p_0.$$

Enfin des techniques plus élaborées et qui intègrent des aspects quantitatifs à côté de caractéristiques essentiellement qualitatives ou structurelles, comme l'analyse spectrale, par exemple [Fil03, Chap. 5] peuvent également être décrites par ce modèle. Cependant, comme pour les techniques de contrôle de parité, la complexité du schéma est telle qu'il est impossible de représenter la fonction de détection de par sa complexité mémoire. La modélisation statistique des techniques antivirales permet d'appréhender de manière beaucoup plus puissante ces techniques très élaborées de détection [Fil06b, Chap. 2].

2.4 Schéma de détection sécurisé

L'étude précédente montre qu'aucun antivirus testé n'offre de résistance en matière d'extraction de schéma de détection ou de contournement. L'action de tout plagiaire s'en trouve donc facilitée. Il est donc intéressant de considérer les solutions qui permettraient de limiter cette action. Nous allons présenter un schéma de détection sécurisé, totalement paramétrable en fonction des besoins, et qui offre une sécurité très satisfaisante contre l'extraction de schéma sans requérir des ressources significativement plus importantes que celles utilisées par les antivirus actuels.

L'extraction d'un schéma, pour cette solution sécurisée, reste malgré tout possible, mais elle n'est réalisable que dans le cas où un groupe important de pirates s'unissent et partagent leurs ressources [9]. Toutefois, même dans ce cas là, la complexité de l'algorithme E-2, pour les paramètres utilisés, rendra cette collusion illusoire. Il en résulte que toute tentative de produire des variantes non détectées à partir de variantes connues est, en pratique, condamnée à l'échec.

9. Une autre solution serait pour un pirate d'émuler plusieurs environnements afin de simuler cette collusion. Le schéma proposé non seulement résiste à cette éventualité mais également l'interdira par une implémentation adéquate (voir la section 2.4.1).

2.4.1 Constructions combinatoires et probabilistes

Principe général

L'approche générale consiste à considérer un motif de détection principal de taille s octets (répartis dans tout le code). Chaque fois qu'un moteur d'analyse de forme est sollicité par un processus de détection, seul un sous-motif de taille k est utilisé avec les contraintes suivantes :

- le paramètre k doit être relativement faible par rapport au paramètre s ;
- chaque sous-motif est choisi aléatoirement ;
- tout sous-motif est fixe pour un utilisateur et une machine donnée. Son choix dépendra de données caractérisant de manière unique un environnement ;
- le schéma de détection ou de contournement ne peut être reconstruit, même partiellement (par exemple seul l'ensemble $\mathcal{S}_{\mathcal{M},\mathcal{M}}$) à moins de disposer d'au moins τ sous-motifs ;
- le nombre π de sous-motifs doit être relativement important ainsi que le paramètre τ.

Ces contraintes – complètement paramétrables, comme nous allons le voir – ont été choisies dans le but de maximiser l'incertitude et les efforts du pirate confronté au problème de l'extraction de schéma. Autrement dit, il ne pourra reconstruire un schéma de détection ou de contournement à moins de réunir des conditions rédhibitoires en pratique. Enfin, s'il parvient à produire une variante non détectée avec un ordinateur donné, relativement à un détecteur donné $\mathcal{D}$ et à un sous-motif donné, la probabilité de rester détectable sur d'autres machines relativement à tout autre sous-motif, doit rester élevée. Il s'ensuit, sous ces contraintes, que la prolifération de variantes reste d'une portée très limitée.

Le problème principal a été de trouver des objets combinatoires réalisant les contraintes sus-mentionnées. Les meilleurs candidats sons sans aucun doute les objets dénommés *designs combinatoires*[10] [BJL99, ColDin96]. Ces structures particulières présentent de très intéressantes propriétés et caractéristiques permettant de satisfaire aux contraintes souhaitées. De plus, leur implémentation et leur mise en œuvre requiert des ressources temps/mémoire relativement limitées.

10. Le terme est difficile à traduire et ne semble pas avoir d'équivalent simple en français. Nous garderons le terme anglo-saxon.

La quatrième contrainte suggérait fortement d'utiliser des structures de partage de secret ou des (π, τ)-*schéma à seuil* [MvOV97, Chap. 12]. Dans notre contexte, le secret à partager est le motif de détection complet $\mathcal{S}_{\mathcal{M},\mathcal{M}}$ et les différentes *parts* de secret auraient été les différents sous-motifs. Malheureusement, les structures de partage de secret ou de schéma à seuil, qui ont été proposées jusqu'ici, ne concernent que des cas dans lesquels les parts sont des nombres [11]. Dans notre cas, les parts sont des objets plus complexes que des nombres (typiquement des ensembles de nombres). Quelques constructions ont été proposées récemment pour étendre les schémas de partage de secret à des structures complexes comme les graphes [KK03]. Toutefois, malgré l'intérêt de ces généralisations, il reste pour le moment théorique. Elles requièrent encore trop de ressources, notamment en mémoire pour être viables dans des antivirus destinés à un usage commercial.

Enfin, notons que la troisième contrainte peut également aider les enquêteurs dans le cadre d'expertises ayant pour but d'incriminer ou d'innocenter l'auteur d'une nouvelle variante non détectée.

Description technique du schéma sécurisé

Nous devons considérer deux aspects pour ce schéma : les objets combinatoires décrivant le motif $\mathcal{S}_{\mathcal{M}}$ lui-même et la fonction de détection $f_{\mathcal{M}}$ (et par conséquent, la fonction $\overline{f_{\mathcal{M}}}$ également). Dans ce qui suit, nous ne rappellerons que les concepts de base concernant les objets combinatoires utilisés. Le lecteur qui souhaiterait les étudier de manière plus détaillée consultera [BJL99, ColDin96].

Les objets combinatoires Nous avons considéré, d'une manière générale des $2-(s,k,\lambda)$ designs (également connus sous le nom de *Balanced Incomplete Block Designs* (BIBD) ou *Designs par blocs, équilibré et complet*). Nous ne rappellerons que la définition de ces objets ainsi que leurs propriétés les plus intéressantes.

Définition 6 *Un* Balanced Incomplete Block Design *(BIBD) est une paire* $(\mathcal{V}, \mathcal{B})$ *où* $\mathcal{V}$ *est un ensemble contenant* v *éléments, ou* points *et* $\mathcal{B}$ *est une famille de* b *parties de* $\mathcal{V}$ *(ou* blocs*, chacune ayant une taille*

11. Il est intéressant de noter que ces structures sont également réalisables par des *designs combinatoires*.

constante k. Cette paire est telle que tout point de $\mathcal{V}$ est contenu dans exactement r blocs et telle que tout paire de points de $\mathcal{V}$ est contenue dans exactement λ blocs. Les valeurs v, b, r, k, λ sont les paramètres du BIBD.

Tout BIBD satisfait alors les propriétés suivantes :

- *Un BIBD existe si et seulement si $vr = bk$ et si $r(k-1) = \lambda(v-1)$.*
- $r = \frac{\lambda(v-1)}{k-1}$.
- $b = \frac{vr}{k}$.

Dans notre contexte, nous avons $\mathcal{S}_{\mathcal{M}} = \mathcal{V}$, $s = v$, $\pi = b = \frac{\lambda v(v-1)}{k(k-1)}$ et $\mathcal{B}$ décrit la famille de sous-motifs, notés $\mathcal{S}^i_{\mathcal{M}}$ pour $i = 1, \ldots, \pi$.

La fonction de détection Nous avons considéré pour ce schéma une fonction booléenne totale à s variables $f : \mathbb{F}_2^s \rightarrow \mathbb{F}_2$ de poids égal à 2^{s-1}. Pour chaque sous-motif de taille k, nous considérons la restriction f^i de f au sous-motif $\mathcal{S}^i_{\mathcal{M}}$, en fixant à une valeur constante les variables X_i non présentes dans ce sous-motif. Ainsi, chaque fonction de détection partielle f^i est de poids 2^{k-1}. D'un point de vue pratique, le choix de la fonction, d'un point de vue structurel (répartition des entrées à valeur non nulle) est intimement lié au choix des octets (indices) contenus dans les $\mathcal{S}^i_{\mathcal{M}}$.

Ainsi, le poids de la fonction totale f et des fonctions partielles f^i est optimal. En effet, ces valeurs maximisent l'effort que l'analyste doit faire pour extraire la fonction de non détection (compte-tenu de l'étape LOGICALMINIMIZE dans l'algorithme E-2). La fonction retenue est la fonction linéaire (notée, pour limiter l'espace, sous sa forme algébrique normale).

$$f(X_1, X_2, \ldots, X_s) = X_1 \oplus X_2 \oplus \ldots X_s.$$

L'intérêt de cette fonction, outre ses propriétés intéressantes, tient au fait qu'elle peut être implémentée de manière très compacte. Notons que du point de vue de l'analyste, retrouver la forme algébrique normale (compacte) précédente, ne peut se faire qu'à partir de la DNF produite par l'algorithme E-2. Cette conversion a une complexité exponentielle en $\mathcal{O}(2^s)$. Elle n'est également possible qu'en réalisant une collusion de taille τ.

Proposition 4 *La fonction $X_1 \oplus X_2 \oplus \ldots X_s$ est faiblement contournable à l'ordre $s-1$.*

Preuve.
Par calcul de la transformée de Walsh, on montre que le seul coefficient de Walsh $\widehat{\chi_f(u)} = 2^s \neq 0$ est celui pour $u = (1, 1, 1, \ldots, 1)$. D'où le résultat. ■

En considérant la fonction de non détection correspondante $1 \oplus X_1 \oplus X_2 \oplus \ldots X_s$, le plagiaire ne disposera pas de variables ou de groupe de variables plus favorables que d'autres pour produire des variantes non détectées. Il devra les considérer toutes simultanément. De plus, cette fonction de détection, selon la définition 3, ne dépend statistiquement que de l'ensemble de variables.

Un autre type de fonction, appartenant à la classe des fonctions fortement contournables, a été également considéré. Il s'agit des fonctions MAJORITÉ.

Définition 7 *On appelle fonction* MAJORITÉ *à n variables, notée* MAJ_n *la fonction booléenne de* $\mathbb{F}_2^n$ *dans* $\mathbb{F}_2$ *telle que*

$$f(x) = 1 \Leftrightarrow \begin{cases} wt(x) \geq \frac{n+1}{2} & si\ n\ impair \\ wt(x) \geq \frac{n}{2} + 1 & si\ n\ pair \end{cases}$$

De plus quand n est pair, exactement $\frac{1}{2}\binom{n}{\frac{n}{2}}$ *valeurs x de poids* $\frac{n}{2}$ *sont telles que* $f(x) = 1$.

Nous ne considérerons que les fonctions MAJ_{2p+1}. Les fonctions MAJ_n sont connues pour être équilibrées quelle que soit la valeur n de variables [ChaHayes98].

Les fonctions MAJ_n sont des fonctions fortement contournables, sauf asymptotiquement. En effet nous avons la proposition suivante.

Proposition 5 *Les fonctions booléennes* MAJ_n *sont immunes aux corrélations à l'ordre 0, pour toutes les variables* x_i *et*

$$P[\mathrm{MAJ}_n(x) = x_i] = \frac{1}{2} + \frac{\binom{n-1}{\frac{n-1}{2}}}{2^n}$$

Le lecteur trouvera la démonstration de cette proposition dans [Fil01]. Cette proposition montre que si les fonctions MAJ_n sont statistiquement dépendantes de chacune de leurs entrées, en revanche elles le sont

identiquement pour toutes ces variables. Donc, aucune ne joue un rôle plus important qu'une autre sur la sortie de la fonction.

Mais les fonctions MAJ_n ont, en tant que fonction de détection (ou de non détection), un intérêt particulier comme le prouve la proposition suivante.

Proposition 6 *Soit une fonction* MAJ_{2p+1}. *Sa formule DNF contient alors* $\binom{2p+1}{p+1}$ *termes. La fonction* $\overline{\mathrm{MAJ}_{2p+1}}$ *contient également* $\binom{2p+1}{p+1}$ *termes, chacun d'entre eux contenant* $p+1$ *variables de la forme* $\overline{x_i}$ *(négation).*

Le lecteur trouvera la démonstration dans [FJL06].

Ce résultat montre que si une fonction de détection est une fonction MAJ_{2p+1} alors le plagiaire devra modifier au minimum $p+1$ variables (octets) pour produire une variante non détectée. Il suffit de considérer des valeurs de p importantes pour compliquer sensiblement sa tâche.

Exemple 2 *Soit la fonction* MAJ_5 *donnée par sa DNF :*

$$\begin{aligned}\mathrm{MAJ}_5 \;=\;& x_4x_3x_2 \vee x_4x_3\overline{x_2}x_1 \vee \overline{x_4}x_3x_2x_1 \vee x_4\overline{x_3}x_2x_1 \vee x_4x_3\overline{x_2x_1} \vee \\ & x_4\overline{x_3}x_2\overline{x_1}x_0 \vee x_4\overline{x_3x_2}x_1x_0 \vee \overline{x_4}x_3x_2\overline{x_1}x_0 \vee \overline{x_4}x_3\overline{x_2}x_1x_0 \vee \\ & \overline{x_4x_3}x_2x_1x_0\end{aligned}$$

La DNF de la fonction de non détection correspondante est alors :

$$\begin{aligned}\overline{\mathrm{MAJ}_5} \;=\;& \overline{x_0x_1x_2} \vee \overline{x_0x_1}x_2\overline{x_3} \vee \overline{x_0x_1}x_2x_3\overline{x_4} \vee \\ & \overline{x_0}x_1\overline{x_2x_3} \vee \overline{x_0}x_1\overline{x_2}x_3\overline{x_4} \vee \overline{x_0}x_1x_2\overline{x_3x_4} \vee x_0\overline{x_1x_2x_3} \vee \\ & x_0\overline{x_1x_2}x_3\overline{x_4} \vee x_0\overline{x_1}x_2\overline{x_3x_4} \vee x_0x_1\overline{x_2x_3x_4}\end{aligned}$$

La fonction de non détection ne vaudra 1 que si au moins trois octets sont modifiés.

Le protocole d'analyse Durant la phase initiale d'installation, le logiciel antivirus collecte un certain nombre d'informations concernant à la fois l'utilisateur et le système :

- le numéro de série du processeur (CPUID) et celui du disque dur (HDID) ;

– l'adresse MAC si elle est disponible (notée `MACadr`) ;
– le nom utilisateur `USRname` et son adresse *e-mail* `@adr` ;
– une valeur secrète ν, dissimulée dans le logiciel antivirus [12].

Précisons, que d'autres paramètres peuvent être considérés. Cette phase est pleinement paramétrable. Les paramètres peuvent également être choisis aléatoirement dans une liste.

Ensuite, le logiciel calcule l'index i du sous-motif qui sera utilisé de manière fixe, comme suit :

$$i = g(H(\text{CPUID} \oplus \text{HDID} \oplus \texttt{MACadr} \oplus \texttt{USRname} \oplus \texttt{@adr} \oplus \nu) \oplus \mathcal{N}).$$

La fonction H est une fonction de hachage dont l'entrée est la somme bit à bit, modulo 2, des données collectées et codées sous forme d'entiers. La fonction g produit une valeur aléatoire comprise entre 1 et π, à partir du résultat de H et d'un entier d'initialisation N, éventuellement public. En conséquence, le même sous-motif de détection i sera utilisé en permanence lors de toute opération de scan par l'utilisateur sur cette machine. L'objectif est double. D'une part, l'auteur potentiel d'une variante la produira relativement à un unique sous-motif donné. D'autre part, en cas d'enquête, l'analyse permettra de prouver ou d'infirmer l'implication de l'utilisateur dans la production de cette variante. Il suffira à l'enquêteur de recalculer l'indice i à partir des informations contenues dans la machine du suspect pour déterminer si le contournement de détection a pu être réalisé relativement au sous-motif i.

2.4.2 Analyse mathématique

Analysons maintenant ce schéma de détection sécurisé. Notons $\{\mathcal{S}_{\mathcal{M}}, f_{\mathcal{M}}\}$ le schéma de détection complet et $\{\mathcal{S}^i_{\mathcal{M}}, f^i_{\mathcal{M}}\}$ le schéma de détection partiel relativement au sous-motif i.

Transinformation du schéma Avec les notations précédentes et par définition de notre schéma, nous avons de manière évidente :

$$I(\mathcal{S}_{\mathcal{M},\mathcal{M}}; \overline{f_{\mathcal{M}}}, \mathcal{S}_{\mathcal{M}}) = H(\mathcal{S}^i_{\mathcal{M},\mathcal{M}}) << H(\mathcal{S}_{\mathcal{M},\mathcal{M}}) - H(\mathcal{S}_{\mathcal{M},\mathcal{M}} | f^i_{\mathcal{M}}, \mathcal{S}^i_{\mathcal{M}}).$$

L'analyse en boîte noire ne permet de retrouver que $\{\mathcal{S}^i_{\mathcal{M}}, f^i_{\mathcal{M}}\}$.

12. Elle peut être obtenue par rétro-ingénierie logicielle mais en cas d'enquête, son utilisation pour produire une variante non détectée constituera une preuve de contrefaçon.

Rigidité du schéma Comme nous l'avons vu précédemment, elle dépend directement du poids de la fonction $\mathrm{wt}(f^i_{\mathcal{M}})$ et de k. Dans le cas de ce schéma, nous avons par conséquent :

$$\mathcal{R}(\mathcal{S}^i_{\mathcal{M}}, f^i_{\mathcal{M}}) = \frac{2^k - 1}{k2^k} = \frac{1}{k} - \frac{1}{k2^k} = \mathcal{O}(\frac{1}{k}).$$

Il est donc préférable d'avoir des valeurs de k relativement petites pour que le schéma ait la meilleure rigidité possible (voir section 2.2.2).

Impact d'une collusion Nous pouvons supposer qu'un groupe d'attaquants se constitue afin de mettre en commun leurs efforts pour extraire un schéma de détection donné. Déterminons la taille minimum que doit avoir cette collusion pour résoudre le problème de l'extraction.

Proposition 7 *Le schéma $\mathcal{SD} = \{\mathcal{S}_{\mathcal{M}}, f_{\mathcal{M}}\}$ ne peut être extrait par une collusion d'au moins $\tau = \lceil \frac{s}{k} \rceil$ membres.*

Preuve.
La preuve est évidente en considérant que chaque bloc contient k points. De là, nous avons $\tau \geq \lceil \frac{s}{k} \rceil$. Cela correspond au cas dans lequel les blocs (sous-motifs) utilisés par la collusion ont une intersection nulle. Dans ce cas, le design est une classe parallèle d'un design dit *résoluble*[13]. ■

Lorsque τ attaquants forment une collusion, ils extraient un motif de détection $\mathcal{S}_{\mathcal{M}}$ de taille s. Voyons à présent quelle forme possède la formule DNF décrivant la fonction de détection.

Proposition 8 *La formule DNF de la fonction de détection, extraite par une collusion d'au moins $\tau = \lceil \frac{s}{k} \rceil$ analystes contient au plus $\frac{s}{k}(2^{k-1})$ termes.*

13. Dans un RBIBD (*Resolvable Balanced Incomplete Bloc Design*), la famille $\mathcal{B}$ est une partition dont chaque partie est une *classe parallèle*. Une classe parallèle est un ensemble de blocs partitionnant l'ensemble $\mathcal{V}$. Deux conditions sont nécessaires pour qu'un BIBD soit résoluble : (1) $k|v$ et (2) $b \geq v + r - 1$. À titre d'exemple, considérons le $(9, 3, 1)$-`RBIBD`. L'ensemble $\mathcal{B}$ est alors

$$\begin{array}{cccc} \{1,2,3\} & \{1,4,7\} & \{1,5,9\} & \{1,6,8\} \\ \{4,5,6\} & \{2,5,8\} & \{2,6,7\} & \{2,4,9\} \\ \{7,8,9\} & \{3,6,9\} & \{3,4,8\} & \{3,5,7\}. \end{array}$$

Preuve.
La preuve est évidente en considérant que l'intersection de toute famille de blocs ne contient aucun bloc du design et que deux quelconques blocs peuvent être d'intersection non vide. La DNF contiendra exactement $\frac{s}{k}(2^{k-1})$ termes quand cette intersection est systématiquement vide (cas d'une classe parallèle). ■

Ce résultat implique que même si τ analystes venaientt à s'unir, la complexité de la procédure LogicalMinimize leur serait extrêmement défavorable et la fonction de non détection ne pourrait être extraite (pour des valeurs appropriées de s et de k).

Probabilité de détection résiduelle Supposons à présent qu'un plagiaire parvienne à extraire un schéma de détection partiel $\{\mathcal{S}^i_{\mathcal{M}}, f^i_{\mathcal{M}}\}$ relativement au sous-motif i et pour un détecteur $\mathcal{D}$ implémentant la schéma sécurisé proposé. Il est alors capable de produire une nouvelle variante, non détectée par $\mathcal{D}$. Quelle est alors la probabilité que cette variante reste détectée une fois dissiminée par le plagiaire (probabilité de détection résiduelle) ?

Proposition 9 *La connaissance du schéma de détection partiel* $\{\mathcal{S}^i_{\mathcal{M}}, f^i_{\mathcal{M}}\}$ *permet de générer une variante qui reste détectée avec une probabilité notée* $P_{détection}$ *telle que*

$$\frac{1}{2} \leq P_{détection} \leq 1.$$

Preuve.
Supposons que le plagiaire utilise $\mathcal{S}^i_{\mathcal{M}}$ et $f^i_{\mathcal{M}}$ pour produire une variante non détectée. Supposons que cette variante se propage sur une machine dont le détecteur utilise le sous-motif j. Les modifications opérées pour produire cette variante vont affecter la détection relativement au sous-motif j avec une probabilité (en pire cas) de $\frac{1}{2}$, sauf si les sous-motifs i et j sont d'intersection vide. Dans ce dernier cas, il est évident, par construction, que $P_{\text{détection}} = 1$. D'où le résultat. ■

Ce résultat montre que les objets combinatoires et la fonction de détection doivent être soigneusement choisis. En particulier, l'utilisation de `RBIBD` est préférable à celle de BIBD simples. Mais, même dans ce

dernier cas, les différentes implémentations et expériences réalisées ont montré que $P_{\text{détection}}$ reste plus proche de 1 que de $\frac{1}{2}$. Dans le cas le plus favorable (cas des `RBIBD`), nous avons $P_{\text{détection}} = \frac{\pi-1}{\pi}$.

2.4.3 Implémentations et performances

Les différentes implémentations et leur test ont montré, jusqu'à présent que les meilleurs objets combinatoires, pour les contraintes fixées, restent les classes parallèles de `RBIBD`. D'autres `BIBD` non résolubles ont également été utilisés avec succès mais dans ce dernier cas, le choix de la fonction de détection est plus délicat. Il faut en effet tenir compte du fait que les blocs peuvent être d'intersection disjointe.

Les ressources mémoire requises par ce schéma sont en $\mathcal{O}(\frac{s^2}{k})$ si l'on considère un $2-(s,k,\lambda)$ design générique (essentiellement due à la taille de la matrice d'incidence du design). Mais des considérations de phylogénie de codes [GGPS98, KWL05] devraient permettre de choisir des objets combinatoires bien meilleurs, pour gérer plusieurs variantes à la fois et ainsi améliorer la complexité mémoire. En termes de complexité de calcul et donc en temps de traitement, les expériences ont montré qu'il n'existait aucune différence significative entre les plus rapides des scanners et une implémentation relativement optimisée du schéma sécurisé que j'ai proposé.

Même si des progrès en termes d'implémentation peuvent encore être faits dans l'avenir, les résultats expérimentaux prouvent que ce schéma peut être utilisé en pratique et peut constituer une alternative, commercialement viable, aux techniques d'analyse de forme actuellement utilisées... avec le bénéfice non négligeable d'une meilleure lutte contre la prolifération virale.

2.5 Problèmes ouverts et axes de recherche futurs

La recherche, l'exploration et le classement des « bonnes » fonctions de détection est un champ de recherche essentiel. Il est également prometteur. Un autre axe de recherche non moins essentiel consiste à déterminer si d'autres propriétés sont à considérer pour ces fonctions,

afin d'augmenter à la fois l'efficacité des schémas de détection et la résistance à l'analyse en boîte noire et par conséquent la production de nouvelles variantes.

Tout cela constitue un ensemble de problèmes ouverts pour lesquels il est essentiel d'apporter des solutions. Parmi eux (la liste est non exhaustive), signalons :

- la classification l'exploration de bonnes fonctions de détection $f_{\mathcal{M}}$;
- est-il possible de trouver des propriétés structurelles pour ces fonctions, permettant d'améliorer l'efficacité des motifs de détection, tout en considérant des tailles de motifs s relativement restreintes, et en offrant une sécurité plus importante ;
- l'étude et l'exploration d'objets combinatoires plus intéressants, et offrant des propriétés encore meilleures que celles mises en évidence dans le schéma sécurisé proposé dans ce chapitre (*pairwise designs*, designs presque résolubles...) [BJL99, ColDin96] ;
- l'utilisation de schémas de partage de secret ou de schéma à seuil...

Enfin signalons que cette étude est pleinement transposable à l'autre grande famille de techniques de détection : l'analyse de comportements. Dans ce contexte, l'analyse est fonctionnelle et non plus formelle. La notion de motif de détection est alors remplacée par celle de famille de comportements identifiés comme malveillants. Les premiers résultats de travaux menés dans ce sens sont très encourageants [FJL06]. Ils ont également permis de prouver la faiblesse extrême des produits antiviraux dans le domaine de l'analyse comportementale. La notion de schéma de détection a été généralisé par celle de stratégie de détection, définie comme suit.

Définition 8 *(Stratégie de détection) Une* stratégie de détection $\mathcal{SD}$ *relativement à un code malveillant donné* $\mathcal{M}$ *est le triplet* $\mathcal{DS} = \{\mathcal{S}_{\mathcal{M}}, \mathcal{B}_{\mathcal{M}}, f_{\mathcal{M}}\}$*, où* $\mathcal{S}_{\mathcal{M}}$ *est un ensemble d'octets,* $\mathcal{B}_{\mathcal{M}}$ *un ensemble de fonctions de programme et* $f_{\mathcal{M}} : \mathbb{F}_2^{|\mathcal{S}_{\mathcal{M}}|} \times \mathbb{F}_2^{|\mathcal{B}_{\mathcal{M}}|} \rightarrow \mathbb{F}_2$ *une fonction booléenne.*

Il est intéressant de noter que cette définition concerne à la fois les codes malveillants connus et ceux éventuellement inconnus (mais néanmoins utilisant des techniques ou modes opératoires connus). En effet, lorsque un code malveillant inconnu $\mathcal{M}$ déclenche une alerte, c'est précisément l'ensemble $\mathcal{B}_{\mathcal{M}}$ qui est alors impliqué. C'est là tout l'intérêt

de la notion de stratégie de détection par rapport à celle de schéma de détection. Si la nature de l'ensemble $\mathcal{S}_{\mathcal{M}}$ est facile à appréhender – un ensemble d'octets –, celle de l'ensemble $\mathcal{B}_{\mathcal{M}}$ l'est probablement moins. En fait, cet ensemble peut être considéré comme un méta-ensemble d'octets de la manière suivante : les comportements peuvent être décrit par des structures d'octets qui correspondent à chaque procédure réalisant une action ou comportement donnés. Accéder à un fichier en lecture, créer un mutex sont des actions pouvant être décrite au moyen de telles structures plus ou moins complexes d'octets, localisées soit sur le disque dur (le code est inactif, l'analyse se fait par émulation de code) ou en mémoire (le code est actif). A titre d'exemple, pour surveiller le comportement consistant à tenter d'ouvrir en écriture le secteur de démarrage maitre ou secondaire, il est possible de dérouter l'interruption 13H, service 3 de la manière suivante :

```
INT_13H:
     cmp     CX, 1   ; est-ce le cylindre 0, secteur 1 ?
     jnz     DO_OLD  ; sinon on rend la main a l'appel 13H
                     ; original
     cmp     DH, 0   ; est-ce la tete 0 ?
     jnz     DO_OLD  ; sinon on rend la main a l'appel 13H
                     ; original
     cmp     AH, 3   ; est-ce un service em ecriture ?
     jnz     DO_OLD  ; sinon on rend la main a l'appel 13H
                     ; original
     .....

DO_OLD:
     jmp     dword ptr CS:[OLD_13H] ;
```

Cette tentative d'écriture est identifiée par un ensemble d'octets décrivant en détail la nature du service et les paramètres afférents. D'un point de vue formel donc, nous avons $\mathcal{B}_{\mathcal{M}} \subset \mathbb{N}_{256}^{\infty}$ (une famille de séquences d'octets de longueur indéfinie). Dans ce qui suit, nous parlerons simplement de comportement. Dire que le comportement $b \in \mathcal{B}_{\mathcal{M}}$ est réalisé signifie que le code contient ou réalise une structure d'octets lors de son exécution.

Les premiers résultats montrent que les fonctions de détection dépendent uniquement de variables booléennes décrivant des octets du

$\mathcal{H}_1 : T_{sig}$	
$\mathcal{H}_2 : T_{sig} \vee (T_{sig} \wedge T_{behav}) = T_{sig}$	(par la loi d'absorption)
$\mathcal{H}_3 : T_{behav} = \bigvee_{i=0}^{b} X_i$	où X_i décrit le $i$ème élément dans la base comportementale $\mathcal{B}_\mathcal{M}$

TABLE 2.3 – Fonctions booléennes modélisant la détection comportementale

code (analyse de forme) uniquement ou bien sont très fortement corrélées à ces octets et que les comportements sont peu ou prou considérés. Les premiers résultats obtenus [FJL06] montrent clairement que la détection comportementale n'est pas réellement utilisée, excepté pour l'antivirus *Viguard*. Cela peut être formulée par les deux hypothèses suivantes, concernant la fonction de détection :

- $\mathcal{H}_1$: la détection comportementale n'est pas implémentée ou elle est inefficace,
- $\mathcal{H}_2$: la détection comportementale est ignorée sans une validation par la détection par analyse de forme (signature par exemple).

Pour le produit *Viguard*, une hypothèse particulière peut être formulée.

- $\mathcal{H}_3$: la détection comportementale consiste à prendre tout comportement potentiellement menaçant.

D'un point de vue mathématique, l'utilisation des fonctions booléennes permet de définir les hypothèse précédentes plus rigoureusement. Ces fonctions sont données dans le tableau 2.3 La notation T_{sig} désigne la restriction fonctionnelle $f_\mathcal{M}^{\mathcal{S}_\mathcal{M}}$ de la fonction générale de détection $f_\mathcal{M}$ à l'ensemble $\mathcal{S}_\mathcal{M}$. Cela signifie que les variables booléennes relatives à l'ensemble $\mathcal{B}_\mathcal{M}$ n'apparaissent pas dans forme normale disjonctive de $f_\mathcal{M}^{\mathcal{S}_\mathcal{M}}$. Inversement, s'agissant de l'hypothèse $\mathcal{H}_3$, nous considérons la restriction fonctionnelle $f_\mathcal{M}^{\mathcal{B}_\mathcal{M}}$ à l'ensemble des comportements $\mathcal{B}_\mathcal{M}$, avec les différences importantes que :

- l'ensemble $\mathcal{B}_\mathcal{M}$ contient tous les comportements possibles qui peuvent potentiellement être utilisés par un code malveillant (incluant ceux qui sont également utilisés par les programmes non malveillants),
- la fonction de détection est la fonction logique OU, dont le poids

est $2^{|\mathcal{B}_{\mathcal{M}}|} - 1$.

L'utilisation de fonctions booléennes pour modéliser et étudier les techniques de détection fondées sur les stratégies de détection, aussi puissante soit elle, est très vite limitée par la complexité des fonctions de détection, lorsque ces dernières ne sont plus triviales. L'approche globale est alors impossible et il est nécessaire d'avoir une vision plus locale, et notamment combinatoire.

La réalisation de certaines structures est alors à considérer. Cela est déjà le cas avec l'analyse sémantique [PCJD07] ou les techniques de preuve de modèle (*model checking*) [Kinder05] dans lesquelles des structures d'arbres ou de graphes sont déjà utilisées. Mais, d'autres structures combinatoires, moins intuitives car moins intimement liées aux aspects formels et/ou fonctionnels du code, peuvent se réveler tout aussi puissantes, voire peut être plus. Une partie de mes recherches futures sera consacrée à l'exploration de telles structures.

Chapitre 3

Fonctions booléennes, diffusion et confusion

3.1 Introduction

Les propriétés d'aléa sont essentielles en cryptographie. Pour l'attaquant, toute quantité produite par un système cryptologique doit ressembler le plus possible à de l'aléa. Cela implique notamment, mais pas uniquement, que le système ne doit pas être prévisible ou prédictible. Ces quantités doivent par conséquent avoir une taille suffisante et être « aléatoires » dans le sens que la probabilité qu'un ensemble particulier de valeurs, pour ces quantités soit réalisé, doit être la plus faible possible, et ce pour tout ensemble de valeurs possibles – ce qui oblige à considérer la loi uniforme comme loi de référence. Si cela n'était pas le cas, le cryptanalyste pourrait tirer un avantage certain, par l'intermiédiaire d'une stratégie recherche optimisée fondée sur d'éventuels biais concernant ces quantités.

D'un point de vue général, tout système de chiffrement symétrique et toute fonction de hachage doit être conçu de sorte à réaliser un générateur aléatoire pour chacun de ses bits de sortie[1]. Nous noterons GPAB un tel générateur (pour Générateur Pseudo-Aléatoire de Bits).

Deux propriétés essentielles doivent être satisfaites : les séquences

1. Cela est également vrai pour les systèmes asymétriques. C'est d'ailleurs systématiquement réalisé par l'utilisation d'objets et d'opérations naturellement sans biais.

produites par un GPAB doivent être statistiquement indistingables d'une « vraie » séquence aléatoire et ces bits doivent être non prédictibles pour un attaquant disposant d'une puissance de calcul bornée. Afin de tester si ces deux propriétés sont effectivement réalisées, de nombreux tests statistiques ont été proposés et sont utilisés. Historiquement, il faut citer les postulats de Golomb [Gol82]. Ces tests sont nécessaires mais pas suffisants et ils ne permettent que d'évaluer si un registre à décalage produit des séquences statistiquement acceptables. Satisfaisantes au regard de ces postulats, ces séquences se révèlent extrêmement prédictibles lorsque l'on applique l'algorithme de Massey-Berlekamp [Massey69]. Cela illustre le fait que la notion d'aléa n'est pas une notion absolue mais relative uniquement à une série de tests statistiques [Fil05b].

De nombreux autres tests statistiques ont été ensuite proposés pour améliorer la vision de ce que l'on qualifie d'« aléatoire ». Citons les tests de références : test de fréquences, test des séries, test poker, test d'autocorrelation...[FIPS140, Knuth81], test universel de Maurer [Maurer92]...

Toutefois ces tests sont essentiellement conçus pour tester les suites produites par des sysèmes de chiffrement par flot (ou les systèmes par bloc en mode flot). Leur application à des systèmes par bloc ou des fonctions de hachage, leur intérêt est loin d'être évident. Dans ce cas, les concepts de *diffusion* et de *confusion* [Shan48] sont généralement préférés. Mais ces concepts sont soit trop empiriquement définis soit de manière trop théorique pour être réellement utilisable.

Tous les systèmes de chiffrement symétriques actuels et les fonctions de hachage actuels satisfont les propriétés aléatoires généralement requises d'un point de vue cryptographique. Aucun biais statistique significatif n'a été à ce jour mis en évidence.

Or le travail du cryptanalyste consiste à tout d'abord identifier de tels biais, dus soit à des erreurs inconnues de conception soit des trappes, et dans les deux cas non révélés par les tests statistiques classiques. Dans ce but, il imagine de nouveaux tests permettant de révéler des biais exploitables. Rappelons que la notion d'aléa est un concept « philosophique ». En pratique, il n'est défini que par rapport un ensemble de tests [Fil05b].

Dans ce chapitre, je vais présenter un nouveau test statistique fondé

sur la distribution du χ^2 et que j'ai appelé test statistique de Möbius. Plus précisément, je définis comme statistique de travail X le nombre de monômes de degré exactement d dans la Forme Algébrique Normale (FAN) des fonctions booléennes modélisant chacun des bits de sortie d'un système cryptographique. L'ensemble de ces d-monômes effectivement représentés dans chaque FAN, sont calculés à l'aide de la transformée de Möbius. Un système cryptographique théoriquement sûr exhibe une distribution fixe de ces d-monômes, déterminée à partir de résultats généraux concernant les fonctions booléennes aléatoires. Cela permet de déterminer par des tests statistiques non paramétriques sur les fonctions booléennes modélisant un système donné sont véritablement aléatoires.

Tous ces tests ont été implémentés pour un certain nombre de systèmes de chiffrement par flot et par bloc, ainsi que pour la plupart des fonctions de hachage en usage. Tous ces systèmes sont connus pour avoir passé avec succès les tests statistiques classiques et sont par conséquent considérés comme ayant de bonnes propriétés cryptographiques. Les résultats montrent que plusieurs de ces systèmes échouent face aux tests statistiques de Möbius. Pour les autres systèmes testés, leur qualité reconnue, n'a pas été informée.

3.2 Caractérisation des fonctions booléennes et résultats

Nous allons voir comment décrire d'une nouvelle manière, une fonction booléenne en utilisant essentiellement sa FAN. Cette forme algébrique normale se calcule au moyen de la transformée de Möbius. Nous en déduirons quelques résultats concernant des propriétés cryptographiques telles que l'équilibre et la corrélation, en considérant la transformée de Walsh.

3.2.1 Structure d'une forme algébrique normale

Une fonction booléenne est une fonction f définie dans $\mathbb{F}_2^n$ et à valeur dans $\mathbb{F}_2$. Il existe 2^{2^n} telles fonctions. Nous définirons une fonction booléenne aléatoire comme une fonction f dont les valeurs sont des va-

riables indépendantes (des valeurs d'entrée), identiquement distribuées (variables *i.i.d.*). Autrement dit :

$$\forall (x_1, \ldots, x_n) \in \mathbb{F}_2^n, \quad P[f(x_1, \ldots, x_n) = 0] = \frac{1}{2}. \tag{3.1}$$

Il est équivalent de dire que chaque $f(x_1, \ldots, x_n)$ est une variable de *Bernoulli* de paramètre $\frac{1}{2}$. Nous noterons cette loi $\mathcal{B}(p)$ avec $p = \frac{1}{2}$ dans notre cas [2]

Le poids d'une fonction booléenne définie sur $\mathbb{F}_2^n$ est donné par $wt(f) = |\{x \in \mathbb{F}_2^n | f(x) = 1\}|$. Une telle fonction sera dite *équilibrée* si $wt(f) = 2^{n-1}$.

La forme algébrique normale est donnée de f est le polynome multivarié donné par $f(x_1, \ldots, x_n) = \bigoplus_{u \in \mathbb{F}_2^n} a_u x^u, \quad a_u \in \mathbb{F}_2$, où $u = (u_1, \ldots, u_n)$ et $x^u = \prod_{i=1}^n x_i^{u_i}$. Les a_u sont donnés par la transformée de Möbius [McCarthy86] de f :

$$a_u = \bigoplus_{x \preceq u} f(x) \tag{3.2}$$

où $\preceq$ décrit l'ordre partiel sur le treillis booléen, c'est-à-dire que $\alpha \preceq \beta$ si et seulement si $\alpha_i \leq \beta_i$ pour tout $1 \leq i \leq n$. Un monôme $a_u x^u$ de la FAN sera de degré k si $a_u = 1$ et si $wt(u) = k$ où $wt(.)$ représente la poids de Hamming. Avec ces notations, nous pouvons à présent donner un premier résultat.

Proposition 10 *La forme algébrique normale (FAN) d'une fonction booléenne aléatoire $f : \mathbb{F}_2^n \to \mathbb{F}_2$ possède 2^{n-1} monômes en moyenne. Pour tout entier k tel que $0 \leq k \leq n$, elle contient en moyenne $\frac{1}{2}\binom{n}{k}$ monômes de degré k.*

Lorsque $k = 0$ (respectivement $k = n$), il est équivalent de dire qu'une fonction booléenne aléatoire sur deux, en moyenne, contient le terme a_0 (resp. $a_{(111\cdots11)}$) dans sa FAN.

Preuve.
Un monôme donné $x_{i_1} x_{i_2} \ldots x_{i_k}$ de degré k sera contenu dans la FAN

2. Tout n-uplet $(x_1, \ldots, x_n)$ est choisi de manière aléatoire et indépendante, ainsi que $f(x_1, \ldots, x_n)$. Il est alors équivalent de choisir f au hasard dans l'ensemble des fonctions booléennes.

si et seulement si $a_u = 1$ dont le support (c'est-à-dire l'ensemble des indices j tels que $u_j = 1$, noté $supp(u)$) est $\{i_1, i_2, \ldots, i_k\}$. Or nous avons

$$\begin{aligned} a_u &= f(\overline{0})) \oplus \bigoplus_{j=1}^{k} f(e_{i_j}) \oplus \left(\bigoplus_{l=1}^{k} \bigoplus_{j=1, j\neq l}^{k} f(e_{i_j} \oplus e_{i_l}) \right) \oplus \ldots \\ &\quad \ldots \oplus f(\bigoplus_{j=1}^{k} e_{i_j}), \end{aligned} \tag{3.3}$$

où $\overline{0} = (0, 0, \ldots, 0)$ et e_i représente le n-uplet dont seule la $i$ème coordonnée est non nulle. Le membre droit de l'équation (3.3) possède $\sum_{j=1}^{k} \binom{k}{j} = 2^k$ termes. Nous avons donc $a_u = 1$ si un nombre impair de termes sont tous égaux à 1. Il y a 2^{k-1} configuations. Chacune d'elles, selon (3.1), a une probabilité $\frac{1}{2^k}$ de valoir 1 comme nous considérons des variables *i.i.d.*. D'où, nous avons $P[a_u = 1] = 2^{k-1} \times \frac{1}{2^k} = \frac{1}{2}$. Ainsi, le nombre de monômes de degré k dans l'ANF sera donné par $P[a_u = 1] \times \binom{n}{k} = \frac{1}{2} \times \binom{n}{k}$. ■

Ce résultat peut être généralisé avec le théorème suivant.

Théorème 2 *Avec les notations de la Proposition 10, le nombre n_k de monômes de degré k suit une loi normale de moyenne $E[n_k] = \frac{1}{2}\binom{n}{k}$ et de variance $V[n_k] = \frac{1}{4}\binom{n}{k}$.*

Pour être mathématiquement rigoureux, nous devrions considérer la distribution binomiale plutôt que la distribution normale et nous devrions écrire que n_k converge en loi vers la distribution normale. Toutefois, la théorie des probabilités [Fell66, Lejeune05] autorise de tels raccourcis de langage dès que les résultats centraux limites sont vérifiés, ce qui est le cas dans notre étude.

Preuve.
La preuve est directe en considérant que a_u, pour tout $u \in \mathbb{F}_2^n$ est une variable aléatoire de Bernouilli, de paramètre $\frac{1}{2}$, avec $E[a_u] = \frac{1}{2}$ et $V[a_u] = \frac{1}{4}$. Comme $n_k = \sum_{wt(u)=k} a_u$, pour des valeurs suffisamment grandes du nombre de u de poids k, le théorème central limite fournit le résultat (dès que $n_k \geq 30$ [Fell66, Lejeune05]). ■

Ce résultat permet l'étude des propriétés aléatoires des fonctions booléennes. Considérons une fonction f utilisées pour le rebouclage d'un

registre à décalage de longueur L. Si f est la fonction constante (sa FAN est réduite à un seul monôme), sa sortie l'est également et n'est par conséquent pas aléatoire. Dans le cas d'un rebouclage linéaire (la FAN de f est de degré 1 et possède au plus L monômes), les propriétés aléatoires ont une portée limitée : les propriétés de linéarité ne sont pas supprimées et il est possible d'exhiber des estimateurs ayant une valeur donnée avec une probabilité 1. De plus, des informations de nature combinatoire sont également disponibles [Gol82].

En d'autres termes, si nous considérons $x = (x_1, \ldots, x_n)$ et $y = (y_1, \ldots, y_n)$ tels que (par exemple) $f(x) = f(y) = 1$ moins la fonction f est aléatoire, plus il est facile d'extraire de l'information sur x et sur y, ou dans certains cas de x à partir de y (ou l'inverse).

Exemple 3 *Considérons la fonction $f(x_1, x_2) = x_1 \oplus x_2$. Tout $x = (x_1, x_2)$ et tout $y = (y_1, y_2)$ tels que $x \neq y$ et $f(x) = f(y) = 1$, satisferont la relation $x_1 \oplus y_1 = 1$. Cela est dû au fait que les valeurs de la table de vérité contiennent sont réparties de manière « structurée » et non aléatoirement réparties. Notons que cette fonction est équilibrée, ce qui fait de l'équilibre une propriété nécessaire mais nullement suffisante.*

La proposition 10 nous permet d'établir le critère suivant :

Corollaire 1 *Une fonction booléenne aléatoire présentant le meilleur compromis en terme de propriétés cryptographiques doit avoir le plus haut degré possible.*

Preuve.
Ce résultat provient du fait qu'une fonction booléenne aléatoire à n variables possède un monôme de degré n dans sa FAN en moyenne avec une probabilité $\frac{1}{2}$ et contiendra $\frac{n}{2}$ termes de degré $n-1$. Selon la borne supérieure du degré d'une fonction booléenne présentant le meilleur compromis en termes de « bonnes » propriétés cryptographiques (équilibre, immunité aux corrélations, non-linéarité...) [Sieg84], pour une fonction immune aux corrélations à l'ordre t nous avons $\deg(f(x_1, \ldots, x_n)) \leq n - t - 1$. Il s'ensuit qu'imposer de telles propriétés à une fonction diminue son degré algébrique. ■

Dans le même temps, il s'ensuit que des structures de nature combinatoires sont introduites et le comportement aléatoire général de ces

fonctions est diminué. La recherche du meilleur compromis cryptographique, pour préserver simultanément le meilleur caractère aléatoire possible (supprimer les structures dans la répartition des variables d'entrée en fonction de leur sortie respective), la fonction doit avoir le plus grand degré possible, dans la limite imposée par la borne de Siegenthaler [Sieg84].

L'étude de ces propriétés combinatoires est l'un de mes axes principaux de recherche. L'idée est de partionner l'ensemble des valeurs d'entrée d'une fonction donnée, en fonction d'un (ou plusieurs) critère combinatoire donné, lequel doit être traduisible en un (ou plusieurs) estimateur présentant un biais par rapport aux lois gouvernant une telle partition dans le cas de fonctions aléatoires. Si cette étude est relativement facile, pour un faible nombre de variables, par une approche exploratoire exhaustive, cette dernière est en revanche impossible les fonctions booléennes utilisées en pratique. C'est la raison pour laquelle je développe les logiciels *CoHS* et VAUBAN dont le but est de rechercher de telles structures combinatoires. Cette approche qui fera l'objet du chapitre 5, a permis ainsi d'identifier de sérieuses faiblesses dans le système de chiffrement E0 du protocole Bluetooth [Fil06d].

3.2.2 Caractérisation des coefficients de Walsh

La transformée de *Walsh Hadamard* d'une fonction booléenne f est donnée par la transformation suivante :

$$\forall u \in \mathbb{F}_2^n, \quad \widehat{\chi_f}(u) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+<x,u>},$$

où $< x, u >$ décrit le produit scalaire usuel sur $\mathbb{F}_2^n$. Le théorème de Xiao-Massey [XioMass88] permet de caractériser l'immunité aux corrélations de f à l'aide de cette transformée :

Proposition 11 *[XioMass88] Une fonction booléenne f est immune aux corrélations à l'ordre t si et seulement si $\forall u \in \mathbb{F}_2^n, \quad 1 \leq wt(u) \leq t \qquad \widehat{\chi_f}(u) = 0$.*

De plus f est équilibrée si et seulement si $\widehat{\chi_f}(0, 0, \ldots, 0) = 0$.

Proposition 12 *Soit une fonction booléenne aléatoire f définie sur $\mathbb{F}_2^n$ avec $n \geq 5$. Pour tout $u \in \mathbb{F}_2^n$, $\widehat{\chi_f}(u)$ est une variable aléatoire distribuée selon une loi normale de moyenne 0 et de variance 2^n.*

Preuve.
Nous pouvons écrire tout d'abord que

$$\widehat{\chi_f}(u) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+<x,u>} = (2^n - 2) \cdot \sum_{x \in \mathbb{F}_2^n} (f(x)+ < x, u >).$$

Comme x et $f(x)$ sont indépendant (par définition), nous pouvons considérer les valeurs $< x, u > +f(x)$ comme des variables aléatoires, indépendantes, identiquement distribuées, et ce pour tout x. Notons $Y = \sum_{x \in \mathbb{F}_2^n} (f(x)+ < x, u >)$. Pour $n > 5$ (en d'autres termes $2^n > 32$), les résultats centraux limites [Fell66, Lejeune05], font que Y a une distribution de Gauss $\mathcal{LG}(E, \sigma^2)$ avec

$$\begin{aligned} E[Y] &= 2^n P[f(x)+ < x, u >= 1] = 2^{n-1} \\ (\sigma_Y)^2 &= 2^n P[f(x)+ < x, u >= 1] P[f(x)+ < x, u > \neq 1] = 2^{n-2}. \end{aligned}$$

D'où, $\widehat{\chi_f}(u)$ a également une distribution de Gauss de moyenne

$$E[\widehat{\chi_f}(u)] = 2^n(1 - 2P[f(x)+ < x, u >= 1]) = 0,$$

et de variance

$$\sigma^2 = 4.2^n P[f(x)+ < x, u >= 1] P[f(x)+ < x, u > \neq 1] = 2^n.$$

■

Si Φ représente la fonction de densité de la loi normale,

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{t^2}{2}\right) dt,$$

et si $p_0 = \Phi(\frac{1}{2^{\frac{n}{2}-1}}) - \frac{1}{2}$, alors nous pouvons écrire que

Lemme 1

$$P[f \ \textit{équilibrée}\ ] = p_0.$$

Preuve.
Pour une fonction booléenne équilibrée, nous avons $\widehat{\chi_f}(0, \ldots, 0) = 0$. Par définition, $\widehat{\chi_f}(u)$, $\forall u \in \mathbb{F}_2^n$ est une valeur paire. Alors, nous avons : $P[\widehat{\chi_f}(u) = 0] = P[0 < \widehat{\chi_f}(u) < 2]$. Le reste s'établit facilement avec la proposition 12. ■

Remarque.- Ce résultat fournit une approximation très précise de la probabilité « exacte » qu'une fonction soit équilibrée. Cette probabilité

exacte est donnée par $p = \frac{\binom{2^n}{2^{n-1}}}{2^{2^n}}$. Le tableau 3.1 compare la probabilités exacte avec celle approximée à l'aide du théorème 1 pour $5 \leq n \leq 19$. Cette approximation est d'autant plus intéressante que la valeur de p est calculatoirement plus difficile à obtenir que celle de p_0, laquelle requiert un temps négligeable.

n	p	p_0	n	p	p_0
5	0.1399	0.1381	13	0.008815	0.008814
6	0.09935	0.09870	14	0.006233	0.006233
7	0.07039	0.07015	15	0.004408	0.004407
8	0.04982	0.49738	16	0.003117	0.003116
9	0.03524	0.03521	17	0.002204	0.002203
10	0.02493	0.02491	18	0.001558	0.001558
11	0.01763	0.01762	19	0.001102	0.001101
12	0.01247	0.01246			

TABLE 3.1 – Probabilités exacte et approchée pour une fonction d'être équilibrée

3.3 Un nouveau test statistique

Je vais maintenant présenter différents tests permettant d'évaluer de nouvelles propriétés statistiques de systèmes de chiffrement symétriques et de fonctions de hachage. Considérons un système cryptologique et spécifions le référentiel d'étude choisi. Soit une clef secrète $K = (k_0, \ldots, k_{n-1})$. Un système de chiffrement par flot peut être modélisé de la manière suivante : chaque bit de sortie i généré à partir de la clef K peut être représenté par une fonction booléenne unique que nous représenterons sous sa forme algébrique normale. Cette forme sera calculée à l'aide de la transformée de Möbius définie par l'équation (3.2).

En d'autres termes, la suite pseudo-aléatoire de N bits produite par un système par flot peut être modélisée par une famille de N fonctions booléennes $(f_t(K))_{0 \leq t < N} = (f_0(K), \ldots, f_{N-1}(K))$ où $f_i(K)$ décrit le i-ème bit produit par le système et modélisé par un polynôme multivariés dont les variables sont les k_i. Ce polynôme est précisément la FAN.

Donc, chaque bit chiffrant la valeur prise par une fonction booléenne $f_t : \mathbb{F}_2^n \mapsto \mathbb{F}_2$.

Selon le même principe, représentons un sysème de chiffrement par bloc utilisant une clef K de n bits et travaillant sur des blocs de clair de m bits. Ainsi comme pour les systèmes par flot, mais avec des fonctions booléennes de sortie évaluées sur l'espace clef et l'espace du clair $P = (p_0, \ldots, p_{m-1})$, pour un bloc de cryptogramme C, nous avons $C = (c_0, \ldots, c_{m-1}) = (f_0(K, P), \ldots, f_{m-1}(K, P))$. Chacun des m bits chiffrés est ainsi la valeur prise par une fonction booléenne $f_t : \mathbb{F}_2^{n+m} \mapsto \mathbb{F}_2$.

Enfin pour une fonction de hachage $H : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$, l'empreinte numérique de m bits (haché) d'un bloc $B = (b_0, \ldots, b_{n-1})$ sera décrite par $(h_t(B))_{0 \leq t < m} = (h_0(B), \ldots, h_{m-1}(B))$.

Dans le reste du chapitre, j'utiliserai indifféremment le terme « bit de sortie » et « fonction booléenne de sortie », pour décrire les quantités produites par un système cryptologique. Avec les notations et remarques précédentes, les deux termes sont équivalents. Enfin, nous considèrerons, en première approche, que les différentes fonctions booléennes de sortie sont statistiquement indépendantes. C'est précisément la vision issue des résultats des tests statistiques usuellement utilisés pour évaluer un système cryptologique [FIPS140].

Il est évisent que toute FAN de fonction booléenne de sortie ne peut, en pratique, être calculée. En effet, elle contient en moyenne 2^{n-1} monômes. Un tel calcul requiert des ressources en temps et en mémoire, exponentielles. Pour ces nouveaux tests, et ce sans perte de généralités, je me suis limité à considérer les monômes de degré au plus 3, c'est-à-dire des FAN tronquées au degré 3. Je nommerai ces fonctions tronquées, des 3-FAN (plus généralement des d-FAN). Il s'agit de FAN partielles dont tous les termes sont de degré 3 au plus. Dans certains cas, des 5-FAN ont été calculées. D'un point de vue pratique, la formule (3.3) a été utilisée pour calculer ces termes. Le résultat observé, pour chaque FAN, est un nombre $\hat{n}_d$ de monômes de degré exactement d.

Je noterai H_0^d l'hypothèse statistique selon laquelle le nombre $\hat{n}_d$ de monômes de degré exactement d est distribué comme précisé dans le théorème 2. En d'autres termes, le système cryptographique ayant produit ces fonctions booléennes de sortie passera avec succès le test – et ainsi n'aura pas présenté de biais statistiques significatifs, ni de

particularités structurelles – quand il satisfera H_0^d.

3.3.1 Le test de la constante affine

L'hypothèse est notée H_0^0. Selon le théorème 2, la probabilité que la constante affine a_0 soit présente dans chaque fonction booléenne de sortie, est donnée par $p = \frac{1}{2}$. De manière équivalente, cela signifie que le nombre de fonctions booléennes de sortie dont la FAN contient cette constante suit une loi normale $\mathcal{N}(\frac{N}{2}, \frac{\sqrt{N}}{2})$ où N est le nombre total de fonctions booléennes de sortie pour le système étudié.

Si X_S, le nombre de fois où $a_0 = 1$, représente la statistique que nous considérons sur l'echantillon S de sortie composé de N FAN, nous pouvons à présent décrire le test bilatéral suivant, dénommé, *test de la constante affine* :

1. Calculer X_S sur S.
2. Fixer un niveau de signification α et choisir un seuil de décision x_α de telle sorte que pour une statistique X distribuée selon la loi normale standard, on ait $P[X > x_\alpha] = P[X < x_\alpha] = \frac{\alpha}{2}$.
3. Si la valeur $\hat{X}_S = \frac{X_S - \frac{N}{2}}{\frac{\sqrt{N}}{2}} > x_\alpha$ ou si $\hat{X}_S < -x_\alpha$ alors H_0^0 est rejetée (le système échoue au test et présente des faiblesses) autrement H_0^0 est conservée (le système passe le test).

3.3.2 Le test des d-monômes

Nous considérons à présent des monômes de degré exactement d dans les FAN des fonctions booléennes de sortie. L'hypothèse attachée sera notée H_0^d.

Avec les notations du théorème 2, le nombre de monômes de degré d dans la FAN d'une fonction booléenne aléatoire est une variable alátoire distribuée selon une loi normale $\mathcal{N}\left(\frac{1}{2}\binom{n}{d}, \frac{1}{2}\sqrt{\binom{n}{d}}\right)$. Nous allons utiliser des tests d'adéquation entre les fréquences théoriques (notées n_d) et celles observées pour le système cryptographique étudié (notées $\hat{n}_d$).

Le premier test, T_1^d considère séparément chaque FAN et de fait privilégie une vision plutôt locale donnant plus de poids au FAN présentant un biais. Le second test, T_2^d, groupe les N FAN de sortie en plusieurs classes. Le test en lui-même considère la distribution du χ^2 à

ν degrés de liberté en considérant les ν carrés des variables aléatoires indépendantes $\frac{(n_d^i - \hat{n}_d^i)}{\sqrt{n_d^i}}$ $(i \leq \nu)$ qui par définition sont normalement distribuées.

Avec le test T_1^d, $\nu = N-1$ tandis que pour le test T_2^d nous choisissons $2 \leq \nu \leq 9$.

1. Calculer pour chacune des ν variables aléatoires, n_d^i et $\hat{n}^i$ (n_d^i est donné par le théorème 2).
2. Fixer un niveau de signification α et choisir un seuil de décision x_α (calculé directement à partir de la fonction de répartition de la loi du χ^2) de telle sorte que pour une statistique X calculée sur un échantillon aléatoire nous avons $P[X > x_\alpha] = \alpha$ (quand X suit une loi du χ^2 à ν degrés de liberté).
3. Calculer la statistique $D^2 == \sum_{i=1}^{\nu} \frac{(n_d^i - \hat{n}_d^i)^2}{n_d^i}$.
4. Si $D^2 > x_\alpha$ alors on doit rejeter l'hypothèse H_0^d (le système cryptologique échoue au test car présente un biais statistique dans la distibution des monômes de degré d) sinon nous la conservons (le système ne présente aucun biais significatif).

Le second test, noté T_2^d, décrit quant à lui, le système d'un point de vue global. En particulier, son objectif est de vérifier si des biais locaux, mis en évidence par le test T_1^d, ont un impact significatif sur le comportement statistique général du système, à un niveau global. Pour ce test, au lieu de considérer les fréquences observées $\hat{n}_d^i$ des d monômes pour chacune des N fonctions booléennes de sortie, le test T_2^d considère le nombre de fonctions booléennes de sortie dont le nombre $\hat{n}_d$ appartient à un intervalle prédéfini $[a, b[$. Les fréquences théoriques pour chaque classe sont calculées à l'aide du théorème 2 par un simple calcul de probabilités.

3.3.3 Tests des d-monômes sur un sous-ensemble fixé de sorties

Pour ces tests, nous considérons les tests précédents présentés dans les section 3.3.1 et 3.3.2 mais en se restreignant à des sous-ensembles particuliers S de fonctions booléennes de sortie. L'objectif est de détecter des sous-ensembles « faibles » de fonctions de sortie. Bien évidemment, le nombre de ces sous-ensembles (au total 2^N) interdit une

étude exploratoire exhaustive aussi l'étude a-t-elle été limitée jusqu'à présent à de faibles valeur de $|S|$. Ces test sont notés $T_i^d|S$. Les critères de choix de sous-ensembles de plus grande taille est une de mes voies de recherche actuelle. Je détaillerai ce point dans la section 3.5.

3.4 Résultats de simulation

Ces tests ont été appliqués sur un certain nombre de systèmes tandis que d'autres auteurs [SAARINEN06] les ont repris sur des systèmes plus récents. Pour les quelques résultats présentés ici, nous avons considéré $\alpha = 0.05, 0.01$ et 0.001. Les résultats qui suivent sont organisés selon les trois grandes catégories de systèmes cryptographiques. Ils montrent pour au moins une d'entre elles, l'efficacité de cette modélisation à base de fonctions booléennes.

3.4.1 Chiffrement par flot

Nous considérerons essentiellement des systèmes présentés pour le projet européen NESSIE [Nesssie]. D'autres systèmes récents, proposés récemment dans le cadre d'*Ecrypt* [ECRYPT], le projet issu de NESSIE, ont montré des faiblesses similaires [SAARINEN06]. Pour ces expériences, le test a porté sur les $N = 6016$ premiers bits de la suite chiffrante. Il est intéressant de noter que :

	T_1^1	T_1^2	T_2^1	T_2^2
Lili-128	**Echec**	**Echec**	**Echec**	**Echec**
RC4 [Schneier96]	Non rejet	Non rejet	Non rejet	Non rejet
Snow	Non rejet	Non rejet	**Echec**	**Echec**
Bgml [Nesssie]	Non rejet	Non rejet	Non rejet	Non rejet

TABLE 3.2 – Systèmes par flot : résultats de tests (niveau de signification $\alpha = 0.001$)

– à l'exception du système Lili-128, tous les autres stystèmes testés satisfont le test de la constante affine ;

- Lili-128 présente des biais statistiques extrêmement importants. Le tableau 3.3 résume les résultats des tests pour ce système de chiffrement ;
- le système Snow présente également des biais extrêmement forts mais seulement en considérant le comportement statistique global. Bien que les différentes primitives cryptographiques soient séparément fortes, la manière de les combiner provoque des dégénérescences et affecte la qualité statistique résultante du système ;
- il est également intéressant d'étudier la convergence des tests mis en œuvre (c'est-à-dire la distance existant entre la valeur réalisée de l'estimateur et le seuil de décision [HT88]). Les systèmes de chiffrement du tableau 3.2 peuvent être ainsi classés selon leur qualité statistique « relative ». Nous observons alors que ($\succeq$ signifie « meilleur que ») :

$$\text{Bgml} \succeq RC4 \succeq \text{Snow} \succeq\succeq \text{Lili-128};$$

- l'existence de « clefs faibles » pour un système comme Lili-128 (par exemple la clef constituée uniquement de 0) explique seulement en partie ces très mauvais résultats statistiques (ils relèvent seulement du test de la constante affine). Un système comme Snow présente également de mauvais résultats alors qu'il ne comporte aucune clef faible.
- Les versions modifiées de Snow et de Lili-128 présentent également des faiblesses comparables.

	T_1^1	T_1^2	T_2^1	T_2^2
D^2	39,344.03	400,839.93	667729.02	1,028,048.45
$\chi^2_{0.001}$	6349.15			

TABLE 3.3 – Lili128 : résultats expérimentaux pour les tests T_1^d et T_2^d.

Les résultats de M. Saarinen [SAARINEN01] obtenus avec les tests des d-monômes sur certains candidats proposés dans le cadre du projet européen eSTREAM sont similaires. Ils ont permis d'exhiber des biais importants sur les systèmes *Mag, Frogbit, F-FCSR, DECIM, ZK-Crypt, Pomaranch, NLS* et *TSC-3*.

	T_1^1	T_1^2	$T_1^1\|p$	$T_1^1\|k$	$T_1^1\|pp$	$T_1^1\|kk$	$T_1^1\|pk$
Encr. + IP	35.06	37.65	34.75	35.57	35.41	33.47	33.25
Decr. + IP	33.68	33.93	34.75	39.74	35.41	39.12	29.95

TABLE 3.4 – DES : valeurs des estimateurs D^2

3.4.2 Systèmes de chiffrement par bloc

Nous allons maintenant considérer les résultats pour deux systèmes connus : les DES [DES77] et l'AES [AES]. Nous avons, pour ces systèmes, considéré les fonctions booléennes de sortie pour les deux opérations de chiffrement et de déchiffrement. Comme chaque fonction booléenne de sortie implique à la fois les variables de clef et de clair (dans le cas du chiffrement ; dans le cas du déchiffrement, le clair est remplacé par le chiffré), les test T_2^d $(d = 1, 2)$ ont été remplacés par les tests T_1^d relativement :

- au nombre n_1 de variables de texte clair d'un côté et de variables de clef de l'autre (notés $T_1^1|p$ et $T_1^1|k$).
- au nombre n_2 de 2-monômes impliquant respectivement les variables clair/clair, clair/clef ou clef/clef (les tests sont notés respectivement $T_1^1|pp$, $T_1^1|kk$ et $T_1^1|pk$).

Le DES.-

Le tableau 3.4 résume les résultats expérimentaux pour l'estimateur D^2 à 63 degrés de liberté. Les seuils de décision sont $\chi^2 = 82.52$ $(\alpha = 0.05)$, $\chi^2 = 92.01$ $(\alpha = 0.01)$ et $\chi^2 = 103.44$ $(\alpha = 0.001)$. Il est intéressant de noter que :

- le DES passe le test de la constante affine, quel que soit le mode et le niveau de signification ;
- la qualité statistique générale est légèrement différente – quoique l'on soit toujours dans la zone d'acceptation – selon que l'on considère l'opération de chiffrement ou de déchiffrement (en particulier quand on considère la clef séparément des variables de clair) ;
- les résultats généraux mettent en évidence l'impact de la permutation initiale IP du DES. Selon qu'elle est ou non utilisée, les performances statistiques sont différentes. Selon les résultats des

tests $T_1^1|S$ et $T_1^2|S$, nous avons ($\succeq$ signifie « meilleur que ») :

$$\{\text{DES Encr. - IP, DES Decr. + IP, DES Decr. - IP}\}$$
$$\succeq$$
$$\text{DES Encr. + IP.}$$

Ce résultat est d'autant plus intéressant que la communauté cryptologique généralement ne considère pas cette permutation. Les résultats prouvent néanmoins qu'elle contribue significativement à la qualité générale du DES.

L'AES.-

Pour cet algorithme, la version travaillant sur des blocs (clair ou chiffré) de 128 bits et des clefs de 128 bits a été considérée (cette version sera notée AES(128, 128)). Le tableau 3.5 donne les résultats expérimentaux détaillés concernant l'estimateur D^2 à 127 degrés de liberté. Le seuil de décision pour un risque $\alpha = 0.05$ est donné par $\chi^2 = 159.59$.

Les principales remarques qui peuvent être tirées de ces résultats (partiels) sont les suivantes :

- l'AES passe avec succès le test de la constante affine dans tous les modes et pour tous les niveaux de signification ;
- la qualité statistique globale de l'AES (128, 128) est plutôt bonne. Toutefois, à des valeurs de d plus grandes, des résultats encore très partiels pour les tests $T_1^1|S$ et $T_1^2|S$ semblent indiquer des biais pour certains ensembles S, notamment ceux contenant les fonctions booléennes de sortie d'indice 52 et 110 (pour le chiffrement). Ces résultats doivent être confirmés par une analyse statistique poussée. Cette analyse constitue, sur l'AES, et plus généralement sur les schémas par bloc, une partie importante de mes travaux de recherche actuels et futurs (voir section 3.5) ;
- Il n'existe pas, pour les tests mis en œuvre de différences significatives entre les fonctions de chiffrement et de déchiffrement.

3.4.3 Fonctions de hachage

Les fonctions de hachage suivante ont été testées : SHA-0 [FIPS180], SHA-1 [FIPS181], Ripemd160 [RMD160], MD4 [MD4], MD5 [MD5],

	T_1^1	T_1^2	$T_1^1\|p$	$T_1^1\|k$	$T_1^1\|pp$	$T_1^1\|kk$	$T_1^1\|pk$
Chiffrement	59.61	71.32	57.84	61.51	64.47	72.34	62.39
Déchiffrement	67.38	62.27	67.21	70.70	71.26	60.11	47.27

TABLE 3.5 – AES(128, 128) : Valeurs des estimateurs D^2

Ripe-MD [RMD128] et Haval [Haval93] (pour cette dernière, toutes les versions ont été testées). Toutes les fonctions de hachage ont passé les

Fonctions de hachage	T_1^1		T_1^2	
	D^2	χ^2	D^2	χ^2
SHA-1	76.87	189.52	70.89	189.52
(5,160)-haval	76.34		79.76	
Ripemd160	77.51		66.72	
(4,160)-haval	83.52		74.18	
(3,160)-haval	83.79		64.28	
SHA-0	97.08		74.50	

TABLE 3.6 – Résultats expérimentaux pour les tests T_1^1 et T_2^1 ($\alpha = 0.05$).

tests appliqués, et ce quel que soit le niveau de signification considéré. Toutefois, quelques remarques de détail peuvent être faites si l'on considère la convergence de chacun de ces tests.

- les différentes fonctions de hachage peuvent être classées selon leur qualité « aléatoire » respective. Ainsi, en considérant les résultats du test T_1^1, le plus inéressant, nous avons le classement suivant ($\succeq$ signifie « meilleur que ») :
  - Haché de 160 bits : SHA-1 $\succeq$ (5, 160)-haval $\succeq$ Ripemd160 $\succeq$ (4, 160)-haval $\succeq$ (3, 160)-haval $\succeq$ SHA-0.
  - Haché de 128 bits : (5,128)-haval $\succeq$ Ripe-MD $\succeq$ MD5 $\succeq$ (4,128)-haval $\succeq$ (3,128)-haval $\succeq$ MD4.
- SHA-1 présente une meilleure qualité statistique globale que SHA-0, notamment au degré $d = 1$. L'inclusion du bit de rotation dans l'expansion du bloc d'entrée de 16 à 80 mots de 32 bits constitue une réelle amélioration que ces tests ont permis de caractériser de

manière formelle.
- Pour la famille des fonctions Haval, la qualité aléatoire augmente avec le nombre de rounds.

Les tableaux 3.6 et 3.7 présentent les résultats des tests T_1^d et T_2^d $(d = 1, 2)$ pour les fonctions de hachage de 160 bits (niveau de signification $\alpha = 0.05$).

Fonctions de hachage	T_2^1		T_2^2	
	D^2	χ^2	D^2	χ^2
SHA-1	0.04	5.99	0.42	5.99
(5,160)-haval	0.17		2.02	
Ripemd160	5.24		2.66	
(4,160)-haval	1.77		3.51	
(3,160)-haval	1.05		5.50	
SHA-0	3.26		0.42	

TABLE 3.7 – Résultats expérimentaux pour les tests T_1^2 et T_2^2 ($\alpha = 0.05$).

3.5 Problèmes ouverts et axes de recherche futurs

Si la modélisation des quantités produites par un système de chiffrement à l'aide de fonctions booléennes s'est révélée d'ores et déjà très intéressante, il reste encore un certain nombre de problèmes ouverts et de points sur lesquels je souhaite travailler dans les années à venir. Je citerai les deux principaux.

- l'étude des fonctions booléennes de sortie pour des monômes de degré $d > 3$. Les contraintes en complexité mémoire rendent cette étude inenvisageable par les approches classiques et « naïve ». L'approche partielle à l'aide de plans factoriel d'expériences [Mont96] est une voie qui m'apparait comme intéressante et prometteuse. Cette approche doit également permettre d'exhiber des éventuelles dépendances entre certaines fonctions booléennes de sortie. Autrement dit, il s'agira d'identifier certains sous-ensembles

S de ces fonctions pour lesquels le test $T_i|S$ mettra en évidence des biais. L'utilisation de plans factoriels a pour objectif de mettre en évidence l'existence éventuelle de variables de clef (ou de clair dans le cas des systèmes par bloc) qui exercent une influence plus importante que les autres ;

– l'exploitation cryptanalytique des faiblesses. Si la mise en évidence de faiblesses statistiques est en soi un résultat intéressant, il n'est qu'une étape liminaire dans la cryptanalyse d'un système. La présence de biais dans la distribution de d-monômes, pour certaines valeurs de d doit être traduite en propriétés facilement exploitables par le cryptanalyste. Le fait que l'on soit obligé de ne considérer que des FAN tronquées à l'ordre d, interdit une approche algébrique (par bases de Gröbner par exemple [FAUG03, ArsFaug04]) du moins traditionnelle. Mon intention est de considérer certains sous-ensembles de fonctions booléennes de sorties (tronquées à l'ordre d) pour lesquelles certaines propriétés structurelles (notamment de treillis booléens [Grätzer03]) permettent de déduire avec une probabilité p la plus éloignée possible de $\frac{1}{2}$, des bits d'information sur la clef.

Chapitre 4

L'attaque par codes à répétition

4.1 Introduction

En octobre 2000, le département du commerce américain (NIST) a choisi, comme remplacement du standard – de fait – de chiffrement DES, l'algorithme de chiffrement par blocs AES. Ce remplacement s'est accompagné d'une généralisation, à l'échelle mondiale, de l'utilisation de ce type de chiffrement et de cet algorithme tout particulièrement.

L'évaluation de sécurité de l'AES, ainsi que des algorithmes concurrents, a été conduite essentiellement par référence aux techniques de cryptanalyses connues et à leurs variantes : la cryptanalyse linéaire de Matsui [Matsui94], la cryptanalyse différentielle [BiSha91] ou leurs différentes variantes... Aucun résultat significatifs n'a permis de ce point de vue de remettre en cause leur niveau de sécurité. Cependant, alors qu'il existe une théorie conséquente, et un savoir-faire en terme d'ingénierie de la cryptographie, concernant l'autre famille de systèmes de chiffrement – les systèmes dits par flot, utilisés par les militaires et les services gouvernementaux de manière quasi-exclusive s'agissant de trafics sensibles – la situation est bien différente en ce qui concerne les systèmes par blocs. La consultation de n'importe quelle monographie de cryptologie montre que les fondements théoriques spécifiques aux systèmes par blocs sont très réduits. A l'exception de quelques techniques de cryptanalyse et de quelques éléments sur les propriétés des

primitives de base – éléments empruntés souvent à la théorie existante de systèmes par flot – aucune réflexion théorique véritable n'existe pour les systèmes par blocs. Ainsi, ne sait-on toujours pas quel est le nombre de tours optimal ou à défaut minimal que doit avoir un tel système. Il manque une formalisation de la structure globale d'un tel système, et en particulier une approche combinatoire. Il est actuellement impossible de prévoir ou de caractériser la présence éventuelle de dégénérescences survenant lors de la combinaison de primitives qui isolément ont de bonnes propriétés.

Cette absence de formalisation au niveau de la structure globale de tels systèmes interdit d'infirmer la possibilité que le concepteur d'un tel algorithme n'y a pas caché des trappes autorisant une cryptanalyse plus facile pour celui qui en connait l'existence et la nature – ce problème, longtemps évoqué pour le DES, n'a jamais reçu de solutions. La conception de systèmes de chiffrement par blocs contenant des trappes *calculatoirement* indétectables est une chose possible [RijPren97]. De la même manière, quel concepteur peut garantir que le système qu'il propose, conçu en prenant les primitives les meilleures d'un point de vue cryptographique, ne produit pas des dégénérescences une fois ces primitives combinées. En particulier, comment prouver qu'il n'existe pas des approximations linéaires globales dans l'AES qui permettrait, par exemple, une cryptanalyse linéaire opérationnelle. Les concepteurs de l'AES, eux mêmes [AES02, Chap 7, §2 page 124] ont toujours reconnu l'existence de ce problème.

Actuellement la plupart des techniques de cryptanalyse utilisent de près ou de loin la notion de corrélation ou de biais statistiques par rapport à une distribution idéale – le plus souvent la loi uniforme. Ces cryptanalyses ne sont possibles que si d'une part, nous sommes capables d'identifier des tels biais ou corrélations, mais également si ces derniers sont suffisamment importants pour réellement mettre à mal la sécurité générale du système et conduire à son abandon. Dans le domaine de la cryptanalyse opérationnelle, ce n'est pas tant la valeur moyenne maximale d'une corrélation ou d'un biais qui importe mais le potential maximal de corrélation lié aux éléments en cause (clair et clef) lors d'une transmission donnée [AES02]. Autrement dit, ce qui compte c'est de spécialiser un modèle à une classe donnée soit de clair soit de clef. A titre d'exemple, dans la cryptanalyse linéaire, Matsui

[Matsui94], considère l'équation suivante, pour le DES à 8 tours

$$\begin{array}{c} P_H[7,18,24] \oplus C_L[7,18,24,29] \oplus C_H[15] \\ \oplus F_1(P_L,K_1)[7,18,24] \oplus F_8(C_L,K_8)[15] \\ \stackrel{p}{\simeq} \\ K_3[22] \oplus K_4[44] \oplus K_5[22] \oplus K_7[22] \end{array} \tag{4.1}$$

qui est vérifiée avec une probabilité moyenne $p = \frac{1}{2}(1+\varepsilon)$. Matsui donne une valeur (théorique) de $\varepsilon_{tho.} = 0.00763$. Or l'évaluation de la dépendance de cette corrélation vis-à-vis de la clef, sur des milliers de clefs, produit de grandes variations, comme indiqué dans le tableau 4.1. Une fluctuation importante des scores a été observée avec une moyenne

Intervalles	*Pourcentage*
$0.0090 < \varepsilon_{obs.}$	10
$0.0080 < \varepsilon_{obs.} \leq 0.0090$	26
$0.0075 < \varepsilon_{obs.} \leq 0.0080$	20
$0.0070 < \varepsilon_{obs.} \leq 0.0075$	16
$0.0070 < \varepsilon_{obs.} \leq 0.0075$	26
$0.0050 < \varepsilon_{obs.} \leq 0.0060$	2

TABLE 4.1 – Fluctuation du ε observé

conforme à la théorie ($\varepsilon_{obs} = 0.00760$). L'étude du tableau montre que les clefs peuvent être classées en trois catégories. Il existe donc une forte dépendance vis-à-vis de la clé globale et pas seulement vis-à-vis des bits impliqués dans l'équation. Selon Harpès et al. [HKM95], Matsui n'avait pas considéré de fonction de clé *maximisante*, ce qui rendait son estimation pessimiste. Le même effet s'agissant des blocs de clair peut être observé.

Ces constatations incitent donc à privilégier une vision locale, conditionnée par certaines classes de clef ou de clair, pour cryptanalyse un système. Il suffit ensuite de considérer une partition des espaces d'entrée (clef et clair) pour disposer d'une collection d'outils de cryptanalyse. Dans le cas des systèmes de chiffrement par blocs, cette approche a été considérée pour l'ensemble des blocs de clair, lesquels prennent, par construction, un part active dans le chiffrement. Cette part a été modélisée et quantifiée dans le chapitre précédent et dans [Fil02].

Dans ce chapitre, nous proposons un nouveau modèle théorique concernant la cryptanalyse des systèmes de chiffrement par blocs et fondée sur certaines propriétés potentiellement mauvaises d'un point de vue cryptographiques, des fonctions booléennes de sortie de tels systèmes. Nous supposons que la restriction de ces fonctions à certains sous-ensembles de blocs de clair permettent d'observer des biais statistiques assez fort pour mener la cryptanalyse. Nous avons donc appelé cette nouvelle attaque *cryptanalyse clair-dépendante par code à répétition* (PDRC). Nous allons ensuite montrer qu'il est possible, avec ce nouveau modèle, de généraliser la cryptanalyse linéaire classique par une version généralisée du modèle : la cryptanalyse RC dans laquelle les blocs de clair cette fois-ci interviennent (aucune dépendance vis-à-vis du clair n'est alors supposée). Cette généralisation se révèle plus efficace que la cryptanalyse linéaire et permet d'aller bien au-delà des limitations qui sont imposées à cette dernière.

4.2 Notations et rappels théoriques

4.2.1 Codes à répétition

Considérons un *canal binaire symétrique* de paramètre p, utilisé comme canal de transmission pour des messages binaires. Rappelons que la matrice des probabilités de modification des bits est une matrice carrée d'ordre deux dont les coefficients sont donnés par $a_{i,j} = q$ si $i \neq j$ et $a_{i,j} = p = 1 - q$ autrement.

En d'autres termes, si un bit b_t est transmis via ce canal alors le bit $\hat{b}_t = b_t \oplus e_t$ sera celui effectivement reçu, et ce avec une probabilité p (probabilité d'erreur du canal). Pour corriger de telles erreurs de transmissions, des codes détecteurs d'erreurs sont communément utilisés, et parmi eux, en particulier les codes dits linéaires. Un code linéaire binaire $[n, k, d]$ est un sous-espace vectoriel de $\mathbb{F}_2^n$, de dimension k. Sa *distance minimale* d est le plus petit poids de Hamming des mots de code non nuls – c'est-à-dire les vecteurs de n bits – de ce code. Autrement dit, $d = min_{x \in \mathbb{F}_2^n}\{wt(x)\}$ avec $wt(x)$ désignant le nombre de bits non nuls dans le vecteur binaire $x = (x_1, \ldots, x_n)$. Un résultat très connu [1] définit le nombre d'erreurs survenues sur un mot de code durant une transmission et qui peuvent être corrigées automatiquement

par un code de distance minimale d. Ce nombre est donné par $\frac{d-1}{2}$.

Un n-code à répétition, défini sur un ensemble de deux symbôles, est un $[n, 1, n]$ code linéaire. Il est composé de deux mots de codes seulement, chacun d'entre eux étant formé de n symbôles identiques. Le codage à l'émission consiste à transmettre chaque bit répétés n fois. Le décodage, dans le cas où $q > p$, se fait par maximum de vraisemblance. Le message, lors de la réception, est découpé en mots de n bits et l'on compte, dans chaque vecteur quel est de nombre de bits, zéro ou un, le plus souvent représenté. Chaque vecteur binaire reçu est décodé en le symbôle 0 si sa distance de Hamming au vecteur nul est moindre que celle au vecteur $(1, 1, 1, \ldots, 1)$, sinon il est décodé en le symbôle 1. Ainsi le décodage par maximum de vraisemblance (décodage MLD) est en fait un décodage selon le principe de majorité.

Exemple 4 *Considérons le message* 01100 *et un code 3-répétition. Après codage à l'émission, la séquence* 000 111 111 000 000 *est transmise. Après action du bruit sur le canal de transmission, la séquence* 010111101110100 *est reçue et décodée en le message* 01110. *Il ne subsiste aucune erreur résiduelle.*

Ces codes sont les plus facilement décodables parmi tous les codes connus et ceux offrant un haut niveau de sureté. De plus, les codes à répétition sont les codes les plus efficaces pour lutter contre des valeurs élevées de probabilité de bruit p [McEl77]. C'est cette propriété qui se révélera particulièrement intéressante pour la cryptanalyse par codes à répétition.

Proposition 13 *[McEl77] Soit $n = 2s+1$. Alors le n-code à répétition corrige au plus s erreurs et est un code parfait. La probabilité d'erreur résiduelle (après décodage) est donnée par*

$$P_{err} = \sum_{i=s+1}^{n} \binom{n}{i} p^i \cdot q^{n-i}. \tag{4.2}$$

Le terme parfait signifie que tout mot (vecteur) de l'espace $\mathbb{F}_2^n$ des vecteurs possibles, est décodable. Au final, la probabilité de décodage avec succès est donnée par

$$P_{succ} = 1 - P_{err}$$

Il est intéressant de signaler que si $p < \frac{1}{2}$, alors $P_{err,2s+1}$ tend vers 0 lorsque $s \to \infty$.

4.2.2 Systèmes de chiffrement par blocs et cryptanalyse linéaire

Un système de chiffrement travaillant sur des blocs de texte clair (respectivement de cryptogramme) P_i (respectivement C_i) de m bits, et utilisant une clef secrète K de n bits (nous noterons ce type de système un (m,n)-système par blocs) est une application surjective de $\mathbb{F}_2^m \times \mathbb{F}_2^n$ vers $\mathbb{F}_2^m$. De plus, chaque fois qu'une clef K est fixée, la restriction de cette application est une permutation sur $\mathbb{F}_2^m$. Il est donc possible de décrire un (m,n)-système par blocs comme un ensemble de 2^n permutations sur $\mathbb{F}_2^m$. Notons que cela ne représente qu'une part infime des permutations possibles (au total $(2^m)!$).

La cryptanalyse linéaire [Matsui94] des systèmes de chiffrement par blocs est une attaque à clair connu, dans laquelle un nombre important de couples de blocs clair/chiffré sont utilisés pour déterminer la valeur d'un sous-ensemble de bits de la clef utilisée, réduisant ainsi la complexité finale de la recherche euxhaustive.

Une condition d'utilisation de la cryptanalyse linéaire sur un système de chiffrement par bloc est de connaitre une ou plusieurs relations probabilistes, linéaires « effectives » liant des bits de bloc de clair P_i, de bloc de chiffré C_i et de clef K de la forme :

$$< P_i, u > \oplus < C_i, w > \overset{p}{\cong} < K, v > \tag{4.3}$$

où $< ., . >$ désigne le produit scalaire usuel sur $\mathbb{F}_2^m$. Si cette équation est vérifiée avec une probabilité $p \neq \frac{1}{2}$ alors en évaluant le membre gauche de l'équation (4.3) pour un grand nombre N de couples de blocs de clair et de chiffré, le membre droit de cette équation peut être évalué par un simple décodage par maximum de vraisemblance. Un bit ou plusieurs bits d'information est alors obtenu sur la clef (un par équation) Cette cryptanalyse est effective si la déviation $|p - \frac{1}{2}|$ est suffisamment importante. Dans [Matsui94], il est établi que la probabilité d'erreur (évaluation fausse du membre droit de l'équation (4.3)) devient faible dès lors que $N > |p - \frac{1}{2}|^{-2}$.

D'un point de vue pratique, l'approximation linéaire décrite par l'équation (4.3) est obtenue en « chaînant » des approximations linéaires obtenues pour un nombre réduit de tours de l'algorithme de chiffrement, lesquelles sont établies en considérant des biais statistiques

locaux dans les primitives constitutives du systèmes (boites de substitution pas exemple). Cela implique que d'autres approximations de ce type, éventuellement avec des probabilités plus fortes, existent peut être et dépendent de la structure globale du système (effets de dégénérescence insoupçonnée résultant de l'itération successive de primitives, par exemple) [AES02, Chap 7 et §2 de la page 124]. Si elles existent, ces corrélations sont actuellement hors de portée des capacités de calcul. En effet, pour un (m, n)-système par blocs, cela requiérait une recherche exploratoire d'une complexité en $\mathcal{O}(2^{2m+n})$. Le théorème de Parseval permet au minimum d'affirmer que de telles corrélations existent.

Proposition 14 *(*Relation de Parseval*) Soit une fonction booléenne à n variables. La relation de Parseval donne pour la transformée de Walsh-Hadamard :*

$$\sum_{u \in \mathbb{F}_2^n} (\widehat{\chi_f}(u))^2 = 2^{2m}$$

On peut envisager le second membre de cette relation comme un invariant pour toutes les fonctions booléennes à n variables. La répartition locale de cet invariant (« l'énergie » totale de la fonction), c'est à dire la répartition des valeurs non nulles $\widehat{\chi_f}(u)$, va déterminer les propriétés de corrélation plus ou moins bonnes de la fonction, d'un point de vue cryptographique. Cette transformée de Walsh-Hadamard, qui a été définie dans la section 2.2.2, est extrêmement utile pour étudier les fonctions booléennes et caractériser la notion de corrélation.

4.3 Cryptanalyse de systèmes de chiffrement par blocs par codes à répétition

4.3.1 Chiffrement par blocs et codes à répétition

Considérons une propriété donnée $\mathcal{I}$ et notons $P_{\mathcal{E}}[\mathcal{I}]$ la probabilité que la propriété $\mathcal{I}$ soit réalisée sur un ensemble $\mathcal{E}$. Cette propriété peut être de n'importe quelle nature (algébrique, combinatoire...). Un système de chiffrement par blocs peut alors être cryptanalysé si, pour des propriétés $\mathcal{I}$, nous avons $P_{\mathbb{F}_2^{m+n}}[\mathcal{I}] \neq \frac{1}{2}$.

Chaque clef K de l'espace des clefs $\mathcal{K} = \mathbb{F}_2^n$ définit une permutation sur l'ensemble $\mathbb{F}_2^m$. En conséquence, des bits d'information sur K

peuvent être retrouvés si $P_{\mathbb{F}_2{}^m}[\mathcal{I}_K] \neq \frac{1}{2}$ en notant $\mathcal{I}_K$ la propriété $\mathcal{I}$ appliquée à la seule clef K. Ainsi, dans ces conditions, si on peut exhiber une telle propriété, valable pour tous les $K \in \mathcal{K}$ (denoted $\mathcal{I}_\mathcal{K}$), alors nous disposerions d'une attaque contre le système en question. Dans le cas particulier de la cryptanalyse linéaire, $\mathcal{I}_\mathcal{K}$ correspond en fait à une équation linéaire probabiliste.

Considérons à présent l'espace des blocs de texte en clair $\mathcal{P} = \mathbb{F}_2^m$ et une partition $(\mathcal{P}_i)_{i \leq 2^k}$ de $\mathcal{P}$ pour une valeur donnée de $k \in \mathbb{N}$. Sans restriction conceptuelle, nous supposerons que $|\mathcal{P}_i| = 2^{m-k}$ pour tout i. Maintenant supposons qu'il existe un sous-ensemble de blocs $\mathcal{P}_i$ tels que $\mathcal{I}_{\mathcal{P}_i}[\mathcal{I}_\mathcal{K}] = p_i \neq \frac{1}{2}$. Comme la clef de chiffrement $K \in \mathcal{K}$ est la même pour tous les blocs de texte chiffré produits, il est alors possible d'assimiler le processus de chiffrement à une transmission via un canal binaire symétrique de paramètre p_i, dans lequel le « bruit » est produit par les blocs de texte clair de $\mathcal{P}_i$ (voir figure 4.1). Le canal ainsi modélisé est directement et intimement déterminé par les $\mathcal{P}_i$. Il suffit de trouver un sous-ensemble de tels $\mathcal{P}_i$ pour lequel le modèle statistique décrivant ce bruit est constant (groupe linguistique ou langue donnés, propriété combinatoire ou algébrique sur les bits de clair d'un bloc...). La version

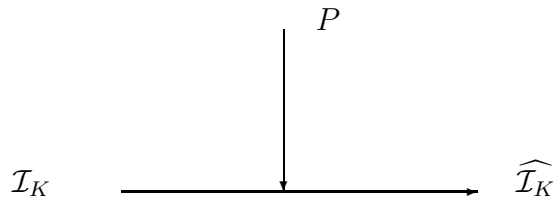

FIGURE 4.1 – Canal binaire symétrique et système de chiffrement par blocs

bruitée $\widehat{\mathcal{I}_K}$ de $\mathcal{I}_K$ est une fonction $f(C)$ – qui peut être non linéaire et avoir une structure complexe – dont les valeurs d'entrée sont des bits de blocs de chiffré. En d'autres termes, le chiffrement de N blocs de clair $P \in \mathcal{P}_i$ est équivalent, selon ce modèle, à la transmission de la même information $\mathcal{I}_K$ sur la clef, au moyen d'un N-code à répétition , au travers d'un canal binaire symétrique de paramètre p_i. Comme illustré sur la figure 4.1, cela signifie que sur $\mathcal{C}_i$ nous avons $P[\mathcal{I}_K = \widehat{\mathcal{I}_K}] = 1 - p_i$.

L'objectif du concepteur du système est de disposer d'un ensemble

de permutations sur $\mathcal{C}$ telles qu'aucune propriété $\mathcal{I}$ sur tout ou partie de la clef ne puisse être exhibée et identifiée facilement sinon que par une recherche exploratoire. Mais la situation est probablement très différente si l'on considère une restriction à un sous-ensemble $\mathcal{C}_i \subsetneq \mathcal{C}$. Si nous avons l'égalité suivante,

$$P_{\mathcal{C}}[\mathcal{I}] = \sum_{i=0}^{2^k} P_{\mathcal{C}_i}[\mathcal{I}] \cdot P[\mathcal{C}_i] = \frac{1}{2}$$

il est toutefois tout à fait possible qu'il existe de nombreux $P_{\mathcal{C}_i}[\mathcal{I}]$ qui soient différents de $\frac{1}{2}$ (il suffit que $\sum_i \epsilon_i = \sum_i (p_i - \frac{1}{2}) = 0$). Cette éventualité semble pouvoir être motivée en partie par le fait que le nombre réel de permutations possibles sur $\mathcal{C}$ qui décrivent un système de chiffrement par blocs est extrêmement négligeable par rapport au nombre total de permutations possibles sur le même espace de blocs de clair. Quelle garantie a le concepteur du système que les permutations réalisées respectent tous les critères cryptographiques (par exemple en terme de corrélation) connus désirables tout en tenant compte de la structure globale du système et non plus sa seule vision locale (qualité des primitives cryptographiques).

4.3.2 Description de l'attaque PDRC

Avec les notations et le modèle définis dans la section précédente, nous pouvons à présent décrire la technique de *cryptanalyse clair-dépendante par code à répétition* [1] d'une manière très simple. Il est important de conserver à l'esprit, encore une fois qu'une indépendance statistique locale vis-à-vis du texte clair (en vertu d'une restriction à un sous-ensemble particulier $\mathcal{C}_i \subsetneq \mathcal{C}$) permet de considérer également une *cryptanalyse à chiffré seul*. Nous allons présenter un premier algorithme A.1 qui met en œuvre un unique code à répétition. La complexité de l'algorithme A.1 est facile à évaluer. Ce dernier effectue seulement N évaluations de la fonction f. Par conséquent, la complexité est en $\mathcal{O}(N)$. Comme N est la longueur du code à répétition considéré, selon ce qui a été montré dans la section 4.2.1, cette complexité dépend finalement

1. L'acronyme PDRC sera conservé par référence à la publication originale [Fil05].

Entrée : $N = 2p + 1$ blocs chiffrés C_j chiffrés par une clef K à partir des blocs de clair $P_j \in \mathcal{C}_i$ $(1 \leq j \leq N)$ et une information probabiliste $\mathcal{I}_\mathcal{K}$ telle que $\mathcal{I}_\mathcal{K} \overset{p_i}{\cong} f(C_j)$ pour une fonction f et pour tous les j.
Sortie : Le bit d'information sur $\mathcal{I}(K)$ pour la clef utilisée.

Initialiser le compteur $ct \leftarrow 0$.
Pour i = 1 to N faire
    Calculer $f(C_i)$.
    Si $f(C_i) = 1$ alors $ct++$.
    Fin si
Fin pour
Si $ct \geq \frac{N+1}{2}$
  alors $\mathcal{I}(K) = 1$
  sinon $\mathcal{I}(K) = 0$.
Fin si.

TABLE 4.2 – Algorithme A.1 de cryptanalyse par code á répétition

seulement de p_i et de p_{succ}, la probabilité de succès de décoder correctement $\mathcal{I}(K)$.

Après recherche, aucune formule suffisamment générale n'a été trouvée dans la littérature permettant de lier directement N aux paramètres p_i et p_{succ}. Il est possible de tabuler cette quantité pour certaines valeurs de ces paramètres. D'une manière générale, il est facile de montrer que pour une valeur fixe de p_i, la probabilité de succès p_{succ} augmente avec N.

Exemple 5 *Considérons* $p_i = 0.49999$. *Alors* $p_{succ} = 0.501784$ *pour* $N = 49999$ *alors que nous avons* $p_{succ} = 0.5025$ *pour* $N = 99999$.

Afin d'augmenter la probabilité de décoder avec succès, un second algorithme a été imaginé. L'algorithme A.2 considère la notion de codes à réprétition *concaténés*. La concaténation de codes a été introduite par Forney en 1966 [FOR66] et généralisée par Zinov'ev en 1976 [2]. Le

principe est d'utiliser deux codes comme représenté dans la figure 4.2. La combinaison d'un code en entrée, d'un canal et d'un décodeur en

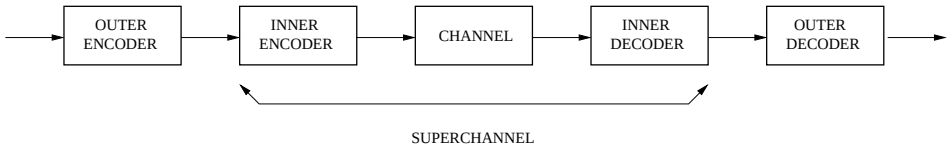

FIGURE 4.2 – Codes concaténés

sortie peut être considéré comme formant un nouveau canal, appelé *supercanal*. L'objectif est d'augmenter la capacité de correction du code d'entrée en utilisant un second code. Lorsque la transmission passe par un canal très bruité – ce qui correspond au cas de la cryptographie –, les codes à répétition sont des codes de choix pour constituer des codes extérieurs dans des codes concaténés classiques.

Dans notre approche cryptanalytique, le supercanal est en fait un canal binaire symétrique de paramètre $p' = 1 - P_{\text{succ}}$ produit par la capacité d'erreur résiduelle du codeur interne. On itére le processus de décodage en utilisant cette fois un code à répétition extérieur pour le supercanal. Le pseudo-code de l'algorithme A.2, donné dans la table 4.3 a une complexité en $\mathcal{O}(N_1 \cdot N_2)$.

Il est à noter que si généralement les codes concaténés ont une meilleure probabilité de succès finale que des codes simples, ce n'est pas systématiquement le cas quand les deux codes sont des codes à répétition. Nous pouvons cependant considérer le résultat suivant.

Proposition 15 *Soit $N = 2p + 1$ blocs de texte chiffré. L'algorithme A.2 a une probabilité de succès supérieure à celle de l'algorithme A.1.*

Preuve.
Ecrivons $N = N_1 \cdot N_2$ où N_1 et N_2 sont deux entiers impairs. Comme l'algorithme A.1 utilise un $[N, 1, N]$-code à répétition, sa capacité de corrections est donnée par $\frac{N-1}{2} = \frac{(N_1 \cdot N_2 - 1)}{2}$.

Supposons maintenant que dans l'algorithme A.2, le code du supercanal soit un $[N_1, 1, N_1]$-code à répétition et que le code extérieur soit un $[N_2, 1, N_2]$ code à répétition.

Entrée : $N_1 \cdot N_2$ (N_1, N_2 impairs) blocs C_j de texte chiffrés, obtenus avec une clef K à partir de blocs de clair $P_j \in \mathcal{C}_i$ $(1 \leq j \leq N)$ et une information probabiliste $\mathcal{I}_\mathcal{K}$ telle que $\mathcal{I}_\mathcal{K} \overset{p_i}{\cong} f(C_j)$ pour un g donné et tous les j.
Sortie : Le bit d'information sur $\mathcal{I}(K)$ pour la clef utilisée.

Initialiser le compteur $ct_1 \leftarrow 0$.
Pour $1 \leq k \leq N_1$ faire
    Initialiser le compteur $ct_2 \leftarrow 0$.
    Pour $1 \leq j \leq N_2$ (k-ieme groupe) faire
        Calculer $f(C_{k,j})$.
        Si $f(C_{k,j}) = 1$ alors $ct_2 + +$.
    Fin pour
    Si $ct_2 \geq \frac{N_2+1}{2}$
       alors $\mathcal{I}(K) = 1$
       sinon $\mathcal{I}(K) = 0$.
    Fin si Si $\mathcal{I}(K) = 1$ alors $ct_1 + +$.
Fin pour Si $ct_1 \geq \frac{N_1+1}{2}$
            alors $\mathcal{I}(K) = 1$
            sinon $\mathcal{I}(K) = 0$.
Fin si.

TABLE 4.3 – Algorithme A.2 de cryptanalyse par code á répétition concaténés

Le supercanal peut donc corriger jusqu'à $\frac{N_1-1}{2}$ erreurs alors que le code extérieur corrigera au plus $\frac{N_2-1}{2}$ erreurs.

Considérons alors le scénario en pire cas. Durant le processus de décodage par le code extérieur, exactement $\frac{N_2-1}{2}$ erreurs sont survenues. Elles correspondent en fait chaque fois à une décision erronée lors du décodage de $\mathcal{I}(K)$ à la sortie du supercanal. En d'autres termes, cela signifie que chaque fois au plus N_1 sont survenues.

Au contraire, pour chacune des $N_2 - \frac{N_2-1}{2}$ décisions correctes par le code extérieur, au plus $\frac{N_1-1}{2}$ erreurs peuvent avoir eu lieu. Au final, au plus

$$N_1 \times \left(\frac{N_2-1}{2}\right) + \left(N_2 - \frac{N_2-1}{2}\right) \times \left(\frac{N_1-1}{2}\right)$$

erreurs peuvent survenir tandis que la décision finale est correcte (décodage sans erreur).

Après simplification de l'expression précédente, nous obtenons le nombre maximum d'erreurs suivant :

$$\frac{3N_1N_2 - N_2 - N_1 - 1}{4} = N_1N_2 - \frac{N_1N_2 + N_2 + N_1 - 1}{4}$$

Il s'agit là précisément du nombre total d'erreurs qui peuvent corrigées avec succès par le code concaténé. D'où le résultat. ■

L'algorithme A.1 reste néanmoins intéressant dans la mesure où il permet d'établir une borne inférieure pour la probabilité de succès de l'algorithme A.2. En effet, la formule (4.2) est difficile à calculer directement, pour l'algorithme A.2, s'agissant de valeurs importantes trop importantes de N.

4.3.3 Critère de résistance contre l'attaque PDRC

La cryptanalyse clair-dépendante par codes à répétition n'est possible que s'il existe un sous-ensmble $\mathcal{C}_i \subset \mathcal{C}$ tel que $P_{\mathcal{C}_i}[\mathcal{I}_\mathcal{K}] \neq \frac{1}{2}$ pour une propriété $\mathcal{I}$ donnée. Cette considération permet de définir le critère de résistance vis-à-vis de la cryptanalyse PRDC.

Proposition 16 *Soit S un (m,n)-système de chiffrement par blocs et soit une propriété $\mathcal{I}$ concernant un sous-ensemble de bits de clefs, relativement à un sous-ensemble de bits de chiffré. S est dit immune à la*

cryptanalyse clair-dépendante par codes à répétition, relativement à la propriété $\mathcal{I}$ si et seulement si $\forall j \in \mathbb{N}$ la partition $(\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_j)$ de $\mathcal{C}$ est telle que

$$\forall k \leq j, \quad P_{\mathcal{C}_k}[\mathcal{I}] = \frac{1}{2}.$$

Tout le travail du cryptanalyste consiste à trouver une propriété $\mathcal{I}$ exploitable et un sous-ensemble de blocs de texte clair permettant de réaliser l'attaque de S par cryptanalyse par codes à répétition. Si cette tâche est complexe, du point de vue du cryptographe, elle l'est encore plus. Confronté à des problèmes de nature combinatoire, cette complexité peut être résumé par les quatre problèmes ouverts qui suivent.

Problèmes ouverts

1. *Problème d'immunité à la cryptanalyse PDRC* .- Soit une propriété $\mathcal{I}$, est-il possible de concevoir un système S qui soit résistant à la cryptanalyse par code à répétition relativement à $\mathcal{I}$?
2. *Problème de trappe faible* .- Fixons un sous-ensemble de blocs de clair $\mathcal{C}_i \subset \mathcal{C}$, est-il possible de concervoir un système S tel que $P_{\mathcal{C}_i}[\mathcal{I}] \neq \frac{1}{2}$ pour une propriété « intéressante » $\mathcal{I}$ (la trappe) ?
3. *Problème de la trappe forte* .- Soit $\mathcal{I}$ une propriété, est-il possible de concevoir un système S tel que pour tout $\mathcal{C}_i \subset \mathcal{C}$ nous ayons $P_{\mathcal{C}_i}[\mathcal{I}] \neq \frac{1}{2}$?
4. *Faisabilité de la cryptanalyse PDRC* .- Soit S un système et $\mathcal{C}_i$ un sous-ensemble de blocs de clair, est-il possible de trouver une ou plusieurs propriétés $\mathcal{I}$ utilisable pour la cryptanalyse clair-dépendante par codes à répétition de S.

Les problèmes 2 et 3 concernent la possibilité de cacher une trappe $\mathcal{I}$ dans le système S. Autrement dit, si cela est effectivement, la connaissance d'une ou plusieurs propriétés $\mathcal{I}$ permet à celui qui détient ce secret de pouvoir retrouver plus facilement qu'une recherche exhaustive, un ou plusieurs bits d'information sur la clef. Enfin, du point de vue du cryptanalyste – qui ne connait, lui, a priori aucune trappe – le problème 4 est essentiel. Nous pouvons énoncer la conjecture suivante.

Conjecture 1 *Il existe toujours une propriété $\mathcal{I}$ pour laquelle un système de chiffrement par bloc S n'est pas immunisé contre la cryptanalyse par code à répétition.*

Si cette conjecture est vérifiée, cela implique que les systèmes de chiffrement par blocs ne sont pas conceptuellement pas sûrs. Le simple fait de réutiliser la clef de bloc en bloc constitue une faiblesse permettant de retrouver des bits d'informations sur la clef, moyennant un grand nombre de blocs de chiffré. Si l'on considère le thèorème de Parseval (voir Proposition 14) et des propriétés $\mathcal{I}$ de approximations linéaires, la théorie indique que cette conjecture est effectivement vérifiée. Tout le problème réside dans la capacité à opérationnellement trouver une propriété $\mathcal{I}$ adéquate. Comme les outils algébriques actuels ne permettant pas de la faire (problème de complexité en temps et en mémoire), d'autres considération, par exemples combinatoires, sont à privilégier.

4.4 Implémentation pratique de l'attaque PDRC

Considérons un (m, n)-système de chiffrement par blocs. Le problème principal est de trouver une propriété $\mathcal{I}_{\mathcal{K}}$ adéquate vérifiée avec un biais suffisant pour un sous-ensemble $\mathcal{C}_i \subset \mathcal{C}$ particulier (lié, par exemple à un codage particulier dans une langue ou un groupe linguistique donnés). En première approche, l'idée est de considérer des approximations linéaires, comme dans la cryptanalyse linéaire. Cependant, nos travaux actuels considèrent des propriétés plus complexes comme des variétés algébriques ou des propriétés combinatoires particulières. Ces approximations sont de la forme

$$< P, u > \oplus < C, w > \overset{q}{\cong} < K, v >$$

où u, v et w sont des masques permettant de sélectionner certains bits. Si nous parvenons à trouver un sous-ensemble de blocs de clair $\mathcal{C}_i$ pour lequel il existe $v', w' \in \mathbb{F}_2^n \times \mathbb{F}_2^m$ tel que

$$< C, w' > \overset{q'}{\cong} < K, v' > \tag{4.4}$$

avec $q' \neq \frac{1}{2}$ alors nous avons trouvé une propriété $\mathcal{I}_{\mathcal{K}}$ permettant de mettre en œuvre la cryptanalyse clair-dépendante par codes à répétition.

Il est important d'expliquer comment les équations ayant la forme de l'équations (4.4) peuvent fonctionner. Sur l'ensemble de l'expace des blocs de clair, ces équations sont normalement – si le système est bien conçu – des permutations et à ce titre, elles sont vérifiées avec une probabilité exacte égale à $\frac{1}{2}$ (en effet ce sont des équations linéaires). Cette constatation est valide pour n'importe quelle clef et donc pour n'importe quelle permutation.

A un niveau local – si l'on considère les blocs de chiffrés produits à partir de sous-ensembles de blocs de clair qui représentent des variétés algébriques ou des sous-ensembles particuliers par exemple – cette équation généralement n'est plus vérifiée avec une probabilité exacte égale à $\frac{1}{2}$. Cela peut être expliqué de la manière suivante. Tout (m,n)-système par bloc peut être décrit par une fonction booléenne f_j sur $\mathbb{F}_2^{m+n}$, relativement à chacun des bits $j, 0 \leq j < m$, de bloc de chiffré (voir le chapitre 3). Considérons à présent une partition $(\mathcal{C}_i)_{1\leq i\leq 2^k}$ de l'espace des blocs de texte clair [2] $\mathbb{F}_2^m$. Sans perte de généralité, nous supposerons que tout sous-ensemble $\mathcal{C}_i$ contient exactement 2^{n-k} éléments. Sur l'ensemble tout entier nous avons

$$P[< K, v >= f_j(K)] = \frac{1}{2}$$

et ce pour tout $v \in \mathbb{F}_2^n$ et tout j. Mais comme nous avons également

$$P[< K, v >= f_j(K)] = \sum_i \frac{1}{2^k} \cdot P[f_{j,\mathcal{C}_i}(K) =< K, v >]$$

où $f_{j,\mathcal{C}_i}$ désigne la restriction de f_j à $\mathcal{C}_i$, nous pouvons supposer comme très probable que pour quelques sous-ensembles $\mathcal{C}_i$, sinon tous, nous ayons $P[f_{j,\mathcal{C}_i}(K) =< K, v >] \neq \frac{1}{2}$. Ce fait a été implicitement reconnu par les concepteurs du système AES [AES02, Chapr. 7, §2 page 124].

Nous illustrerons cet effet de biais local, avec la permutation suivante (cas d'école).

Exemple 6 *Soit f la permutation sur $\mathbb{F}_2^8$ donnée par la table suivante :*

2. La notion de « texte clair » est à considérer d'une manière très générale. Le terme de clair désigne l'information à chiffrer sans préjuger de sa nature : langage compréhensible, information compressée...

```
(215, 100, 200, 204, 233, 050, 085, 196,
 071, 141, 122, 160, 093, 131, 243, 234,
 162, 183, 036, 155, 004, 062, 035, 205,
 040, 102, 033, 027, 255, 055, 214, 156,
 075, 163, 134, 126, 249, 074, 197, 228,
 072, 090, 206, 235, 017, 022, 049, 169,
 227, 089, 016, 005, 117, 060, 248, 230,
 217, 068, 138, 096, 194, 170, 136, 010,
 112, 238, 184, 189, 176, 042, 225, 212,
 084, 058, 175, 244, 150, 168, 219, 236,
 101, 208, 123, 037, 164, 110, 158, 201,
 078, 114, 057, 048, 070, 142, 106, 043,
 232, 026, 032, 252, 239, 098, 191, 094,
 059, 149, 039, 187, 203, 190, 019, 013,
 133, 045, 061, 247, 023, 034, 020, 052,
 118, 209, 146, 193, 222, 018, 001, 152,
 046, 041, 091, 148, 115, 025, 135, 077,
 254, 147, 224, 161, 009, 213, 223, 250,
 231, 251, 127, 166, 063, 179, 081, 130,
 139, 028, 120, 151, 241, 086, 111, 000,
 088, 153, 172, 182, 159, 105, 178, 047,
 051, 167, 065, 066, 092, 073, 198, 211,
 245, 195, 031, 220, 140, 076, 221, 186,
 154, 185, 056, 083, 038, 165, 109, 067,
 124, 226, 132, 053, 229, 029, 012, 181,
 121, 024, 207, 199, 177, 113, 030, 080,
 003, 097, 188, 079, 216, 173, 008, 145,
 087, 128, 180, 237, 240, 137, 125, 104,
 015, 242, 119, 246, 103, 143, 095, 144,
 002, 044, 069, 157, 192, 174, 014, 054,
 218, 082, 064, 210, 011, 006, 129, 021,
 116, 171, 099, 202, 007, 107, 253, 108)
```

et notons les bits d'entrée comme suit $x = (x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0)$. *Les bits de sortie seront notés* $f(x) = y = (y_7, y_6, y_5, y_4, y_3, y_2, y_1, y_0)$. *Considérons maintenant la restriction de* f *au sous-ensemble défini par* $(x_7, x_6, x_5, x_4) = (1, 1, 1, 0)$.

Pour ce sous-ensemble, nous avons

$$P[x_0 \oplus x_3 = y_0] = \frac{5}{16} \neq \frac{1}{2}$$

et

$$P[x_0 \oplus x_3 = y_0 \oplus y_1] = \frac{5}{8} \neq \frac{1}{2}.$$

Du point de vue du cryptographe, ces considérations suggèrent fortement que la notion de « *sécurité prouvable* » doit être fortement relativisée. Chaîner des primitives cryptographiques (que ce soit dans les réseaux de substitution-permutations ou dans des schémas de Feistel) ayant isolément des propriétés cryptographiques optimales ne garantit nullement que des phénomènes plus ou moins locaux conduisant à des dégénérescences insoupçonnées et incontrolables n'existent pas. La complexité combinatoire des ensembles impliqués au niveau global est telle qu'il est impossible de le déterminer.

4.5 Généralisation : la cryptanalyse par code à répétition

L'attaque clair-dépendant précédente peut être généralisée en considérant du clair connu. Ce scénario est communément considéré dans la plupart des attaques publiées. Dans cette situation, une propriété $\mathcal{I}_{\mathcal{K}}$ donnée est évaluée en faisant intervenir cette fois le texte clair et le texte chiffré. Considérons, une fois encore, le cas particulier de la cryptanalyse linéaire. Les équations probabilistes de la forme :

$$< P, u > \oplus < C, w > \overset{q}{\cong} < K, v >$$

peuvent être réécrites comme suit :

$$< K, v > \overset{q}{\cong} < P, u > \oplus < C, w > .$$

En conséquence, nous pouvons utiliser, une fois encore, les codes à répétition pour retrouver les bit d'information $< K, v >$. Comme la connaissance du texte clair est requise pour calculer le membre droit de

l'équation, l'attaque PDRC devient une cryptanalyse par codes à répétition (RCC). L'algorithme A.2 reste pour l'essentiel, le même. Notons que dans le cadre RCC, aucune recherche exhaustive sur une partie de la clef (comme dans la cryptanalyse linéaire de Matsui [Matsui94]) n'est nécessaire.

La cryptanalyse RC améliore-t-elle la cryptanalyse linéaire ? Dans la cryptanalyse de Matsui [Matsui94, §6], deux équations sur 16 tours ont été établies pour le DES, en utilisant la meilleure approximation connue sur 14 tours. Chacune d'entre elles est vérifiée avec une probabilité de $0.5 - 1.19 \times 2^{-21}$ et 26 bits de clef sont ainsi retrouvés (en considérant à la fois les équations pour le chiffrement et pour le déchiffrement). L'attaque requiert de disposer de 2^{43} couples de blocs clair/chiffré. La probabilité de succès est alors en moyenne de 85 % [Matsui94b]. Cette probabilité tombe à 10 % si l'on ne dispose que de 2^{38} couples. Les 30 bits restant – la clef étant de 56 bits – sont retrouvés par une recherche exhaustive.

Afin de comparer l'attaque RC avec la cryptanalyse linéaire, nous allons considérer les deux meilleures approximations des 16 tours du DES (en chiffrement et en déchiffrement), puisque nous ne pouvons effectuer dans RC de recherche exhaustive sur la clef, comme dans l'attaque de Matsui. L'équation de chiffrement [Matsui94] est donnée par

$$\begin{aligned} & P_H[7,18,24] \oplus P_L[12,16] \\ & \oplus C_H[15] \oplus C_L[7,18,24,29,27,28,30,31] \\ \stackrel{q}{\cong}\ & K_1[19,23] \oplus K_3[22] \oplus K_4[44] \oplus K_5[22] \oplus K_7[22] \oplus K_8[44] \\ & \oplus K_9[22] \oplus K_{11}[22] \oplus K_{12}[44] \oplus K_{13}[22] \oplus K_{15}[22] \\ & \oplus K_{16}[42,43,45,46]. \end{aligned}$$

Cette équation est vérifiée avec une probabilité égale à $p = 0.5 - 1.49 \times 2^{-24}$. Pour l'attaque RC, l'équation est réécrite comme suit

$$\begin{aligned} & 1 \oplus P_H[7,18,24] \oplus P_L[12,16] \oplus C_H[15] \\ & \oplus C_L[7,18,24,29,27,28,30,31] \\ \stackrel{q}{\cong}\ & K_1[19,23] \oplus K_3[22] \oplus K_4[44] \oplus K_5[22] \oplus K_7[22] \oplus K_8[44] \\ & \oplus K_9[22] \oplus K_{11}[22] \oplus K_{12}[44] \oplus K_{13}[22] \\ & \oplus K_{15}[22] \oplus K_{16}[42,43,45,46], \end{aligned}$$

Elle est maintenant vérifiée avec une probabilité égale à $p = 0.5 + 1.49 \times 2^{-24}$. L'algorithm A.1 donne les résultats suivants (tableau 4.4) et peuvent être comparés à ceux de Matsui pour la cryptanalyse linéaire. Deux bits d'informations concernant la clef sont retrouvés mais

Attaque	Couples clair/chiffré	Proba. de succès
Matsui [Matsui94b]	2^{43}	85 %
Matsui [Matsui94]	2^{49}	78.5 %
Attaque RC	2^{45}	81 %

TABLE 4.4 – Comparaison de la cryptanalyse linéaire de Matsui et de la cryptanalyse RC pour le DES

sans essai exhaustif contraitement à la cryptanalyse de Matsui. et en considérant une probabilité plus faible que pour la cryptanalyse linéaire ($p = 0.5 + 1.49 \times 2^{-24}$) au lieu de $p = 0.5 - 1.19 \times 2^{-21}$). Si nous disposions de plus d'équations du même type (au total 25), l'algorithme A.1 retrouverait la totalité de la clef avec la même complexité (autrement dit 2^{45} lectures de couples clairs/crypto).

Les quelques remarques suivantes permettent de constater que, pour des mêmes biais, l'attaque RC est plus efficace que la cryptanalyse linéaire :

- La probabilité de succès de l'attaque RC est évaluée en utilisant la formule $P_{succ} = 1 - P_{err}$ où P_{err} est calculée au moyen de la formule (4.2). Mais il est important de se rappeler qu'elle concerne l'algorithme A.1 lequel a été montré moins efficace que l'algorithme A.2. La probabilité de succès donnée pour la cryptanalyse RC est donc une borne inférieure.
- La cryptanalyse linéaire ne peut efficacement manipuler que des approximations linéaires, pour lesquelles, en outre, le poids du masque v sur la clef doit être de faible poids. En effet, dans la complexité générale de cette cryptanalyse intervient une recherche exhaustive en $\mathcal{O}(2^{wt(v)})$. La cryptanalyse RC, quant à elle peut considérer n'importe quel masque v sur la clef, puisqu'aucune étape de recherche exhaustive partielle sur la clef n'est nécessaire.
- La cryptanalyse RC fonctionne également sur des approximations de degré strictement supérieur à 1 (fonctions non linéaires).

Les bits d'informations ainsi obtenus peuvent ensuite servir à une résolution de systèmes d'équations par bases de Gröbner [FAUG03, ArsFaug04].

4.6 Conclusion

Les attaques PDRC et RC décrivent un nouveau modèle de cryptanalyse utilisant les codes à répétition. Toutes exploitent directement le fait que de bloc en bloc, la même clef secrète est réutilisée. Cette réutilisation semble donc une faiblesse potentielle, du moins à partir d'un certain nombre de blocs. Déterminer à partir de combien de blocs il y a danger n'est pas facile car cela dépend de plusieurs paramètres :

- en premier lieu du système considéré, des primitives utilisées, du nombre de tours...et des effets plus ou moins importants, s'ils existent des dégérescences au niveau global. Or les évaluer est actuellement hors de portée.
- de l'existence de propriétés algébriques ou combinatoires réalisant des biais statistiques par rapport à la loi uniforme. Or là également, si la théorie assure qu'elle existent (théorème de Parseval pour les seules approximations linéaires), les identifier constitue un problème ouvert.

L'étude de l'ensemble des m fonctions booléennes de sortie d'un systèmes de chiffrement par blocs, sous l'angle combinatoire est une voie à explorer. L'obtention de propriétés cryptographiques optimales pour ces fonctions booléennes au niveau des primitives nécessitent de faire certains compromis [FiFont98]. Quels sont les effets de ces compromis sur la structure générale d'un système par blocs ? Actuellement, aucune étude n'a traité ce problème ni n'a pu garantir la totale absence de propriétés indésirables dégradant la sécurité générale des systèmes de chiffrement par blocs.

La direction que je considére concerne l'étude des fonctions booléennes sous l'angle combinatoire. Les quelques travaux existants, notamment ceux de Dillon [DIL72, DIL74] sur les ensembles à différences et les fonctions courbes, ont eu pour objectifs d'identifier des structures combinatoires permettant de construire des fonctions booléennes – utilisées comme briques de base dans des systèmes de chiffrement complexes – possédant des propriétés cryptographiques fortes. Mon objectif

est diamétralement opposé : pour des fonctions booléennes fixées – les fonctions de sortie d'un système de chiffrement par exemple – je souhaite trouver des structures combinatoires révélant des faiblesses cryptographiques. Ces structures peuvent concerner les différentes représentations possibles d'une fonction booléennes : table de vérité, forme algébrique normale, forme normal disjunstive... Une fois ces structures identifiées, une seconde étape s'occupe de les traduire en biais statistiques utilisables dans une cryptanalyse classique (cryptanalyse par codes à répétition, attaques par corrélation...). Malgré quelques succès très limités mais néanmoins prometteurs, le travail n'en est qu'à ses débuts.

En attendant, l'utilisation des systèmes de chiffrement par blocs devrait se faire selon le principe de précaution et être réservé à la protection d'informations peu sensibles. C'est la voie que le gouvernement fédéral américain semble adopter (voir [DES77, page 2 §6] et [FIPS197, page 1 §6].

Chapitre 5

Preuve de cryptanalyse à apport de connaissance nulle

5.1 Introduction

La fonctionnalité de chiffrement est à la base de la plupart des mécanismes ou protocoles de sécurité. Celui qui parvient à la contourner, s'il parvient à garder cette capacité secrète, non seulement accède aux informations traitées mais également aux informations traitantes (celles assurant la sécurité du système). Le simple concept du mot de passe permet d'illustrer de manière pertinente cette constatation. Mais des protocoles plus complexes (Wep, WPA, Bluetooth, GSM...) dépendent encore plus lourdement du chiffrement.

Dans ce chapitre, je me concentrerai sur le protocole Bluetooth et son algorithme de chiffrement E0. Ce protocole est utilisé pour la communication sans fil. De très nombreux matériels : ordinateurs ou téléphones portables, assistants digitaux personnels, imprimantes, ordinateurs de bord de voitures... La norme Bluetooth fonctionne avec une sécurité multi-niveaux laquelle est directement et intimement assurée par des mécanismes cryptographiques d'authentification et de chiffrement. Le cœur dédié au chiffrement est architecturé autour de l'algorithme E0 dont l'entropie de la clef est de 128 bits. La taille de la clef permet donc d'interdire, en pratique, toute attaque exhaustive. Il faudrait effectivement plusieurs milliards de siècles pour essayer toutes les clefs. Mais un bon cryptologue sait qu'il ne s'agit là que d'une condition nécessaire à

la sécurité mais nullement suffisante.

A ce jour, cependant aucune attaque publiée n'est sérieusement parvenue à remettre en question la sécurité de l'algorithme E0. Les quelques attaques connues n'ont qu'un intérêt théorique [AK03, NC03, BLEI01, FLUH01, FLUH02, GBM02, KRAUSE02, LW05, LU04, LU05]. À moins d'accepter des conditions irréalistes, E0 reste utilisé dans la norme Bluetooth et la sécurité offerte par le protocole peut être considérée comme élevée.

Cependant, un problème se pose concernant le caractère judicieux ou non de publier une attaque qui serait, elle, opérationnelle et remettrait en cause très sérieusement la sécurité du chiffrement Bluetooth. Dans le cas de services spécialisés, la question ne se pose pas. Il s'agit de profiter le plus longtemps de la faiblesse non soupçonnée par celui qui utilise le système. En revanche, quand ce protocole est très largement utilisé par des entreprises, des administrations ou des particuliers, peut-on vraiment taire une telle possibilité ? Un bon juriste, peut être un peu spécieux aidé par un journaliste peu scrupuleux, finalement ne pourrait-il pas invoquer la « mise en danger informatique » d'autrui voire tout simplement la mise en danger d'autrui quand cette faiblesse concerne des équipement médicaux vitaux, par exemple ?

Alors qu'il est incontestablement compréhensible d'alerter les ingénieurs, les décideurs et plus largement les professionnels de la sécurité, il existe certainement d'autres raisons de considérer cela comme condamnable et finalement de ne rien dire. C'est Corneille au pays de la sécurité informatique. Ces raisons de ne rien dire peuvent être motivées :

– par le fait que révéler toutes données techniques reproductibles et à portée opérationnelle, donne le champ libre aux attaquants. Dans le cas de système de chiffrement embarqués et implémentés « en dur » (WEP, Bluetooth, GSM...), changer le cœur de chiffrement se révèle extrêmement coûteux voire rédhibitoire pour la survie même de la norme et des entreprises qui l'implémentent dans leurs appareils. En outre, cela nécessiterait trop de temps. Des mois voire des années sont nécessaires pour changer tous ces appareils. Les industriels chercheront à rentabiliser leurs investissements avant de procéder à cette mutation. Les utilisateurs sont eux-mêmes réticents, pour des problèmes de coût, à changer d'appareil. Mais pendant cette longue phase de transition et

d'attentisme, un grand nombre d'attaques surviendrait ;
– le fait que révéler de telles informations techniques, permettant de mettre à mal la sécurité d'un système, est condamné depuis quelques années par de nombreux pays dont la France (*Loi pour la Confiance en l'Economie Numérique (LCEN)*[1]. De plus, de telles publications peuvent, par ailleurs, être poursuivie pour atteinte au copyright. Le meilleur exemple est probablement la loi américaine du *Digital Millenium Copyright Act* [DMCA98]. Plus récemment des lois comme la DAVSI, en France, vont dans ce sens.

Cependant, retenir de telles connaissances en les communiquant aux seuls développeurs n'est par une solution admissible. Elle est susceptible d'inciter ces derniers à ne rien faire pour les raisons commerciales précédemment évoquées.

La question est alors la suivante : comment concilier les deux aspects, d'une manière irrévocable et incontestable, sans révéler aucune information techniques utilisables par un attaquant, et tout en prouvant que l'on dispose effectivement de connaissances permettant de cryptanalyser de manière opérationnelle un système de chiffrement donné ?

Dans ce chapitre, je vais répondre à cette question et montrer comment résoudre ce problème en l'illustrant par le cas du chiffrement Bluetooth (E0). Des résultats récents et significatifs obtenus récemment au laboratoire de virologie et de cryptologie [Fil06d, Fil07] ont permis d'identifier des faiblesses cryptographiques sérieuses, de nature combinatoire, dans E0. Elles ont permis de construire une cryptanalyse opérationnelle de cet algorithme. Je vais montrer comment prouver la réalité de cette attaque, vérifiable en temps polynomial par quiconque, sans cependant révéler une quelconque information sur la manière dont elle a été menée ni sur les faiblesses qui l'ont rendue possible. J'ai appelé ce type de preuve, *preuve de cryptanalyse de type Zero-knowledge*.

1. Selon l'article 323-3-1 du code pénal amendé par la LCEN de juin 2004, condamne cette infraction, à des peines allant jusqu'à trois ans de prison et 45.000 euros d'amende [LCEN]. D'autres articles de loi, tout aussi répressifs concernant spécifiquement la cryptologie peuvent s'ajouter.

5.2 Concepts et notations

Tout d'abord, il convient de rappeler en quoi consiste le *Zero-knowledge*. Il s'agit d'une propriété caractérisant un protocole de preuve interactive. Dans ce protocole, un *prouveur* doit convaincre un *vérifieur* de la validité d'une affirmation. Alors qu'aucune restriction n'est imposée au prouveur, en revanche, le vérifieur ne peut utiliser que des algorithmes polynomiaux (éventuellement probabilistes). Par Zero-knowledge [2], il faut comprendre que le vérifieur est finalement convaincu sans qu'il soit nécessaire pour le prouveur de donner des informations quant à la manière dont a été établie l'affirmation. Ce concept a été introduit pour la première fois par Goldwasser et al. [GMR89].

Rappelons maintenant succinctement la définition d'un système de chiffrement par flot. Il s'agit d'un système de chiffrement symétrique – en d'autres termes, la même clef, déployée préalablement à la transmission aupr'es de l'émetteur et du destinataire, est employée pour le chiffrement et le déchiffrement – qui fonctionne de la manière suivante : le texte clair, envisagé comme une suite de bits $m_0, m_1, \ldots, m_i, \ldots$ est chiffrée en un cryptogramme $c_0, c_1, \ldots, c_i, \ldots$ au moyen d'une séquence pseudo-aléatoire $s_0, s_1, \ldots, s_i, \ldots$ générée par le système. L'opération de chiffrement/déchiffrement la plus courante est la fonction XOR :

$$c_i = m_i \oplus s_i.$$

La suite chiffrante est produite en fait par un automate à état fini dont l'état initial est la clef secrète partagée par l'émetteur et le destinataire.

Comme pour tout autre système de chiffrement, la première attaque pouvant être menée consiste à faire une recherche exhaustive des clefs jusqu'à trouver la bonne : celle qui à partir du texte chiffrée produit un clair intelligible. Dans le contexte du présent chapitre, nous allons tenter de retrouver la clef quand cette dernière réalise une suite binaire de sortie réalisant des propriétés remarquables, fixées à l'avance. L'intérêt de cette approche est d'une part qu'il permet de se mettre dans des conditions similaires à celles d'une attaque réelle : le choix des propriétés doit représenter la même contrainte, ou du moins des contraintes similaires, que le texte clair choisi par celui que l'on attaque. D'autre

2. L'usage français impose de parler d'apport de connaissance nul. Pour ne pas alourdir le textte, je conserverai le terme anglo-saxon, plus pratique.

part, ces propriétés ne doivent être réalisables, soit par une attaque déja publiée soit par une recherche exhaustive ou une recherche aléatoire sur les clefs.

Concernant ce dernier aspect, définissons rigoureusement la complexité liée à ce type de recherche sur les clefs (exhaustive ou aléatoire). Considérons une clef secrète K de n bits et considérons une propriété mathématique $\mathcal{P}$ réalisée par la suite chiffrante correspondante, elle-même de longueur n. Retrouver une clef produisant cette suite chiffrante particulière requiert $n^{\mathcal{O}(1)}.\mathcal{O}(2^{n-m})$ opérations où 2^m représente le nombre total de clefs pour lesquelles la propriété est également réalisée. Notons que dans le cas d'une recherche exhaustive classique ($m = 0$), il n'existe qu'une seule clef produisant la suite chiffrante fixée.

Une recherche *aléatoire* consiste, elle, à essayer au hasard un grand nombre de clefs jusqu'à en obtenir une qui produit effectivement la propriété $\mathcal{P}$ considérée.

Enfin, le *poids de Hamming* d'une suite $(s_t)_{0\leq t\leq n}$, noté $wt((s_t)_{0\leq t\leq n})$ est le nombre de bits non nuls de cette suite :

$$wt((s_t)_{0\leq t\leq n}) = \{0 \leq t \leq n | s_t = 1\}.$$

5.3 Le chiffrement Bluetooth

Les mécanismes de sécurité du protocole de communication Bluetooth sont présentés dans la partie H du volume 2 des spécifications officielles [BlueSpec]. Selon le standard Bluetooth, la couche de sécurité est située au niveau hardware (couche dite *Baseband*), lequel est contrôlé par les couches logicielles supérieures (niveau applications et utilisateurs). Ces mécanismes de sécurité comprennent la génération et la gestion des clefs, l'authentification des utilisateurs et du matériel et le chiffrement des données échangées à l'aide de ce protocole. L'algorithme de chiffrement est l'algorithme par flot E0.

Lors de chaque communication entre deux appareils utilisant ce protocole, une première étape d'authentification mutuelle et d'échange de clefs intervient. Son objectif est de partager une clef secrète (la clef de liaison ou *link key*), elle-même utilisée ensuite pour générer la clef de chiffrement proprement dite K_C. Cette dernière est construite à partir

de la clef de liaison en cours, un *offset* de chiffrement (COF) jouant en quelque sorte le rôle de marquant cryptographique, et un nombre aléatoire publique (EN_RAND).

Pour chiffrer un paquet (trame de bits), la clef secrète K_C est modifiée pour produire une autre clef K'_C. Cette dernière est alors utilisée de manière linéaire avec les valeurs publiques, l'adresse MAC (adresse Bluetooth) et une valeur d'horloge différente de paquet en paquet, pour finalement constituer l'état initial d'un système de chiffrement à double-niveau décrit sur le schéma 5.1 et dont le coeur est l'algotithme E0. Ce dernier génère une suite chiffrante binaire K_{cipher}, additionnée modulo 2 bit à bit au texte clair. Le déchiffrement est réalisé de la même manière, la fonction XOR étant involutive.

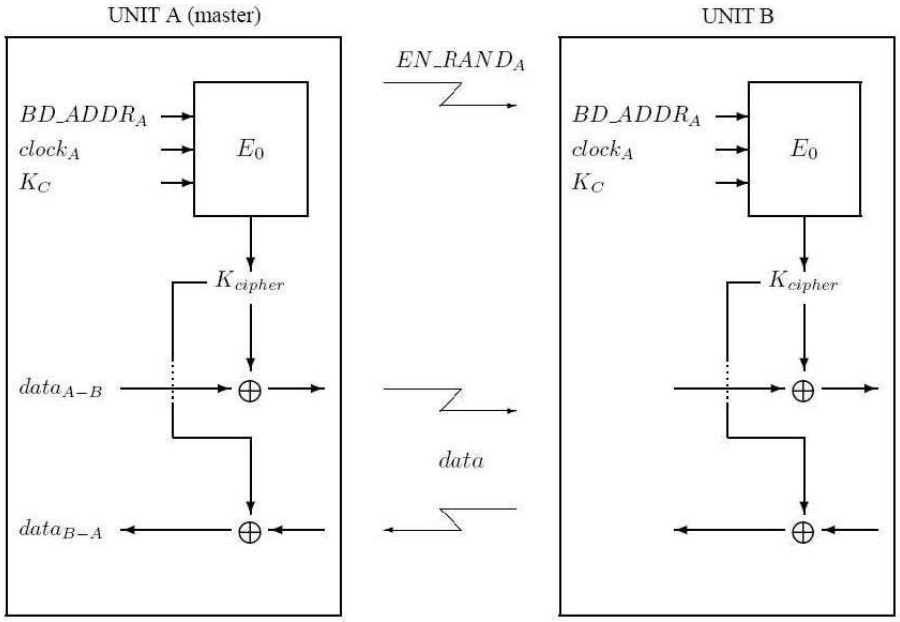

FIGURE 5.1 – Description fonctionnelle du chiffrement Bluetooth

Toute attaque opérationnelle contre le cœur E0 aura pour conséquence de remettre en question toute la sécurité du protocole Bluetooth. Outre le fait qu'il serait alors possible de manipuler les données chiffrées échangées entre les appareils communiquant (insertion de codes malveillants par exemple), couplée avec d'autres attaques opérationnelles

contre la partie authentification et négociation des clefs du protocole, récemment publiées [SW05], la capacité de cryptanalyse (retrouver une clef secrète de chiffrement) est susceptible de mettre définitivement à mal la sécurité générale du protocole Bluetooth.

5.3.1 L'algorithme de chiffrement E0

Pour mieux comprendre la difficulté de cryptanalyse du coeur de la logique de chiffrement de E0, nous allons présenter succintement cet algorithme.

L'algorithme E0 met en œuvre des registres à décalage à rebouclage linéaire (LFSR), primitives classiques pour ce type d'algorithme. Leur sortie respective est combinée au moyen d'une machine (automate) à états finis, laquelle n'est en fait qu'une fonction booléenne vectorielle définie sur $\mathbb{F}_2^{16}$. Cette fonction est appelée fonction de sommation additive. A valeur dans $\mathbb{F}_2$, elle produit la suite de sortie, utilisée comme suite chiffrante et lors du processus d'initialisation, comme suite aléatoire d'initialisation.

L'algorithme E0 fonctionne avec une clef K_C, une adresse Bluetooth de 48 bits, une suite d'horloge CLK_{26-1} et une valeur (graine) aléatoire RAND de 128 bits. La figure 5.2 présente le processus de mise à la clef de l'algorithme de chiffrement. Quatre registres sont utilisés $(LFSR_1, \ldots, LFSR_4)$ de longueur respectivement égale à $L_1 = 25, L_2 = 31, L_3 = 33$ et $L_4 = 39$ dont le polynome de rétro-action est spécifié dans le tableau 5.1. La longueur totale des registres vaut 128, ce qui correspond donc à la taille de la clef secrète qui initialise ces registres à $t = 0$.

Ces polynomes primitifs ont été choisis afin de réaliser le meilleur compromis entre les contraintes industrielles d'implémentation d'une part et les besoins de qualité statistique requis pour une application cryptographique de haut niveau de sécurité. Notons x_t^i le t-ième symbole produit par le registre $LFSR_i$. La valeur y_t est calculée à partir du quadruplet $x_t^1, x_t^2, x_t^3, x_t^4$ à l'aide de l'équation suivante :

$$y_t = \sum_{i=1}^{4} x_t^i,$$

où la somme est calculée sur les entiers. Ainsi, y_t peut prendre les valeurs

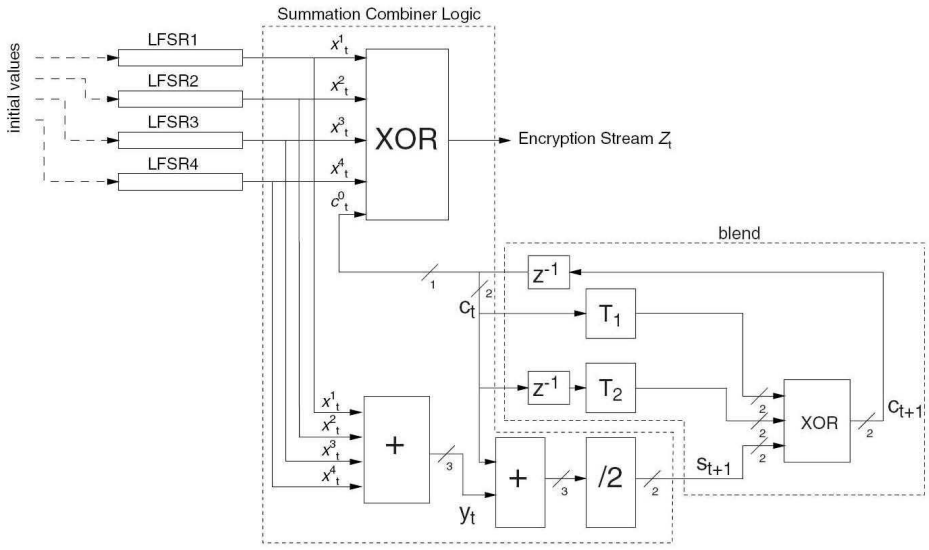

FIGURE 5.2 – Description fonctionnelle de la procédure de chiffrement E0

0, 1, 2, 3 ou 4. La sortie de la fonction de combinaison est obtenue à l'aide des équations suivantes :

$$z_t = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus c_t^0 \qquad \in \{0,1\},$$

$$s_{t+1} = (s_{t+1}^1, s_{t+1}^0) = \lceil \frac{y_t + c_t}{2} \rceil \qquad \in \{0,1,2,3\},$$

$$c_{t+1} = (c_{t+1}^1, c_{t+1}^0) = s_{t+1} \oplus T_1[c_t] \oplus T_2[c_{t-1}],$$

où $T_1[.]$ et $T_2[.]$ représentent deux bijections linéaires sur $GF(4)$ et données dans le tableau 5.2. L'algorithme E0 doit initialiser les quatres registres avec la clef secrète K'_C et quatre bits supplémentaires, lesquels sont généralement publics, initialisent les variables c_{-1} et c_0. La clef K'_C et ces quatre bits sont produits lors d'une phase d'initialisation impliquant E0, la clef secrète K_C, une adresse Bluetooth de 48 bits, la suite d'horloge CLK_{26-1} et la valeur RAND de 128 bits.

i	L_i	Polynomes de rétro-action
1	25	$x^{25} \oplus x^{20} \oplus x^{12} \oplus x^8 \oplus 1$
2	31	$x^{31} \oplus x^{24} \oplus x^{16} \oplus x^{12} \oplus 1$
3	33	$x^{33} \oplus x^{28} \oplus x^{24} \oplus x^4 \oplus 1$
4	39	$x^{39} \oplus x^{36} \oplus x^{28} \oplus x^4 \oplus 1$

TABLE 5.1 – Les polynomes de rétro-action de E0

x	$T_1[x]$	$T_2[x]$
00	00	00
01	01	11
10	10	01
11	11	10

TABLE 5.2 – Bijection linéaire de E0

5.3.2 Etat de l'art de la cryptanalyse de E0

Le système de chiffrement par flot E0 est à ce jour considéré comme un système sûr offrant un niveau de sécurité élevé. La meilleure preuve est qu'il est toujours employé et qu'aucune mise en garde n'a été publié concernant une faiblesse quelconque. De plus, selon les critères du NIST (suite statistique STS [NIST_STS]) référence en la matière, E0 présente des propriétés statistiques excellentes compatibles avec un niveau de sécurité cryptologique élevé. Aucun biais d'aucune sorte n'a été identifié.

Plusieurs cryptanalyses de E0 ont été publiées. Elles peuvent être reparties en deux catégories, selon la quantité de bits de sortie requis pour le succès de ces attaques :

– **les attaques à suite longue (*Long keystream attacks*)**.- Ces attaques envisagent le chiffrement Bluetooth (algorithme E0) en dehors de son contexte réel d'utilisation. Elles requièrent, pour fonctionner, des tailles de suites de sortie tellement importantes que ces attaques ne remettent nullement en cause la sécurité de E0. Pour la plupart d'entre elles, il s'agit d'attaques par corrélation simple ou par corrélation rapide. Autrement dit, elles ex-

ploitent l'existence de corrélations entre les sorties des différents registres de E0 et la suite de sortie produite par l'algorithme. Deux attaques [AK03, NC03] considérent des techniques de linéarisation d'équations non linéaires dont les variables sont consituées des bits de la clef secrète. Le tableau 5.3 résume les quantités de bits de sortie nécessaire pour que les attaques soient couronnées de succès (colonne Data) et les complexités des attaques (étape de précalcul, complexité temps et mémoire).

Attaques	Data	Précalcul	Compl. (en temps)	Compl. mémoire
Fluhrer - Lucks [FLUH01]	2^{43}	-	2^{73}	2^{51}
Fluhrer [FLUH02]	$2^{12.4}$	2^{80}	2^{65}	2^{80}
Golic & al. [GBM02]	2^{17}	2^{80}	2^{70}	2^{80}
Armknecht - Krause [AK03]	2^{24}	-	2^{68}	2^{48}
Courtois [NC03]	2^{24}	-	2^{49}	2^{37}
Lu & Vaudenay [LU04]	$2^{39.6}$	-	2^{40}	2^{35}
Lu & al. [LU05]	$2^{28.4}$	2^{38}	2^{38}	2^{33}

TABLE 5.3 – Complexités comparées des attaques à suite longue sur E0

– **les attaques à suite courte (*Short keystream attacks*)**.- Ces attaques considèrent des suites de sortie courtes (de l'ordre de 128 bits). Malgré leur complexité très élevée (qui rendent leur intérêt uniquement théorique), elles constituent une approche plus réaliste que les précédentes. Dans cette catégorie, il existe peu de cryptanalyses publiées : utilisation de diagrammes de décision binaires [KRAUSE02], méthode de *backtracking* [LW05]... Toutes ont pour intérêt essentiel d'avoir ouvert de nouvelles pistes de réflexion et d'avoir renouvelé les outils et les méthodes en cryptanalyse. Le tableau 5.4 présente et compare les complexités de ces quelques attaques à suite courte.

Attaques	Bits de sortie connus	Complexité attaque
Bleichenbacher [BLEI01]	128	2^{100}
Krause [KRAUSE02]	128	2^{81}
Levy - Wool [LW05]	128	2^{86}

TABLE 5.4 – Complexités comparées des attaques à suite courte sur E0

5.4 Preuve de cryptanalyse de type Zero-knowledge

Les attaques connues à ce jour et présentées dans la section précédentes requièrent toutes des conditions irréalistes pour fonctionner en pratique : quantité extravagante de bits de sortie et/ou complexité en temps voire en mémoire rédhibitoirement élevées. La conclusion est qu'à ce jour, l'algorithme E0 peut être considéré comme sûr.

Considérons à présent des suites de sortie de longueur n (on suppose donc de connaitre, d'un point de vue cryptanalytique, une quantité égale de texte clair, cadre de travail de toutes les cryptanalyses connues pour E0). L'idée centrale de la *preuve de cryptanalyse de type Zero-knowledge* est de considérer des propriétés mathématiques concernant la suite de sortie qui ne peuvent être réalisées calculatoirement, à moins de connaitre effectivement une ou plusieurs faiblesses conceptuelles dans l'algorithme de chiffrement. La définition suivante précise les choses.

Définition 9 *(Preuve de cryptanalyse de type Zero-knowledge) Soit un cryptosystème S_K et une propriété mathématique $\mathcal{P}$ concernant la suite de sortie σ_K^n, produite par S_K et de longueur n. Aucune méthode connue autre que la recherche exhaustive ou la recherche aléatoire ne permet de réaliser la propriété $\mathcal{P}$ pour la suite σ_K^n. Alors, une* preuve de cryptanalyse de type Zero-knowledge *du système S_K consiste à exhiber une ou plusieurs clefs secrète $K_1, K_2, \ldots, ...K_m$ telles que les suites de sorties $(\sigma_{K_i}^n)_{1 \leq i \leq m}$ réalisent la propriété $\mathcal{P}$ et telles que la vérification de cette propriété se fasse en temps polynomial. De plus, la propriété $\mathcal{P}$ ne doit fournir aucune information sur la manière dont elle a été obtenue.*

Le protocole proposé dans la définition 9 n'est pas véritablement Zero-knowledge (d'où l'usage de l'expression « *de type Zero-knowledge* ») et

ce pour les raisons suivantes :

1. un article n'est pas un moyen interactif ;
2. l'auteur de la cryptanalyse joue à la fois le rôle du prouveur et répond à des questions qui ne lui ont pas été posées par le vérifieur, à savoir le lecteur de l'article.

Un autre point qu'il est essentiel de considérer tient au fait que le lecteur/vérifieur peut opposer à l'auteur/prouveur le fait qu'il a essayé un certain nombre de clefs aléatoires, produit des suites de sortie et qu'il a finalement conservé celles présentant des propriétés intéressantes, prétendant les avoir choisies *a priori*. En d'autres termes, l'auteur/prouveur peut tenter de tromper et manipuler le lecteur/vérifieur en utilisant une approche par recherche exhaustive des clefs réalisant une ou plusieurs propriétés mathématiques remarquables pour la suite de sortie. Aussi, la ou les propriétés mathématiques $\mathcal{P}$ doivent être soigneusement choisies de sorte que :

- la probabilité de les obtenir par recherche aléatoire ou exhaustive sur l'espace des clef doit être suffisamment faible pour interdire cette approche calculatoirement ; cet aspect sera traité dans les sections 5.4.1, 5.4.2 et 5.4.3 ;
- les attaques connues ne peuvent permettre de retrouver des clefs produisant la ou les propriétés considérées par l'auteur/prouveur.
- afin de convaincre réellement le lecteur/vérifieur, un grand nombre de clefs secrètes doit pouvoir être retrouvées et exhibées. Ainsi, il n'est pas possible de soupçonner que l'auteur/prouveur a eu « de la chance ».

Comme il n'existe aucune autre méthode que les recherches exhaustive et aléatoire pour produire les suites σ_K^n ayant la propriété $\mathcal{P}$ et que ces attaques sont irréalisables en pratique, quiconque parvient à produire de telles clefs doit obligatoirement connaître des faiblesses non publiées pour parvenir à un tel résultat. La probabilité de réaliser la propriété $\mathcal{P}$ par le biais d'une recherche exhaustive fournit directement une borne supérieure quant à la complexité de la cryptanalyse de type zero-knowledge utilisée par l'auteur/prouveur. Cette borne étant triviale, il n'est pas possible de déduire une quelconque information quant à la méthode utilisée.

Le dernier aspect sur lequel insister concerne le fait de d'accepter si la connaissance d'une faille conceptuelle dans l'algorithme de chiffre-

ment implique la possibilité de le cryptanalyser. L'approche académique traditionnelle considère généralement les modèles cryptanalytiques établis suivants :

- soit le cryptanalyste connaît une certaine quantité de bits de la suite de sortie et tente de retrouver la clef secrète ayant généré cette suite,
- soit le cryptanalyste tente de distinguer efficacement une suite de sortie produite par le système de chiffrement d'une suite aléatoire [3]

Dans notre contexte, le modèle considéré n'est pas si éloigné du premier cas, tant que les propriétés mathématiques considérés pour la suite de sortie ne sont pas reproductibles, si ce n'est que par une véritable technique de cryptanalyse (autrement dit différente de la recherche exhaustive par exemple) : nous fixons *a priori* certaines suites de sortie, réalisant une ou plusieurs propriétés $\mathcal{P}$ données et nous cherchons la clef secrète correspondante pour chacune d'entre elles. En d'autres termes :

- dans le cas classique, un cryptanalyste tente de retrouver $K = E_0^{-1}(\sigma_K)$ pour une suite de sortie fixée *a priori* σ_K ;
- dans notre cas, nous considérons un sous-ensemble $\mathcal{S}^{\mathcal{P}}$ de suites de sortie réalisant *a priori* une ou plusieurs propriétés données $\mathcal{P}$, et on souhaite retrouver $K_{\mathcal{S}^{\mathcal{P}}} = E_0^{-1}(\mathcal{S}^{\mathcal{P}})$.

Dans le reste de ce chapitre, nous considérerons des suites de sortie de longueur $n = 128$. Le choix de cette valeur est dicté par deux considérations :

- cette taille de suite est plus réaliste dans le cadre d'un usage opérationnel de l'algorithme E0 dans le protocole Bluetooth (voir la section 5.3) ;
- choisir un suite de sortie de taille très limitée renforce le niveau d'efficacité et la portée de l'attaque et de là, celle de la preuve de cryptanalyse de type Zero-knowledge.

Enfin, comme les suites de sortie de l'algorithme E0 présentent toutes les propriétés statistiques exigées par un système de chiffrement de haut niveau de sécurité, sous l'hypothèse que les clefs secrètes sont vues

3. Notons que disposer d'un distingueur efficace pour un système cryptologique ne signifie pas que l'on soit capable de cryptanalyser ce système. Le distingueur concerne un problème TRANSEC (*Transmission Security*) alors que la cryptanalyse concerne plutôt un problème COMSEC (*Communication Security*).

comme des variables aléatoires sur $\mathbb{F}_2^{128}$, toute suite de sortie de E0 σ_K^{128} sera également considérée comme une variable aléatoire distribuée uniformément sur $\mathbb{F}_2^{128}$. Nous allons maintenant présenter les propriétés $\mathcal{P}$ utilisées.

5.4.1 La propriété du poids de Hamming

Dans un premier temps, nous allons considérer une propriété simple. Nous souhaitons trouver des clefs secrètes K telles que la suite $\sigma_K^{128} = S_K$ ait un poids de Hamming au plus égal à une valeur donnée k. Cette suite sera notée $\sigma_K^{128,k+}$. L'intérêt est de considérer des valeurs faibles de k dans la mesure où une faible densité de suite (poids de Hamming faible) est une propriété difficile à réaliser statistiquement. De manière symétrique, il est possible de considérer des suites dont le poids de Hamming est au moins égal à une valeur k pour des valeurs élevées de ce paramètre.

Pour une formulation plus rigoureuse, la probabilité d'obtenir une suite $\sigma_K^{128,k+}$ au hasard, est donnée par la formule (5.1).

$$P[\sigma_K^{128,k+}] = \frac{1}{2^{128}} \times \left(\sum_{i=0}^{k} \binom{128}{i} \right) = p_{k+} \tag{5.1}$$

Cette égalité se démontre aisément en considérant des résultats élémentaires de combinatoire.

De manière similaire, nous pouvons également considérer des suites de sortie de longueur $n = 128$ dont le poids de Hamming vaut exactement k. Ces suites seront notées $\sigma_K^{128,k}$ et leur probabilité de réalisation par une recherche aléatoire est donnée par

$$P[\sigma_K^{128,k}] = \frac{1}{2^{128}} \times \binom{128}{k} = p_k.$$

En d'autres termes, utiliser la propriété du poids de Hamming permet d'affirmer que si nous voulons trouver une clef secrète K produisant à partir d'un système S une suite $\sigma_K^{128,k+}$ (respectivement une suite $\sigma_K^{128,k}$), **aucune autre méthode** qu'une recherche exhaustive ou une recherche aléatoire, de complexité $\mathcal{C}_{k+} = \frac{1}{p_{k+}}$ (respectivement $\mathcal{C}_k = \frac{1}{p_k}$) n'est connue. L'impossibilité calculatoire de ces deux approches fait que

si malgré tout une telle clef peut être exhibée, cela n'est possible qu'en vertu d'une attaque utilisant des faiblesses non publiées. Le tableau 5.5 donne les valeurs numériques de complexité pour différentes valeurs de k. Ces résultats montrent que la complexité $\mathcal{C}_{k+}$ est systématiquement

k	$\mathcal{C}_{k+}$	$\mathcal{C}_k$	k	$\mathcal{C}_{k+}$	$\mathcal{C}_k$
30	$2^{30.52}$	$2^{31.04}$	22	$2^{46.36}$	$2^{46.69}$
29	$2^{32.27}$	$2^{32.76}$	21	$2^{48.66}$	$2^{48.97}$
28	$2^{34.08}$	$2^{34.54}$	20	$2^{51.04}$	$2^{51.33}$
27	$2^{35.97}$	$2^{36.39}$	19	$2^{53.51}$	$2^{53.78}$
26	$2^{37.90}$	$2^{38.31}$	18	$2^{56.06}$	$2^{56.31}$
25	$2^{39.91}$	$2^{40.30}$	15	$2^{64.27}$	$2^{64.48}$
24	$2^{41.99}$	$2^{42.35}$	10	$2^{80.18}$	$2^{80.31}$
23	$2^{44.14}$	$2^{44.48}$	5	$2^{99.96}$	$2^{100.02}$

TABLE 5.5 – Complexité pour la réalisation de la propriété du poids de Hamming (approche aléatoire ; $n = 128$)

inférieure à la complexité $\mathcal{C}_k$. D'un point de vue pratique, il est intéressant de considérer des valeurs de $k \leq 22$ pour la preuve de cryptanalyse de type zero-knowledge.

5.4.2 La propriétés des sous-suites constantes (*runs*)

L'objectif, cette fois, est de trouver des clefs secrètes pour lesquelles le système S_K produit une séquence de sortie dont les r premiers bits sont identiques (0 ou 1). Sans perte de généralité, nous considérerons des bits nuls (*run* de zéros). Les suites satisfaisant cette propriété seront notées $\sigma_K^{128,r}$. La probabilité de trouver une telle séquence au hasard est donnée par la formule suivante :

$$P[\sigma_K^{128,r}] = \frac{2^{128-r}}{2^{128}} = \frac{1}{2^r} = p_r.$$

La complexité résultante pour ce problème est alors :

$$\mathcal{C}_r = 2^r.$$

Comme pour la propriété du poids de Hamming, si nous parvenons à trouver une clef secrète K, telle que le sysème produise une suite $\sigma_K^{128,r}$, pour des valeurs relativement importantes de r, alors nous avons prouvé de manière zero-knowledge que nous connaissons effectivement une attaque plus efficace que les recherches exhaustive ou aléatoire, lesquelles possèdent une complexité telle, qu'elles sont calculatoirement irréalisables.

Toute autre structure que des runs peut être considérée pour cette preuve, à la condition expresse que le vérifieur puisse être convaincu que cette propriété n'a pas été obtenue par un simple chiffrement et un choix *a posteriori*. C'est la raison pour laquelle le choix s'est porté sur des runs de zéros, que l'on peut considérer comme une sous-suite « remarquable ».

5.4.3 Cumul des propriétés du poids de Hamming et des run

Dans ce cas de figure, l'objectif est de trouver des clefs secrètes telles que le système S_K produit une séquence dont les r premiers bits sont tous à zéro et dont le poids de Hamming est égal à k. Une telle séquence sera notée $\sigma_K^{128,r,k}$. Il est alors facile de démontrer que la probabilité de trouver aléatoirement une telle clef est donnée par la formule suivante :

$$P[\sigma_K^{128,r,k}] = \frac{\binom{128-r}{k}}{2^{128}} = p_{r,k}.$$

La complexité correspondante vaut alors

$$\mathcal{C}_{r,k} = \frac{2^{128}}{\binom{128-r}{k}}.$$

De manière identique, nous pouvons considérer des suites de sortie contenant des sous-suites contigües constituées de bits identiques (appelées *run*). Ces sous-suites peuvent être localisées n'importe où dans la suite de sortie, et non plus uniquement en position initiale. De telles séquences seront notées $\sigma_K^{128,r+,k}$. Pour ces suites, la probabilité de trouver par une recherche aléatoire une clef produisant une telle suite est donnée par

$$P[\sigma_K^{128,r,k}] = \frac{(128-r)\binom{128-r}{k}}{2^{128}} = p_{r+,k}.$$

La complexité résultante est alors

$$\mathcal{C}_{r+,k} = \frac{2^{128}}{(128-r)\binom{128-r}{k}}.$$

La table 5.6 compare les différentes complexités $\mathcal{C}_k, \mathcal{C}_{k+}, \mathcal{C}_r$ et $\mathcal{C}_{r,k}$ pour différentes valeurs de k et de r. Les complexités données dans le tableau 5.6 concernent la recherche d'une seule clef secrète K. La complexité correspondante s'agissant de la recherche de ν clefs nécessitent de considérer un facteur multiplicatif du même ordre. Les expériences de cryptanalyse réalisées au laboratoire ont permis de retrouver un peu plus de 48 000 clefs soit un facteur de 2^{16}.

(r, k)	$\mathcal{C}_{k+}$	$\mathcal{C}_k$	$\mathcal{C}_r$	$\mathcal{C}_{r,k}$	$\mathcal{C}_{r+,k}$
(69, 29)	$2^{32.27}$	$2^{32.76}$	2^{69}	$2^{72.28}$	$2^{66.40}$
(69, 27)	$2^{35.97}$	$2^{36.39}$	2^{69}	$2^{72.57}$	$2^{66.69}$
(69, 25)	$2^{39.91}$	$2^{40.30}$	2^{69}	$2^{73.25}$	$2^{67.36}$
(68, 27)	$2^{35.97}$	$2^{36.39}$	2^{68}	$2^{71.71}$	$2^{65.80}$
(67, 26)	$2^{37.90}$	$2^{38.31}$	2^{67}	$2^{71.24}$	$2^{65.31}$
(67, 24)	$2^{41.99}$	$2^{42.35}$	2^{67}	$2^{72.27}$	$2^{66.34}$
(66, 29)	$2^{32.27}$	$2^{32.76}$	2^{66}	$2^{69.49}$	$2^{63.53}$
(66, 26)	$2^{37.90}$	$2^{38.31}$	2^{66}	$2^{70.45}$	$2^{64.50}$
(65, 28)	$2^{34.08}$	$2^{34.54}$	2^{65}	$2^{68.87}$	$2^{62.89}$
(65, 27)	$2^{35.97}$	$2^{36.39}$	2^{65}	$2^{69.23}$	$2^{63.25}$
(65, 26)	$2^{37.90}$	$2^{38.31}$	2^{65}	$2^{69.69}$	$2^{63.71}$
(64, 27)	$2^{35.97}$	$2^{36.39}$	2^{64}	$2^{68.44}$	$2^{62.44}$
(63, 29)	$2^{32.27}$	$2^{32.76}$	2^{63}	$2^{66.87}$	$2^{60.85}$
(63, 28)	$2^{34.08}$	$2^{34.54}$	2^{63}	$2^{67.23}$	$2^{61.20}$
(63, 27)	$2^{35.97}$	$2^{36.39}$	2^{63}	$2^{67.67}$	$2^{61.64}$
(62, 29)	$2^{32.27}$	$2^{32.76}$	2^{62}	$2^{66.04}$	$2^{59.99}$
(62, 28)	$2^{34.08}$	$2^{34.54}$	2^{62}	$2^{66.43}$	$2^{60.38}$
(62, 27)	$2^{35.97}$	$2^{36.39}$	2^{62}	$2^{66.91}$	$2^{60.86}$
(62, 26)	$2^{37.90}$	$2^{38.31}$	2^{62}	$2^{67.47}$	$2^{61.43}$
(60, 23)	$2^{44.14}$	$2^{44.48}$	2^{60}	$2^{68.52}$	$2^{62.43}$

TABLE 5.6 – Comparaison des complexités pour les propriétés du poids de Hamming et des runs (recherche aléatoire ; $n = 128$)

5.5 Preuve de cryptanalyse de type Zero-knowledge pour E0

Des faiblesses conceptuelles importantes ont été identifiées dans l'algorithme E0. À ma connaissance, aucune publication ouverte n'en a jamais fait mention. Etant donnée leur portée opérationnelle, il n'est pas judicieux, ni probablement légal de les rendre publiques. Comme il a été évoqué dans l'introduction de ce chapitre, l'intérêt de la preuve de cryptanalyse de type Zero-knowledge réside précisément dans la capacité de communiquer sur des faiblesses de nature cryptologique tout en satisfaisant aux clauses de confidentialité et aux contraintes légales.

Ces faiblesses sont essentiellement de nature combinatoire. Elles ont été identifiées à l'aide de deux suites logicielles que je développe depuis 1999 à des fins d'analyse cryptologique :

1. la suite *CoHS* (*Combinatorics over Huge Sets*). Il s'agit d'un scanner de « faiblesses combinatoires ». Cette suite en fait recherche dans la description mathématique d'un système de chiffrement (l'algorithme au sens premier du terme) des faiblesses de nature combinatoire. Par exemple, certaines structures comme des designs combinatoires sont plus représentées ou moins représentées que ne l'autorise la théorie dans le cadre d'un système cryptologiquement sûr ;
2. la suite *Vauban* traduit les irrégularités combinatoires détectées en outils statistiques ou algébriques, utilisables dans une cryptanalyse.

Ces deux suites ne sont pas publiques.

Les trois propriétés présentées précédemment ont été utilisées pour la cryptanalyse de E0, et ce avec succès. Pour un grand nombre de valeurs k et r intéressantes, un grand nombre de clefs secrètes ont pu être retrouvées. Pour la propriété des *runs*, sans perte de généralité, des sous-suites nulles ont été considérées. Les bits de mémoire c_{-1} et c_0 ont été également fixés à zéro. La raison de ce choix tient au défi plus grand que représenent ces valeurs. En effet, pour E0 – comme pour beaucoup de systèmes de chiffrement – la clef nulle $K = (0, 0, 0, \ldots, 0)$ produit la suite de sortie nulle. L'objectif est donc de trouver une clef non nulle produisant des suites proches de la suite nulle. Or, à la fois les qualités

statistiques excellentes de E0 et les propriétés d'avalanche du système compliquent singulièrement la tâche et interdisent des approches locales fondées sur des variations de voisinages discrets.

La phase de précalcul (recherche des irrégularités combinatoires et traduction pour la cryptanalyse) a duré approximativement une semaine sur un Athlon 64. Ce travail est effectué une fois pour toutes. Pour chaque couple de valeurs (k, r) et pour des runs différents, la phase de cryptanalyse a ensuite été lancée sur quatre machines DEC 9000. Les premières clefs ont été retrouvées en moins d'une heure tandis qu'environ cinq semaines ont suffi à retrouver un peu plus de 48 000 clefs secrètes. Quelques uns des résultats les plus significatifs sont fournis en annexe .4.

Le tableau 5.7 indique le nombre de clefs retrouvées pour les conditions les plus significatives. Les résultats les plus significatifs concernent

k	Propriété du poids de Hamming	(r, k)	Prop. cumulées
19	1	(69, 29)	1
20	5	(69+, 27)	1
21	18	(69+, 25)	1
22	38	(66+, 26)	3

TABLE 5.7 – Nombre de clefs retrouvées pour quelques valeurs de (k, r) significatives ($n = 128$)

la clef produisant une suite de sortie $\sigma_K^{128,69,29}$. Retrouver une telle clef, en l'état actuel des connaissances, n'aurait été possible que par une recherche exhaustive dont la complexité moyenne est en $\mathcal{O}(2^{72.28})$. Or cette approche exhaustive est irréalisable avec les ressources de calcul actuelles. Ce résultat prouve donc que seule la connaissance de faiblesses dans l'algorithme a permis d'obtenir ce résultat par le biais d'une cryptanalyse plus efficace que la recherche exhaustive.

L'étude de complexité de la cryptanalyse réalisée sur E0 – et qui a permis de retrouver près de 48 000 clefs – a été empiriquement évaluée dans un premier temps. En comparant le nombre de clefs effectivement traitées rapporté au temps requis par une recherche exhaustive simple, cette complexité a été évaluée à $\mathcal{O}(2^{35})$. Une analyse de complexité théorique a ensuite confirmé cette valeur. La preuve ne sera pas donnée

ici. En effet, elle constitue une information qui nuirait au caractère Zero-knowledge recherché pour cette attaque.

Finalement, à la fois les propriétés considérées et les clefs obtenues ne permettent de déduire une quelconque information sur la méthode de cryptanalyse elle-même (du moins à ce jour). Autrement dit, le lecteur/vérifieur ne peut obtenir aucune information sur la nature des failles conceptuelles de E0. Toutefois cette certitude est le fait de l'expérience. Une preuve rigoureuse requiérait de publier ces failles. Seul l'avenir confirmera la nature véritablement Zero-knowledge de cette preuve. Si personne ne parvient à reproduire ces propriétés en un temps au plus égal, de manière empirique et admissible, nous pouvons considérer qu'elle est effectivement de nature Zero-knowledge.

A ce jour, aucune des attaques connues pour E0 ne permet de reproduire des résultats similaires. Mais, déterminer si les attaques présentées dans le tableau 5.4 peuvent, sous une forme modifiée ou améliorée, produire des résultats similaires, ce problème reste pour le moment ouvert.

5.6 Travaux futurs, perspectives et conclusion

La preuve de cryptanalyse de type Zero-knowledge présentée dans ce chapitre permet, dans des conditions acceptables et dans le respect de la législation en la matière, de communiquer sur des faiblesses conceptuelles de systèmes de chiffrement, opérationnellement exploitables, en prouvant leur effectivité mais sans divulguer aucune information utilisable par une personne mal intentionnée.

Ce schéma de preuve est dans une certaine mesure déjà utilisé pour la cryptanalyse des fonctions de hachage. Produire deux messages M et M' différents tels que $H(M) = H(M')$ suffit à convaincre de la capacité à cryptanalyser H. Toutefois, à l'heure actuelle la nature des messages M et M' produits (par exemple voir [WFLY04]) permet de retrouver la technique de cryptanalyse.

Concernant les systèmes de chiffrement par blocs, évoqués dans le chapitre 4, la preuve de cryptanalyse de type Zero-knowledge est parfaitement applicable. Il suffit de considérer un bloc de texte clair fixe (éventuellement pourvu d'un sens ou présentant des propriétés

mathématiques « remarquables ») et de fixer pour le bloc de cryptogramme correspondant des propriétés propriétés mathématiques « remarquables », comme nous venons de le faire pour E0. C'est actuellement l'un de mes axes de recherche. La recherche de biais combinatoires est actuellement lancée sur plusieurs systèmes de chiffrement par blocs actuels. Les conditions expérimentales sont les suivantes :

– texte clair nul $K[0] = K[1] = 0x0LL$ (mot des 64 bits),
– propriétés du poids de Hamming cumulées avec celle des *runs*.

Les premiers résultats montrent que si l'espoir est permis de parvenir à des résultats au moins aussi pertinents que ceux obtenus pour E0, l'étape de précalcul sera plus longue. Cette contrainte est en réalité facilement admissible dans la mesure où ce précalcul est fait une fois pour toute. Mais cela illustre le fait que la complexité – en particulier combinatoire – des systèmes par bloc est de loin plus élevée que celle des systèmes par flot. Mais cela n'implique nullement que les premiers offrent plus de sécurité par rapport au second. C'est là un problème ouvert et un débat qui ne sont pas prêts d'être refermés.

En revanche, la richesse combinatoire des systèmes par bloc est telle que les possibilités de dissimuler des trappes autorisant un décryptement plus « facile » à quiconque en connaît la nature exacte, sont en nombre quasi-illimité. Les premiers résultats ont confirmé ce point. Mais surtout les trappes qui peuvent être imaginées ne se limitent pas à de simples corrélations entre des bits de clair, de cryptogramme et de clef. Il est en effet possible de construire des biais combinatoires non traduisibles en biais statistiques mais en caractéristiques algébriques (variétés particulières). La conception de telles trappes, leur étude du point de vue théorique constitue un de mes axes de recherche que je souhaite développer.

Chapitre 6

Conclusion

L'utilisation des fonctions booléennes en virologie et en cryptologie, comme dans bien d'autres domaines se révèle puissante non seulement pour décrire des phénomènes complexes mais également comme outils pour agir sur ces phénomènes.

Mon sentiment premier, d'il y a quelques années, concernant les fonctions booléennes était celui de pratiquement tout ceux qui osent se frotter à ces objets : un sentiment de rapide impuissance face à leur complexité et ce, dès un nombre relativement faible des variables d'entrée ; celui, également de se sentir rapidement submergé dès que l'on amorce la moindre tentative d'analyse. Ce sentiment est encore plus fort et difficile à éradiquer s'agissant de l'utilisation des fonctions booléennes en cryptologie et plus particulièrement en cryptanalyse.

En effet, les « bonnes » fonctions ne doivent exhiber aucune structure, du moins de structures ou propriétés calculatoirement identifiables. Mais ces structures ou ces propriétés existent, la théorie l'assure. Le meilleur exemple est celui des corrélations existant entre le texte chiffré et les bits de la clef, dans un système de chiffrement par bloc par exemple, résumées par l'équation suivante :

$$< K, w > \oplus < P, u > \oplus < C, v > \overset{p}{\simeq} \epsilon, \tag{6.1}$$

où ϵ est une constante dans $\{0, 1\}$. Un bon cryptographe aura œuvré pour que ces corrélations soient gommées pour des masques u, v et w de poids faibles, ceux qui calculatoirement peuvent encore être explorés exhaustivement. Mais cela a pour conséquence de potentiellement

concentrer les corrélations à des ordres plus élevés des masques. En effet, selon le théorème de Parseval, l'« énergie » totale d'une fonction à n variables, donnée par

$$\sum_{u \in \mathbb{F}_2^n} (\widehat{\chi_f}(u))^2 = 2^{2m},$$

est constante (où $\widehat{\chi_f}(.)$ désigne la transformée de Walsh-Hadamard). Des corrélations existent, probablement à des ordres élevés de poids des masques et pour des probabilités relativement importantes p ; mais les techniques actuelles ou les visions actuelles utilisées dans l'étude des fonctions booléennes sont impuissantes à les exhiber. La solution, pour le cryptanalyste, est là, cachée dans ces fonctions et donc inaccessible.

Avec le temps ce sentiment d'impuissance s'est estompé, au fur et à mesure de mes recherches, à force de tentatives et de beaucoup d'échecs. Du moins, l'espoir de pouvoir les appréhender de manière plus efficace à des fins « opérationnelles » est-il né peu à peu. Cet espoir s'est nourri de la vision combinatoire que j'ai développée au fil des ans et des études de fonctions booléennes. Plutôt que de tenter vainement d'embrasser les fonctions booléennes dans leur globalité, j'ai acquis la conviction, qui guide depuis mon travail de recherche, qu'il faut préférer une vision locale consistant à considérer une partition adéquate de l'ensemble $\mathbb{F}_2^n$ et à rechercher pour chacune des parties de cet ensemble, des propriétés qui lui sont spécifiques.

Autrement dit, il s'agit de conditionner l'étude d'une fonction relativement à un sous-ensemble de ses entrées. Les premières tentatives et analyses suggèrent fortement qu'une équation comme l'équation 6.1, est valide avec une probabilité p significative, sur des sous-ensembles particuliers de $\mathbb{F}_2^n$. Il ne s'agit donc plus seulement de choisir astucieusement les bits des données de clair P, de cryptogramme C et de clef K mais également d'identifier des sous-ensembles d'entrées $x \in \mathbb{F}_2^n$.

Afin d'illustrer simplement cette approche, supposons que la fonction booléenne (exemple didactique) suivante

$$f_j(p_1, \ldots, p_n, k_1, \ldots, k_n) = k_i \oplus k_j \oplus k_l \oplus g_i(k_1, \ldots, k_n, p_1, \ldots, p_n),$$

décrive le j-ième bit de sortie d'un algorithme de chiffrement par bloc en fonction des bits de clair p_i et des bits de clef k_i. Il est alors possible

de partitionner l'espace $\mathbb{F}_2^{2n}$ des blocs de clair et de clef en considérant les valeurs P et K telles que

$$g(k_1, \ldots, k_n, p_1, \ldots, p_n) \overset{p'}{\simeq} \epsilon,$$

avec des valeurs potentiellement plus fortes p'. La fonction g peut elle même être partitionnée selon le même principe, créant à chaque fois de nouvelles « variétés » concentrant des propriétés désirables pour le cryptanalyste. Disposant ainsi d'un « catalogue » de telles variétés sur le clair permettant de trier préalablement le clair [1], ce dernier n'a plus qu'à choisir la ou les équations valides pour la variété correspondante. Cette approche pouvant être généralisée à d'autres propriétés ou estimateurs que les fonctions linéaires usuellement considérées, par exemples des structures combinatoires particulières.

Cette nouvelle approche, fondée sur la combinatoire des ensembles des variables d'entrée de fonctions de booléennes et sur celle des valeurs de sortie dans le cas de fonctions booléennes vectorielles, est celle qui guide ma recherche actuelle et ce, très probablement pour de nombreuses années encore.

Un autre aspect auquel je souhaite me consacrer concerne la généralisation des fonctions booléennes à d'autres algèbres de Boole que celle classique, munie des lois ET et OU. La recherche de partitions adéquates de l'ensemble $\mathbb{F}_2^n$ peut être envisagée en considérant les opérations d'union, d'intersection et de complémentation. Considérer d'autres types de lois ou d'opérations pourra très probablement déboucher sur une vision plus adaptée à la sélection de partitionnements intéressants.

Ces deux axes très généraux de ma recherche actuelle et future permettent finalement une certaine vision unifiée de la virologie informatique et de la cryptologie. Dans les deux cas, la formalisation à l'aide de fonctions booléennes $f \in \mathcal{F}(\mathbb{F}_q^n, \mathbb{F}_q^m)$ débouche sur le souci de trouver ensuite des sous-ensembles de valeurs de x, telles que $f(x)$ réalise des propriétés désirables selon le domaine désiré :

– décrire des biais statistiques ou des relations combinatoires entre les variables de clef, de clair et de chiffré pour le cryptanalyste,

1. Même si le partitionnement se traduit par un nombre exponentiel de parties, l'approche reste valide si l'on considère que le clair qui intéresse le cryptanalyste en pratique, concernera finalement qu'un sous-ensemble réduit de ces parties.

– décrire des caractéristiques particulières de classes de codes malveillants, débouchant sur une meilleure détection, en virologie informatique. Du point de vue de l'attaquant au contraire, il peut s'agir d'identifier des « zones de non détection » dans l'espace d'entrée, qu'il pourra exploiter pour produire des codes non détectables, relativement aux fonctions de détection utilisées.

La tâche est immense et les idées nombreuses. Cette idée seule, en soi, est déjà stimulante.

Mon souhait est de pouvoir attirer de nombreux étudiants vers ces nouvelles voies de recherche et leur faire partager l'excitation qui est la mienne face aux défis à relever ; de leur faire découvrir les mondes passionnants de la virologie et de la cryptologie et les outils excitants pour l'esprit que sont les fonctions booléennes, tout cela avec le double souci de la rigueur procurée par la formalisation mais également de la réalité opérationnelle. Dans des domaines comme la cryptologie et la virologie informatique comprendre l'espace dans lesquels évoluent les objets et les outils est certes intéressant, être capable d'agir de manière profonde sur cet espace est encore plus satisfaisant.

Bibliographie

[Adleman88] Adleman L. M. (1988), An Abstract Theory of Computer Viruses. In *Advances in Cryptology- CRYPTO'88*, pp 354-374, Springer.

[AES] *http ://www.nist.gov/aes/*

[Angluin88] Angluin D (1988), "Queries and Concept Learning", *Machine Learning*, 2-4 : 319–342.

[AK03] Armknecht F. and Krause M. (2003), "Algebraic Attacks on Combiners with Memory". In *Advances in Cryptology - CRYPTO'03*, LNCS 2729, pp. 162–175, Springer Verlag.

[ArsFaug04] Ars G. et Faugère J.-C. (2004), Comparison of XL and Gröbner basis alhorithms over Finite Fields. Rapport de recherche INRIA RR-5251, juillet 2004.

[Beau84] Beauchamp K. G. (1984), *Applications of Walsh and related functions*, Microelectronics and Signal Processing Series, Academic Press, ISBN 0-12-084180-0.

[BJL99] Beth T, Jungnickel D, Lenz H (1999), *Design Theory*, Cambridge University Press, ISBN 0-5214-4432-2 (Vol. 1) and ISBN 0-5217-7231-1

[BiSha91] E. Biham, A. Shamir, *Differential Cryptanalysis of DES-like Cryptosystems*, J. of Cryptology, **4 - 1**, (1991), 3–72.

[BLEI01] Bleichenbacher D. (2001), Personal communication in Jakobsson M. and Wetzel S., "Security weaknesses in Bluetooth" in *Proc. RSA Security Conf. – Cryptographer's Track*, LNCS 2020, pp. 176–191, Springer-Verlag.

[BKM05] Bonfante G., Kaczmarek M. et Marion J.-Y. (2005), On abstract computer virology from a recursion-theoretic perspective, *Journal in Computer Virology*, 1 :(3-4).

[RMD128] A. Bosselaers, B. Preenel editors, *Intregrity Primitives for Secure Information Systems : Final Report of RACE Integrity Primitives Evaluation RIPE-RACE 1040*, LNCS 1007, Springer, 1995.

[CABIR] Description of the *Epoc.Cabir* virus. `www.f-secure.com/v-descs/cabir.shtml`

[CAM88] Camion P., *Majority Decoding of Large Repetition Codes for the R-ary Symmetric Channel*. In : Proceedings of the AAECC'88 Conference, Lecture Notes in Computer Science 357, 458–466, Springer Verlag, 1989.

[ChaHayes98] Chakrabarty K. et Hayes J. P. (1998), Balanced Boolean Functions. *IEE Proc.-Comput. Digit. Tech.*, Vol. 145, No. 1.

[Chess00] Chess D. M., White S. R. (2000) An undetectable computer virus, *Virus Bulletin Conference*, September.

[Cohen86] Cohen F. (1986) *Computer viruses*, Ph. D Thesis, University of Southern California, Janvier 1986.

[ColDin96] Colbourn CJ and Dinitz JH (1996) *Handbook of Combinatorial Designs*, CRC Press, ISBN 0-8493-8948-8

[CJ04] Christodorescu M, Jha S (2004) "Testing Malware Detectors". In *Proceedings of the ACM International Symposium on Software Testing and Analysis (ISSTA'04)*

[PCJD07] Preda M. D., Christodorescu M., Jha S, Debray S (2007), "A Semantic-based Approach to Malware Detection", In : *Proceedings of the Symposium on Principles of Programming Languages - POPL'07*, January 2007.

[NC03] Courtois, N. (2003), "Fast Algebraic Attacks on Stream Ciphers with Linear Feedback". In *Advances in Cryptology - CRYPTO'03*, LNCS 2729, pp. 176-194, Springer-Verlag.

[AES02] Daemen J., Rijmen V., *The Design of Rijndael : AES - The Advanced Encryption Standard*, Springer Verlag, 2002.

[DKF63] Denis-Papin M, Kaufmann A et Faure R (1963) *Cours de calcul booléien appliqué*, Coll. Bibliothèque de l'ingénieur électricien-mécanicien, Albin Michel éditeur

[DIL72] Dillon J. F. (1972) A Survey of Bent Functions, *NSA Technical Journal*, Special Issue, 191–215.

[DIL74] Dillon J. F. (1974) *Elementary Hadamard Difference Sets*, PhD Thesis, University of Maryland.

[RMD160] H. Dobbertin, A. Bosselaers, B. Preenel, RIPEMD-160 : a Strengthened Version of RIPEMD. In. *D. Gollman ed., Fast Software Encryption, Third International Workshop*, LNCS 1039, Springer, 1996.

[ECRYPT] eSTREAM, ECRYPT Stream Cipher Project, `http://www.ecrypt.eu,org/stream`

[FAUG03] Faugère J.-C., Joux A. (2003), Algebraic cryptanalysis of hidden field equations (HFE) cryptosystems using Gröbner bases. In *Advances in Cryptology - CRYPTO 2003*, LNCS 2729, pp. 44-60, Springer.

[Fell66] W. Feller, *An Introduction to Probability Theory*, Wiley, 1966.

[FiFont98] Filiol E., Fontaine C. Highly Nonlinear Balanced Boolean Functions with a good Correlation Immunity. In *Advances in Cryptology - EUROCRYPT'98*, Lecture Notes in Computer Science 1403, Springer Verlag, 1998.

[Fil01] Filiol E. (2001), *Techniques de reconstruction en cryptologie et en théorie des codes*, Thèse de doctorat, Ecole Polytechnique.

[Fil02] Filiol E (2002) A New Statistical Testing for Symmetric Ciphers and Hash Functions. In : *Proceedings of the 4th International Conference on Information and Communication Security 2002*, Lecture Notes in Computer Science, Vol. 2513, pp. 342–353, Springer Verlag.

[Fil03] Filiol E. (2003), *Les virus informatiques : théorie, pratique et applications*, collection IRIS, Springer Verlag France, XXIV+388 pages, ISBN 2-287-20297-8, 2003.

[Fil05b] Filiol E. (2005) *Computer Viruses : from Theory to Applications*, IRIS International series, Springer Verlag, ISBN 2-287-23939-1.

[Fil05] Filiol E. (2005) Repetition Codes Cryptanalysis of Block Ciphers. *Journal of the Indian Statistical Association*, Volume 42, Number 9, 2004.

[Fil05b] Filiol E. (2005) La simulabilité des tests statistiques. *Journal de la sécurité informatique MISC*, numéro 22.

[FHZ05] Filiol E., Helenius M. et Zanero S. (2005) Open problems in computer virology, *Journal in Computer Virology*, Vol. 1, Nr. 3-4.

[Fil06a] Filiol E. (2006) Malware Pattern Scanning Schemes Secure Against Black-box Analysis. *Journal in Computer Virology*, 2 (1).

[Fil06b] Filiol E. (2006) Techniques virales avancées, collection IRIS, Springer Verlag France, à paraitre.

[Fil06c] Filiol E. (2006) Preuve de type zero knowledge de la cryptanalyse du chiffrement Bluetooth. *Journal de la sécurité informatique MISC*, numéro 26, juillet-août 2006.

[Fil06d] Filiol E. (2006), Zero-knowledge-like Proof of Cryptanalysis of Bluetooth Encryption, *IACR Preprint 2006/303*. Disponible sur `http://eprint.iacr.org/2006/303`

[FJL06] Filiol E., Jacob G. et Le Liard M. (2006), Evaluation Methodology and Theoretical Model for Antiviral Behavioural Detection Strategies, *Journal in Computer Virology, WTCV'06 Special Issue*, G. Bonfante & J.-Y. Marion eds, (3), 1.

[Fil07] Filiol E. (2007), Zero-knowledge-like Proof of Cryptanalysis of Bluetooth Encryption, *International Journal in Information Technology*, accepté pour publication, à paraitre.

[DES77] FIPS 46-3, *Data Encryption Standard*, Federal Information Processing Standards Publication 46-3, US Dept of Commerce/NIST, 1977, Reaffirmed October 25th, 1999.

[FIPS140] FIPS 140-1, *Security Requirements for Cryptographic Modules*, Federal Information Processing Standards Publication 140-1, US Dept of Commerce/NIST, 1994.

[FIPS180] FIPS 180, *Secure Hash Standard*, Federal Information Processing Standards Publication 180, US Dept of Commerce/NIST, 1993.

[FIPS181] FIPS 180-1, *Secure Hash Standard*, Federal Information Processing Standards Publication 180-1, US Dept of Commerce/NIST, 1995.

[FIPS197] FIPS 197, *Advanced Encryption Standard*, Federal Information Processing Standards Publication 197, US Dept of Commerce/NIST, November 26th, 2001.

[FLUH01] Fluhrer S. and Lucks S. (2001), "Analysis of the E0 Encryption System", *Selected Areas in Cryptography - SAC 2001*, LNCS 2259, pp. 38–48, Springer-Verlag.

[FLUH02] Fluhrer S. (2002), "Improved Key Recovery of Level 1 of the Bluetooth Encryption System", available at `http://eprint.iacr.org/2002/068`

[FOR66] G.D. Forney (1966), *Concatenated Codes*, M.I.T Press, Cambridge MA.

[Forrest97] Forrest S., Hofmeyr S. A. et Somayaji A. (1997) Computer Immunology. In *Communications of the ACM*, Vol. 40, No 10, Octobre, pp. 88-96.

[Fortinet] Fortinet, `http://www.fortinet.com/FortiGuardCenter/global_threat_stats.html`

[GGPS98] Goldberg LA, Goldberg PW, Phillips CA and Sorkin GB (1998)"Constructing computer virus phylogenies," *Journal of Algorithms*, 26 : 188–208

[GOLD01] Goldreich O. (2001), *Foundations of Cryptography – Basic Tools*. Cambridge University Press, Cambridge.

[GMR89] Goldwasser S., Micali S. and Rackoff C. (1989), "The Knowledge-complexity of Interactive Proof Systems", *SIAM Journal on Computing*, Vol. 18, pp. 186–208.

[GBM02] Golic J., Bagini V. and Morgani G., "Linear cryptanalysis of Bluetooth stream cipher". In *Advances in Cryptology - EUROCRYPT'02*, LNCS 2332, pp. 238–255, Springer, 2002.

[Gol82] S.W. Golomb, *Shift Register Sequences*, Aegean Park Press, 1982.

[Grätzer03] Grätzer G. (2003) *General Lattice Theory*, 2nd edition, Birkhauser, ISBN 3-7643-6996-5.

[HKM95] C. Harpes, G. G. Kramer, J. L. Massey 1995), A Generalisation of Linear Cryptanalysis and the Applicability of Matsui's Piling-up Lemma. In *Advances in Cryptology, Eurocrypt'95*, Lecture Notes in Computer Science, Springer Verlag 963, 24-38, 1995.

[HT88] R.V. Hogg, E.A. Tanis, *Probability and Statistical Inference*, MacMillan, 1988.

[KA94] Kephart JO and Arnold W (1994) "Automatic Extraction of computer Virus Signatures". In *Proceedings of the 4th Virus Bulletin International Conference*, pp. 179–194, Virus Bulletin Ltd

[KWL05] Karim Md E, Walenstein A and Lakhotia A (2005) "Malware Phylogeny Generation using Permutations of Code", *Journal in Computer Virology*, Vol. 1, 1-2, pp. 13–23

[KV94] Kearns M and Vazirani U (1994) *An Introduction to Computational Learning Theory*, MIT Press, Cambridge, MA, ISBN 0-262-11193-4

[Kinder05] Kinder J., S. Katzenbeisser, C. Schallhart et H. Veith (2005), Detecting Malicious Code by Model Checking. In *Proceedings of the 3rd DIMVA Conference*, LNCS 3548, pp. 174 – 187, Springer Verlag.

[Knuth81] D.E. Knuth *The Art of Computer Programming*, Vol. 2, Addison Wesley, 1981.

[KK03] Kulesza K and Kotulski Z (2003) "On Secret Sharing for Graphs", `arxiv.org/cs.CR/0310052`

[KRAUSE02] Krause M. (2002), "BDD-based cryptanalysis of keystream generators". In *Advances in Cryptology - EUROCRYPT 02*, LNCS 2332, pp. 222–237, Springer-Verlag.

[LEIT94] Leitold F. (1994), *Theory and practice of the detection of computer viruses* (en hongrois), thèse de doctorat Académie des sciences hongroises, Budapest.

[Lejeune05] Lejeune M. (2005), *Statistique : la théorie et ses applications*. Collection Statistique et probabilités appliquées, Springer Verlag.

[LW05] Levy O. and Wool A. (2005), "A Uniform Framework for Cryptanalysis of the Bluetooth E0 Cipher". Available at `eprint.iacr.org/2005/107.pdf`

[LCEN] Loi pour la confiance en l'économie numérique (Law for Confidence in the e-Economy), Journal Officiel, June 22nd, 2004. Une présentation détaillée et commentée de cette loi peut être trouvée dans [Fil03, Chap. 5].

[LU04] Lu Y. and Vaudenay S. (2004), "Faster correlation attack on Bluetooth keystream generator E0". In *Advances in Cryptology - CRYPTO 04*, LNCS 3152, pp. 407–425, Springer-Verlag.

[LU05] Lu Y., Meier W. and Vaudenay S. (2005), "The Conditional Correlation Attack : A Practical Attack on Bluetooth Encryption". In *Advances in Cryptology - CRYPTO'05*, LNCS 3621, pp. 97–117, Springer Verlag.

[1] MacWilliams F.J., N.J.A. Sloane N.J.A., *The Theory of Error-Correcting Codes*, North-Holland, 1977.

[Massey69] J.L. Massey, Shift-Register Synthesis and BCH Decoding, *IEEE Trans. on Inf. Th.*, Vol. IT-15, pp 122–127, 1969.

[Matsui94] Matsui M., "Linear Cryptanalysis Method for DES Cipher". In : *Advances in Cryptology - Eurocrypt'93*, Lecture Note in computer Science 765, 386–397, Springer Verlag, 1994.

[Matsui94b] Matsui M., "The First Experimental Cryptanalysis of the Data Encryption Standard". In : *Advances in Cryptology - Crypto'94*, Lecture Notes in Computer Science 839, pp 1–11, Springer Verlag, 1994.

[Maurer92] U. Maurer, A Universal Statistical Test for Random Bit Generators, *J. of Cryptology, 5* pp 89-105, 1992.

[McCarthy86] P. J. McCarthy. *Introduction to Arithmetical Functions.* Springer, 1986.

[McClus56] McCluskey EJ (1956) Minimization of Boolean Functions, *Bell System Tech. J.*, Vol. 35, Nr 5, pp. 1417–1444

[McEl77] McEliece R. (1977), *The Theory of Information and Coding*, Addison Wesley.

[MvOV97] Menezes AJ, van Oorschot PC and Vanstone SA (1997), *Handbook of Applied Cryptography*, CRC Press, ISBN 0-8493-8523-7

[Mich00] Michaels JG (2000) "Algebraic Structures". In *Handbook of Discrete and Combinatorial Mathematics*, Rosen KH editor, pp. 344–354, CRC Press, ISBN 0-8493-0149-1

[Mont96] Montgomery D. C. (1996), *Design and Analysis of Experiments*, Wiley, ISBN 0-471-15746-5.

[Nesssie] *http ://www.cryptonessie.org*

[OS98] D. Olejár, M. Stanek, On Cryptographic Properties of Random Boolean Functions, *Electronic Journal of Universal Computer Science, Vol. 4, Issue 8*, 1998.

[Papadim95] Papadimitriou CH (1995), *Computational Complexity*, Addison Wesley, ISBN 0-201-53082-1

[Quine52] Quine VW (1952), "The Problem of Symplifying Truth Functions", *The American Mathematical Monthly*, Vol. 59, Nr. 8, pp. 521–531

[Quine53] Quine VW (1955), "A Way to Simplify Truth Functions", *The American Mathematical Monthly*, Nov. 1955.

[RijPren97] Rijmen V., Preneel B. (1997), A Family of Trapdoor Ciphers. In *Biham E. ed, Fast Software Encryption*, Lecture Notes in Computer Sciences 1267, pp. 139–148.

[MD4] R.L. Rivest, The MD4 Message Digest Algorithm, *Advances in Cryptology - CRYPT0'90*, LNCS 537, Springer, 1991.

[MD5] R.L. Rivest, The MD5 Message Digest Algorithm, Internet Request for Comment 1321, April 1992.

[SAARINEN01] Saarinen, M.-J., "A Software Implementation of the BlueTooth Encryption Algorithm E0". Available at `http://www.jyu.fi/~mjos/e0.c`

[SAARINEN06] Saarinen M.-J. O. (2006), Chosen-IV Statistical Attack on eSTREAM Stream Ciphers, eSTREAM, ECRYPT Stream Cipher Project, Report 2006/013, `http://www.ecrypt.eu.org/stream/`

[SCHMALL02] Schmall M. (2002), *Classification and Identification of Malicious Code Based on Heuristic Techniques Utilizing Meta Languages*, Thèse de doctorat, Faculté d'Informatique, Université de Hambourg.

[Schneier96] B. Schneier, *Applied Cryptography*, Wilew et Sons, 2nd ed., 1996.

[SW05] Shaked Y. and Wool A. (2005), Cracking the Bluetooth PIN. In *Proc. 3rd USENIX/ACM Conf. Mobile Systems, Applications, and Services (MobiSys)*, Seattle, pp. 39–50, ISBN 1-931971-31-5.

[Shan48] Shannon CE (1948), "A Mathematical Theory of Communication". *Bell Syst. Tech. J.*, Vol. 27, pp. 379–423 and 623–656.

[Sieg84] T. Siegenthaler, Correlation Immunity of Nonlinear Combining Functions for Cryptographic Applications, *IEEE Trans. on Inf. Th.*, Vol. IT 35, pp 776–780, 1984.

[BlueSpec] "Specification of the Bluetooth system", v.2.0. Core specification, 2004. Available from `http://www.bluetooth.org/foundry/adopters/document/Core_v2.0_EDR/en/1/Core_v2.0_EDR.zip`

[Spin03] Spinellis D. (2003), Reliable Identification of Bounded-length Viruses is NP-complete, *IEEE Transactions in Information Theory*, 49 :(1).

[UK_DTI04] Stimms S, Potter C and Beard A (2004), "2004 Information Security Breaches Survey", UK Department of Trade and Industry, 2004. Disponible sur `http://www.security-survey.gov.uk`.

[NIST_STS] Revised NIST Special Publication 88-22 (2000), "A Statistical Test Suite for the Validation of Ramdom Number Generator and Pseudo-random Number Generator for Cryptographic Applications". National Institute of Standard and Technology, US Commerce Department's Technology Administration, `http://csrc.nist.gov/rng/rng2.html`

[Szor05] Szor P (2005), *The Art of Computer Virus Research and Defense*, Symantec Press and Addison Wesley, ISBN 9-780321-304544

[Tur36] Turing A. M. (1936) On computable numbers with an application to the *Entscheidungsproblem*, Proc. London Math. Society, 2, 42, pp. 230-265.

[USPP05] US Government Protection Profile, *Anti-virus Applications for Workstations in Basic Robustness Environments*, Version 1.0, January 2005. Available at `www.iatf.net/protection_profiles/index.cfm`

[DMCA98] U.S. Copyright Office Summary (1998), "The Digital Millenium Copyright Act of 1998", `http://www.copyright.gov/legislation/dmca.pdf`

[VXHeavens] VX Heavens Database, `vx.netlux.org`

[WFLY04] Wang X., Feng D., Lai X. et Yu H. (2004) Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, disponible sur `http://eprint.iacr.org/2004/199`

[Weg87] Wegener I (1987), *The complexity of Boolean functions*, Wiley

[XioMass88] Xiao G.-Z. et Massey J. L. (1988), A spectral characterization of correlation-immune combining functions, *IEEE Transactions on Information Theory*, Vol. IT-34 Nr 3, pp 569–571.

[Haval93] Y. Zheng, J. Pieprzyk, J. Seberry (1993), HAVAL - A One-way Hashing Algorithm with Variable Length of Output, *Advances in Cryptology - AUSCRYPT'92*, LNCS 718, Springer.

[2] V.A. Zinov'ev (1976), "Generalized Concatenated Codes", *Problemy Peredachi Informatsii*, *12 - 1*, pp. 5-15.

[ZZ04] Zuo Z. et Zhou M. (2004), Some further theoretical results about computer viruses, *The Computer Journal*, 47 :(6).

[ZZ05] Zuo Z. et Zhou M. (2005), On the Time Complexity of Computer Viruses, *IEEE Transactions in Information Theory*, 51 :(8).

Annexes

.1 Résultats d'analyse d'antivirus

Quelques résultats particulièrement parlants de l'analyse en boîte noire des principaux antivirus sont donnés ici. Ils ont été obtenus à partir de trente variantes du ver *W32.Bagle*. Le tableau 1 présente les produits testés. Toutes les variantes testées sont été soumises d'abord

Produits	Version	Base de signatures
Avast	4.6.691	0527-2
AVG	7.0.338	267.9.2/52
Bit Defender	7.2	09/16/05
DrWeb	4.33.2.12231	02/25/06 (104186)
eTrust	7.1.194	11.5.8400 (Vet)
eTrust	7.1.194	23.65.44 (InoculateIT)
F-Secure 2005	5.10-480	09/15/05
G-Data	AVK 16.0.3	KAV-6.818/BD-16.864
KAV Pro	5.0.383	09/19/05
McAfee 2006	-	DAT 4535
NOD 32	2.5	1.1189 (08/08/05)
Norton 2005	11.0.2.4	09/15/05
Panda Titanium 2006	5.01.02	01/17/06
Sophos	5.0.5 R2	3.98
Trend Office Scan	6.5 - 7.100	2.837.00

TABLE 1 – Antivirus testés (versions & bases de signatures)

à l'algorithme E-1 de la section 2.3.2 puis, en cas d'échec à l'algorithme

général E-2 (section 2.3.3). Chaque table contient les indices des octets contenant la signature plutôt que la signature elle-même.

Produit	Taille de la signature (en octets)	Signature (indices)
Avast	29	14,128 → 14,144, 14,146 → 14,157, 14,159
AVG	7,297	0 - 1 - 60 - 200 - 201 - 220 - 469...
Bit Defender	6	0 - 1 - 60 - 200 - 201 - 206
DrWeb	12	0 - 1 - 60 - 200 - 201 - 206 - 220... 222 - 461 - 465 - 469
eTrust/Vet	3,700	0 - 1 - 60 - 200 - 201 - 206 - ...
eTrust/InoculateIT	3,700	0 - 1 - 60 - 200 - 201 - 206 - ...
F-Secure 2005	11	0 - 1 - 60 - 200 - 201 - 206 220 - 240 - 241 - 461 - 469
G-Data	6	0 - 1 - 60 - 200 - 201 - 206
KAV Pro	11	The same as *F-Secure 2005*
McAfee 2006	7	0 - 1 - 60 - 200 - 201 - 206 - 220
NOD 32	7,449	0 - 1 - 60 - 200 - 201 - 204 - 205...
Norton 2005	0	(not an AND function)
Panda Tit. 2006	9	0 - 1 - 60 - 200 - 201 - 206 220 - 461 - 541
Sophos	28	0 - 1 - 60 - 200 - 201 - 206 - 220...
Trend Office Scan	7	0 - 1 - 60 - 200 - 201 - 206 - 220

TABLE 2 – Motifs de détection pour *W32/Bagle.A*

Produit	Taille de la signature (en octets)	Signature (indices)
Avast	9	8,162 - 8,166 - 8,170 - 8,173 - 8,175 8,180 - 8,181 - 8,187 - 8189
AVG	13,288	0 - 1 - 60 - 216 - 217 - 222 - 236...
Bit Defender	87	2,831 - 2,964 - ....
DrWeb	1,773	0 - 1 - 60 - 216 - 217 - 222 - 236...
eTrust/Vet	4,306	0 - 1 - 60 - 216 - 217 - 222 - ...
eTrust/InoculateIT	4,306	0 - 1 - 60 - 216 - 217 - 222 - ...
F-Secure 2005	105	0 - 1 - 60 - 216 - 217 - 222 - ...
G-Data	3	2831 - 2964 - 2965
KAV Pro	105	The same as *F-Secure 2005*
McAfee 2006	12,039	0 - 1 - 60 - 216 - 217 - 222 - ...
NOD 32	4,793	0 - 1 - 60 - 216 - 217 - 220 - 221...
Norton 2005	0	(not an AND function)
Panda Tit. 2006	37	0 - 1 - 59 - 60 - 216 - 217 - 222...
Sophos	15,028	0 - 1 - 2 - 4 - 8 - 12 - 13 - 16 - 24...
Trend Office Scan	146	0 - 1 - 60 - 216 - 217 - 222 - ...

TABLE 3 – Motifs de détection pour *W32/Bagle.E*

Produit	Taille de la signature (en octets)	Signature (indices)
Avast	8	7,256 → 7,259 7,278 → 7,281
AVG	7,276	5739 - 5864 - 5866 - ...
Bit Defender	3,342	7,385 - 7,386 - ....
DrWeb	2,896	0 - 1 - 60 - 144 - 145 - 150 - 164...
eTrust/Vet	4,320	0 - 1 - 60 - 144 - 145 - 150 - 164...
eTrust/InoculateIT	1,311	0 - 1 - 60 - 144 - 145 - 150 - 164...
F-Secure 2005	3,128	0 - 1 - 60 - 144 - 145 - 150 - 164...
G-Data	2,954	0 - 1 - 60 - 144 - 145 - 150 - 445...
KAV Pro	3,128	The same as *F-Secure 2005*
McAfee 2006	8,084	0 - 1 - 60 - 144 - 145 - 150 - 164...
NOD 32	9,629	0 - 60 - 144 - 145 - 148 - 149 - ...
Norton 2005	8	0 - 1 - 60 - 144 - 145 150 - 164 - 445
Panda Tit. 2006	437	0 - 1 - 32 → 35 - 60 - 64...
Sophos	3,429	0 - 1 - 60 - 144 - 145 - 150 - 164...
Trend Office Scan	63	7,750 - ...

TABLE 4 – Motifs de détection pour *W32/Bagle.J*

Produit name	Taille de la signature (en octets)	Signature (indices)
Avast	10	14,567 - 14,574 → 14,577 14,581 - 14,585 → 14,588
AVG	13,112	533 → 663 - 665 - ...
Bit Defender	6,093	0 - 1 - 60 - 128 - 129 - 134 - ...
DrWeb	6,707	0 - 1 - 60 - 128 - 129 - 132 - 133...
eTrust/Vet	4,343	0 - 1 - 60 - 128 - 129 - 134 - ...
eTrust/InoculateIT	1,276	0 - 1 - 60 - 128 - 129 - 134 - ...
F-Secure 2005	54	0 - 1 - 60 - 128 - 129 - 548 - ...
G-Data	53	0 - 1 - 60 - 128 - 129 - 548 - ...
KAV Pro	54	The same as *F-Secure 2005*
McAfee 2006	4,076	0 - 1 - 60 - 128 - 129 - 134 - ...
NOD 32	17,777	0 - 1 - 60 - 128 - 129 - 132 - 133...
Norton 2005	51	0 - 1 - 60 - 128 - 129 - 134 - 429 - ...
Panda Tit. 2006	1659	0 - 1 - 4 - 8 - 12 - 13 - 16 - 24...
Sophos	7,538	0 - 1 - 60 - 128 - 129 - 134 - 148...
Trend Office Scan	44	0 - 1 - 60 - 128 - 129 - ...

Table 5 – Motifs de détection pour *W32/Bagle.N*

Produit	Taille de la signature (en octets)	Signature (indices)
Avast	8	12,916 → 12,919 12,937 → 12,940
AVG	14,575	533 → 536 - 538 - ...
Bit Defender	8,330	0 - 1 - 60 - 128 - 129 - 134 - ...
DrWeb	6,169	0 - 1 - 60 - 128 - 129 - 134 - ...
eTrust/Vet	1,284	0 - 1 - 60 - 128 - 129 - 134 - ...
eTrust/InoculateIT	1,284	0 - 1 - 60 - 128 - 129 - 134 - ...
F-Secure 2005	59	0 - 1 - 60 - 128 - 129 - 546 - ...
G-Data	54	0 - 1 - 60 - 128 - 129 - 546 - ...
KAV Pro	59	The same as *F-Secure 2005*
McAfee 2006	12,1278	0 - 1 - 60 - 128 - 129 - 134 - ...
NOD 32	21,849	0 - 1 - 60 - 128 - 129 - 132 - 133 - ...
Norton 2005	6	0 - 1 - 60 - 128 - 129 - 134
Panda Tit. 2006	7,579	0 - 1 - 60 - 134 - 148 - 182 - 209...
Sophos	8,436	0 - 1 - 60 - 128 - 129 - 134 - 148...
Trend Office Scan	88	0 - 1 - 60 - 128 - 129 - ...

TABLE 6 – Motifs de détection pour *W32/Bagle.P*

.2 Motif de détection et fonction de non détection de Norton pour *W32.Bagle.E*

L'algorithme E-1 a montré que l'antivirus Norton n'utilisait pas, pour détecter le ver *W32.Bagle.E*, la fonction de détection ET. C'est également le cas pour d'autres variantes de ce ver.

Dans ces cas particuliers, l'algorithme E-2, quant à lui, extrait systématiquement le motif de détection $\mathcal{S}_{\mathcal{M},\mathcal{M}}$ et la fonction de non détection $\overline{f_{\mathcal{M}}}$. Nous donnons ici uniquement la forme normal disjonctive de $\overline{f_{\mathcal{M}}}$, non minimale obtenue en sortie de procédure LOGICALMINIMIZE. Il est à signaler que d'un point de vue complexité la minimisation logique, les fonctions utilisées par Norton sont faibles et n'offrent aucune résistance à l'attaquant. En effet, elles ne dépendent que d'un faible nombre de variables (autrement dit la taille en octets est limitée : entre 15 et 25 selon les variantes). De plus, les formes disjonctives sont creuses.

Pour le ver *W32.Bagle.E*, l'algorithme E-2 a produit le motif de détection suivant dont les indices sont

$$E = \{0, 1, 60, 61, 62, 63, 216, 217, 222, 223, 236, 237, 645, 646, 647, 648\}.$$

La fonction de non détection est donnée avec la convention que x_i signifie que l'octet localisé à l'index i dans le code viral n'a pas été modifié tandis que $\overline{x_i}$ indique qu'il a été modifié.

$$
\begin{aligned}
&\overline{f_{\mathrm{W32/Bagle.E}}} = \\
&\overline{x_0}.x_{60}.x_{61}.x_{62}.x_{63}.x_{216}.x_{217}.x_{222}.x_{223}.x_{236}.x_{237}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &\overline{x_0}.\overline{x_1}.\overline{x_{60}}.\overline{x_{61}}.\overline{x_{62}}.\overline{x_{63}}.x_{216}.x_{217}.x_{222}.x_{223}.x_{236}.x_{237}.x_{646}.x_{646}.x_{647}.x_{648} \\
\vee\ &\overline{x_0}.\overline{x_1}.\overline{x_{60}}.\overline{x_{61}}.\overline{x_{62}}.\overline{x_{63}}.\overline{x_{216}}.\overline{x_{217}}.\overline{x_{222}}.\overline{x_{223}}.\overline{x_{236}}.\overline{x_{237}}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &\overline{x_0}.\overline{x_1}.\overline{x_{60}}.\overline{x_{61}}.\overline{x_{62}}.\overline{x_{63}}.\overline{x_{216}}.\overline{x_{217}}.\overline{x_{222}}.\overline{x_{223}}.\overline{x_{236}}.\overline{x_{237}}.\overline{x_{645}}.\overline{x_{646}}.\overline{x_{647}}.\overline{x_{648}} \\
\vee\ &x_0.\overline{x_1}.x_{60}.x_{61}.x_{62}.x_{63}.x_{216}.x_{217}.x_{222}.x_{223}.x_{236}.x_{237}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.\overline{x_{60}}.\overline{x_{61}}.\overline{x_{62}}.\overline{x_{63}}.x_{216}.x_{217}.x_{222}.x_{223}.x_{236}.x_{237}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.\overline{x_{60}}.x_{61}.x_{62}.x_{63}.x_{216}.x_{217}.x_{222}.x_{223}.x_{236}.x_{237}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.\overline{x_{60}}.\overline{x_{61}}.x_{62}.x_{63}.x_{216}.x_{217}.x_{222}.x_{223}.x_{236}.x_{237}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.\overline{x_{60}}.\overline{x_{61}}.\overline{x_{62}}.\overline{x_{63}}.x_{216}.x_{217}.x_{222}.x_{223}.x_{236}.x_{237}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.x_{60}.\overline{x_{61}}.x_{62}.x_{63}.x_{216}.x_{217}.x_{222}.x_{223}.x_{236}.x_{237}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.\overline{x_{62}}.x_{63}.x_{216}.x_{217}.x_{222}.x_{223}.x_{236}.x_{237}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.\overline{x_{63}}.x_{216}.x_{217}.x_{222}.x_{223}.x_{236}.x_{237}.x_{645}.x_{646}.x_{647}.\overline{x_{648}} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.x_{62}.x_{63}.\overline{x_{216}}.\overline{x_{217}}.\overline{x_{222}}.\overline{x_{223}}.\overline{x_{236}}.\overline{x_{237}}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.x_{62}.x_{63}.\overline{x_{216}}.x_{222}.x_{223}.x_{236}.x_{237}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.x_{62}.x_{63}.\overline{x_{216}}.\overline{x_{217}}.\overline{x_{222}}.\overline{x_{223}}.x_{236}.x_{237}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.x_{62}.x_{63}.x_{216}.\overline{x_{217}}.x_{222}.x_{223}.x_{236}.x_{237}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.x_{62}.x_{63}.x_{216}.x_{217}.\overline{x_{222}}.x_{223}.x_{236}.x_{237}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.x_{62}.x_{63}.x_{216}.x_{217}.\overline{x_{223}}.x_{236}.x_{237}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.x_{62}.x_{63}.x_{216}.x_{217}.x_{222}.x_{223}.\overline{x_{236}}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.x_{62}.x_{63}.x_{216}.x_{217}.x_{222}.x_{223}.\overline{x_{237}}.x_{645}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.x_{62}.x_{63}.x_{216}.x_{217}.x_{222}.x_{223}.x_{237}.\overline{x_{645}}.\overline{x_{646}}.\overline{x_{647}} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.x_{62}.x_{63}.x_{216}.x_{217}.x_{222}.x_{223}.x_{237}.\overline{x_{645}}.\overline{x_{646}}.\overline{x_{647}}.x_{648} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.x_{62}.x_{63}.x_{216}.x_{217}.x_{222}.x_{223}.x_{237}.\overline{x_{645}}.x_{646}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.x_{62}.x_{63}.x_{216}.x_{217}.x_{222}.x_{223}.x_{237}.x_{645}.\overline{x_{646}}.x_{647}.x_{648} \\
\vee\ &x_0.x_1.x_{60}.x_{61}.x_{62}.x_{63}.x_{216}.x_{217}.x_{222}.x_{223}.x_{237}.x_{645}.\overline{x_{647}}.x_{648}
\end{aligned}
$$

.3 Implémentation de référence de l'algorithme E0

L'implémentation de E0 donnée ici est en langage C. C'est celle qui a été utilisée pour la cryptanalyse présentée dans ce mémoire. Elle reprend en grande part l'implémentation de référence de Saarinen [SAARINEN01]. Elle est donnée afin que le lecteur puisse vérifier les résultats donnés ci-après.

.3.1 Header File « include.h »

```
#include "stdio.h"
#include "stdlib.h"

#define mot64 unsigned long long int
#define mot32 unsigned long int
#define int32 long int
#define mot16 unsigned int
#define mot08 unsigned char
```

.3.2 Fichiers d'entête « e0light.h »

```
#include <stdio.h>
typedef unsigned char mot08;
typedef unsigned long long mot64;

const mot08 e0_fsm[16][16] = {
{ 0,  0,  0,  1,  0,  1,  1,  1,  0,  1,  1,  1,  1,  1,  1,  2},
{ 5,  4,  4,  4,  4,  4,  4,  7,  4,  4,  4,  7,  4,  7,  7,  7},
{11, 11, 11,  8, 11,  8,  8,  8, 11,  8,  8,  8,  8,  8,  8,  9},
{14, 13, 13, 13, 13, 13, 13, 12, 13, 13, 13, 12, 13, 12, 12, 12},
{ 3,  3,  3,  2,  3,  2,  2,  2,  3,  2,  2,  2,  2,  2,  2,  1},
{ 6,  7,  7,  7,  7,  7,  7,  4,  7,  7,  7,  4,  7,  4,  4,  4},
{ 8,  8,  8, 11,  8, 11, 11, 11,  8, 11, 11, 11, 11, 11, 11, 10},
{13, 14, 14, 14, 14, 14, 14, 15, 14, 14, 14, 15, 14, 15, 15, 15},
{ 1,  1,  1,  0,  1,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  3},
{ 4,  5,  5,  5,  5,  5,  5,  6,  5,  5,  5,  6,  5,  6,  6,  6},
{10, 10, 10,  9, 10,  9,  9,  9, 10,  9,  9,  9,  9,  9,  9,  8},
{15, 12, 12, 12, 12, 12, 12, 13, 12, 12, 12, 13, 12, 13, 13, 13},
{ 2,  2,  2,  3,  2,  3,  3,  3,  2,  3,  3,  3,  3,  3,  3,  0},
{ 7,  6,  6,  6,  6,  6,  6,  5,  6,  6,  6,  5,  6,  5,  5,  5},
```

```
{ 9,  9,  9, 10,  9, 10, 10, 10,  9, 10, 10, 10, 10, 10, 10, 11},
{12, 15, 15, 15, 15, 15, 15, 14, 15, 15, 15, 14, 15, 14, 14, 14}
};

static mot64 e0_r1, e0_r2, e0_r3, e0_r4;
static int e0_state, e0_x, e0_z;
```

.3.3 Procédure de chiffrement

```
#include "e0light.h"

int e0(mot64 K1, mot64 K2, mot08 KA, mot08 * suite, mot64 nbbit)
 {
  unsigned long int i;
  int t;

  /* Register Initialisation */
  e0_r1 = (K1 & 0x1FFFFFFL);
  e0_r2 = ((K1 >> 25) & 0x7FFFFFFFL);
  e0_r3 = (((K1 >> 56) | (K2 << 8)) & 0x1FFFFFFFFLL);
  e0_r4 = (K2 >> 25);

  e0_state = KA;

  for(i = 0; i < nbbit;i++)
   {
    e0_r1 = ((e0_r1 << 1) & 0x1ffffe) | (((e0_r1 >> 7)
           ^ (e0_r1 >> 11) ^ (e0_r1 >> 19) ^ (e0_r1 >> 24)) & 1);
    e0_r2 = ((e0_r2 << 1) & 0x7ffffffe) | (((e0_r2 >> 11)
           ^ (e0_r2 >> 15) ^ (e0_r2 >> 23) ^ (e0_r2 >> 30)) & 1);
    e0_r3 = ((e0_r3 << 1) & 0x1fffffffell) | (((e0_r3 >> 32)
            ^ (e0_r3 >> 27) ^ (e0_r3 >> 23) ^ (e0_r3 >> 3)) & 1);
    e0_r4 = ((e0_r4 << 1) & 0x7fffffffell) | (((e0_r4 >> 38)
            ^ (e0_r4 >> 35) ^ (e0_r4 >> 27) ^ (e0_r4 >> 3)) & 1);
e0_x = ((e0_r1 >> 23) & 1) | ((e0_r2 >> 22) & 2)
          | ((e0_r3 >> 29) & 4) | ((e0_r4 >> 28) & 8);

    e0_state = e0_fsm[e0_state][e0_x];
    t = e0_x ^ (e0_x >> 2);
    t ^= t >> 1;
```

```
    suite[i] = (t ^ (e0_state >> 2)) & 1;
   }
 }
```

.3.4 Programme principal

```
#include "include.h"

#define N 128
#define KA 0         /* Initial memory bits   */

int main(int argc, char * argv[])
 {
  mot64 i, j, i0, i1, i2 , i3, K[2];
  mot32 m;
  mot08 * suite, ka, k;

  K[0] = <------- bits  0 --  63 of secret key
  K[1] = <------- bits 64 -- 127 of secret key

  suite    = (mot08 *)calloc(N, sizeof(mot08));
  suite_sc = (mot08 *)calloc(N, sizeof(mot08));
  suite_ka = (mot08 *)calloc(N + 2, sizeof(mot08));

  e0(K[0], K[1], KA, suite, 128LL);
  printf("Output sequence\n\n");
  for(i = 0L; i < 128;i++) printf("%01d", suite[i]);
  printf("\n\n");
  free(suite);
 }
```

.4 Valeurs de preuve 0-K

Dans cette section, nous donnons quelques unes de clefs produisant des suites de sorties réalisant des propriétés significatives. Les résultats plus complets sont disponibles sur demande. La notation reprend le codage utilisé dans la procédure principale du programme.

```
(69+, 25)
```

```
    K[0] = 0x27D5C62B6FDD0146 K[1] = 0x4B01AAE56E878393

(68+, 27)
    K[0] = 0xFAA732EC24CBBF08 K[1] = 0xF7D90592E202CFE3

(67+, 24)
    K[0] = 0x73CD595AD3FD6A26 K[1] = 0x4E5BB736824EFAC4

(67+, 26)
    K[0] = 0x481AC9D68A265BB6 K[1] = 0x9C49E65F2C5AC7EC

(66+, 26)
    K[0] = 0x19D2C332127ACF17 K[1] = 0x3616434EA1A991A
    K[0] = 0xD15D3CA3C5240B4D K[1] = 0x11BDAC9BE5D608D2
    K[0] = 0x1168C994D63DBEE1 K[1] = 0xA52DB3C47F6E4B78

(66+, 29)
    K[0] = 0x4AA088310330E134 K[1] = 0x886554F41774B5DF

(65+, 26)
    K[0] = 0x499B5A23B09E73C7 K[1] = 0xBF9A060F485F8708

(65+, 27)
    K[0] = 0x49EE7FAEDE74A51B K[1] = 0x9EF861C90E85C6A0

  (65+, 28)
    K[0] = 0xCB9E8BC74B91EA42 K[1] = 0x4575201CFBDC7FF9

(64, 27)
    K[0] = 0x09F51F2AEE52BBCC K[1] = 0x345991408FD0A40B

(63+, 27)
    K[0] = 0x3AF59A1AB3849A22 K[1] = 0xA8F0630AAB90E4EE

(63+, 29)
    K[0] = 0xC98D344092E7B8A6 K[1] = 0x18FFAA9AB4BB0FB2
    K[0] = 0x3395F4E0AA7F2AAA K[1] = 0x7D3C8F1CC1A9FB61
    K[0] = 0x60595B6C3F81FBC7 K[1] = 0x39608B22C62E8C79

(63+, 28)
```

```
    K[0] = 0x0DB55B6143A3DF6A K[1] = 0xC69A087CB6FA29E5

(62, 29)
    K[0] = 0x18C1077579DD290B K[1] = 0x5B672FC8D0CCE243

(62, 27)
    K[0] = 0xF11D6526C305E816 K[1] = 0x35BE571A69C9B6EA

(62+, 29)
    K[0] = 0xC88DDB3D2D6415F4 K[1] = 0xA219615A07B7BFFF
    K[0] = 0xC40BA27939383C32 K[1] = 0xC1692DEF036E7049
    K[0] = 0x9D45CC6215D1E5B3 K[1] = 0x39CB14370AEB1CB2

(61+, 29)
    K[0] = 0xF1F70889D3A6FF5D K[1] = 0x4DD6D71E317B540B
    K[0] = 0x1B9456D34AA3E596 K[1] = 0x9E183710E7B6138B

(62+, 27)
    K[0] = 0x44F646AB3AED19E0 K[1] = 0xC3BC20A780A2BA3E
    K[0] = 0x42461FB9C07F3F9D K[1] = 0x746A780C6A649D6B

(62+, 26)
    K[0] = 0x7B1B5463C802FFB5 K[1] = 0xA3FDF5940264D28B
    K[0] = 0x89E14644C0AD64BB K[1] = 0xC077883C768664D5
    K[0] = 0x33E24602D7A02C18 K[1] = 0xBF3C9A7CD53C865D

(62+, 28)
    K[0] = 0x125D85B3A3353C2A K[1] = 0xA8E12FDAD9269406

(61+, 27)
    K[0] = 0x2FA83A7A4959C2FE K[1] = 0xCCF65606210D32C9

(61+, 26)
    K[0] = 0xF01896F8455DDBD5 K[1] = 0x604AC5B5048A233D

(60+, 23)
    K[0] = 0xB8F7ABBACC30347F K[1] = 0xEEDC60766DAA3F32
```