# THESE de DOCTORAT

---


présentée par

**Eric FILIOL**


pour obtenir le grade de

Docteur de l'Ecole Polytechnique

**Spécialité :** Informatique et Mathématiques Appliquées

---

# Techniques de reconstruction en cryptologie et théorie des codes

---

Soutenue le 28 Mars 2001,

devant le jury composé de :

| | | | |
|---|---|---|---|
| Mr | C. CARLET | Examinateur | Université Paris VIII - Paris |
| Mme | P. CHARPIN | Directrice | I.N.R.I.A. Rocquencourt |
| Mr. | H. GILBERT | Rapporteur | C.N.E.T. - Issy-les-Moulineaux |
| Mr. | D. HACCOUN | Rapporteur | Ecole Polytechnique de Montréal - Canada |
| Mr | G. KABATIANSKI | Examinateur | ITTP, Académie des Sciences de Russie |
| Mr. | S. HARARI | Président | Université de Toulon et du Var |
| Mr. | J.F. MICHON | Rapporteur | Université de Rouen |
| Mr | F. MORAIN | Examinateur | Ecole Polytechnique - France |

# Remerciements

*J'aimerais remercier l'ensemble du projet CODES de l'INRIA pour la stimulante et chaleureuse ambiance qui y règne et la disponibilité de chacun. Les discussions y sont très faciles, variées, ouvertes et toujours très intéressantes.*

*En particulier, je remercie Pascale Charpin, qui m'a accueilli en thèse au sein de son équipe malgré mon éloignement dans la lande bretonne. Elle a su me faire découvrir et aimer les codes correcteurs d'erreurs qui m'ont permis d'avoir souvent une perspective différente dans mon métier de cryptologue militaire. Ses qualités humaines ont fait de ces trois années de thèse un moment privilégié.*

*Je remercierai ensuite Anne Canteaut qui avec patience, quelques boîtes de feutres rouges et rigueur m'a appris à rédiger des textes scientifiques autrement que pour moi-seul. Nous avons eu ensemble des discussions passionnantes et toujours très riches. Grâce à elle, j'ai acquis le goût du travail en équipe qui rendent le métier de chercheur si passionnant et en font une véritable aventure humaine.*

*Je remercie également Caroline Fontaine. Travailler avec elle a été un grand plaisir : sa bonne humeur et sa gentillesse y sont pour beaucoup.*

*Merci aussi à Daniel Augot et Nicolas Sendrier. Ils ont tous deux largement contribué à faire de mon séjour au projet CODES un moment agréable et stimulant. J'ai pu partager leur passion de l'informatique et de Linux.*

*Je ne saurais oublier les autres membres du projet : Hervé Alavoine, Francis Blanché, Pierre Loidreau, Christine Pourcelot, Lancelot Pecquet, Antoine Valembois. Ils m'ont apporté beaucoup et ont fait de mes passages au projet des instants agréables.*

*Je tiens également à remercier les chercheurs étrangers en visite au projet CODES : le regretté Ed Assmus, Hans Dobbertin, Grigory Kabatianski et Victor Zinoviev. Cela a été à la fois un privilège enrichissant et un honneur de pouvoir discuter avec eux et ainsi de pouvoir bénéficier de leur vision personnelle de la théorie des codes ou de la cryptologie.*

*Merci enfin à Christelle Guiziou-Cloître pour sa disponibilité, son aide et surtout sa gentillesse. Je tiens à exprimer mes plus vifs remerciements à Sami Harari pour avoir accepté d'être le président de mon jury de thèse.*

*Je voudrais exprimer ma profonde gratitude à Henri Gilbert, David Haccoun et Jean-Francis Michon pour m'avoir fait l'honneur d'être les rapporteurs de cette thèse. En particulier, je remercie chaleureusement David Haccoun et Henri Gilbert pour l'intérêt qu'il ont porté à mes travaux, la disponibilité et la gentillesse dont ils ont fait preuve. Cela a été à la fois un privilège, un honneur et un plaisir de recevoir leurs critiques, leurs commentaires et leurs conseils.*

*Je remercie également François Morain d'avoir accepté de faire partie de mon jury de thèse. Cela a été pour moi un grand plaisir en même temps qu'un honneur. Puisse-t-il retrouver le sommeil, un moment perturbé par la lecture de ma thèse !*

*Merci également à Claude Carlet d'avoir accepté de figurer dans mon jury de thèse. Son intérêt pour mes travaux sur les fonctions booléennes et ses conseils concernant des études futures dans ce domaine me font lui exprimer la plus grande gratitude.*

*Enfin je voudrais remercier Grigory Kabatianski d'avoir accepté dêtre examinateur dans mon jury de thèse. Cela a été un très grand honneur pour moi et surtout un grand plaisir : quiconque l'a rencontré une fois ne peut manquer d'attester sa gentillese et son humour.*

*Cette thèse a été financée par le ministère de la Défense, armée de terre, alors que j'étais en poste aux Ecoles de Coëtquidan. Je tiens tout spécialement à remercier le colonel Max Mayneris. Sans lui, cette thèse n'aurait jamais eu lieu. Il était convaincu que l'armée de terre se devait de mener des recherches dans le domaine de la sécurité des communications et des informations. Transmetteur enthousiaste, il a su me faire partager sa passion pour son arme. Dispensant gentillesse sans compter, ses qualités intellectuelles et surtout humaines m'ont guidé pendant ces années de thèse. J'espère qu'il profite pleinement de sa retraite pyrénéenne.*

*Je remercie également le lieutenant-colonel Le Clainche, le chef de bataillon Vitasek, le chef de bataillon Pierret, le major Marsan, l'adjudant-chef Asper, l'adjudant-chef Hennequin, l'adjudant Carré. Tous ont largement contribué à la réussite de cette thèse.*

*L'écriture d'une thèse est une activité longue et fastidieuse. Grâce au lieutenant-colonel Porte, commandant l'École du Corps Technique et Administratif, j'ai pu m'aérer l'esprit et partager le plaisir d'activités physiques et rustiques avec les élèves officiers des 23ème (Base Logistique de Na-San) et 24ème promotions (Général Doumenc). Son dynamisme, son sens de l'hu-*

*main et sa bonne humeur ont été là au bon moment.*

*Merci à tous ceux que je n'aurais pas dû oublier.*

*Enfin je remercie mon épouse Laurence et mon fils Pierre. Eux savent plus que quiconque la difficulté de faire une thèse. Ils ont supporté mon absence et quelquefois ma présence....*

# Introduction

*« Le véritable maître d'une chose n'est pas celui qui la possède mais celui qui peut la détruire. »*

Frank L. Herbert - Dune.

*« Le seul véritable pouvoir est celui qui s'exerce sur les lois de la Nature, elles-mêmes gouvernant aux hommes, et non pas le pouvoir sur les hommes. Le premier n'est possible que grâce à l'amour et la confiance en Dieu. Le second n'est possible que par la lacheté des hommes. »*

Poète Persan

Lors de la transmission d'une information, deux préoccupations peuvent, simultanément ou non, affecter la conception de la chaîne de transmission.

La première consiste à vouloir protéger le canal utilisé contre toute atteinte **non malveillante** de l'information vis-à-vis du canal utilisé. Il s'agit là de veiller à la *sureté* de l'information . Cet aspect de la protection de l'information concerne presque toute la chaîne (voir schéma 1) dans son acception la plus large (transmission au sens usuel du terme, stockage sur disque optique de type CD, disque dur, mémoire,... le support jouant ici le rôle de canal, l'écriture et la lecture des données étant représentées par, respectivement, la source et la destination de la chaîne). Cet impératif de sûreté concerne non seulement l'information proprement dite, mais également le support, les deux étant indissociablement liés. En effet, cette sûreté dépendra fortement de ce couple. Elle est assurée par les nombreux et puissants objets de la théorie des codes : les codes correcteurs d'erreurs.

Un code est une convention *publique*, largement diffusée, (par exemple le code Morse, les codes correcteurs des disques compacts, le code ascii,...)

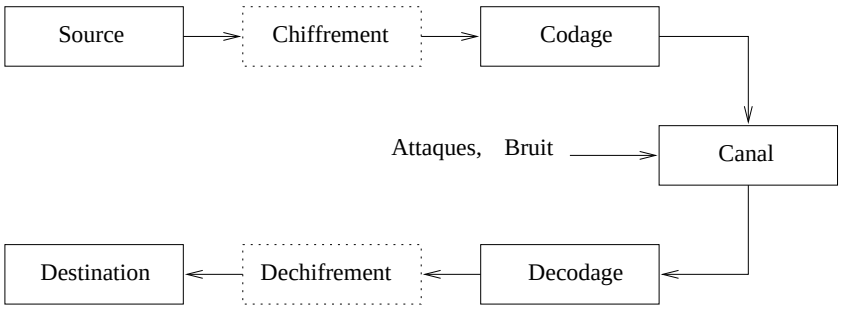


FIGURE 1 – Chaîne de transmission

assurant la protection de l'information pour un type de canal donné (essentiellement le type et les paramètres de bruit sont considérés). Le bruit affectant la chaîne de transmission peut être de nature très variable : défaillances (pannes) dans des mémoires vives ou de masse, perturbations ionosphériques, rayonnements divers lors d'une transmission hertzienne, rayures sur un CD,...

Le principe général de la protection par codage est l'ajout de redondance, c'est à dire d'information supplémentaire, la plus optimale possible, en terme de coût, de volume,..., autant de contraintes dépendant encore une fois largement de la nature du canal mais aussi de la nature de l'information elle-même (compressée ou non par exemple). Ainsi une information sera codée avec un code de Reed-Solomon si le canal est un disque compact, alors qu'un code convolutif sera préféré pour une transmission de type G.S.M.

La seconde préoccupation concerne la *sécurité* de l'information. Elle concerne la protection contre toute atteinte **malveillante** visant sa confidentialité, son intégrité, son authentification,.... C'est ici le règne de la cryptologie dont la fonctionnalité la plus connue est la confidentialité assurée par le chiffrement. D'une manière générale, un mécanisme cryptologique est une convention comportant des éléments les plus *secrets* possibles (les clés) sur lesquels repose toute la sécurité.

Cette seconde préoccupation, bien antérieure historiquement à la première, ne concerne ici que l'information elle-même et ne dépend pas du canal (toutefois la mise en pratique peut révéler des failles d'implémentation importantes mais elles ne concernent plus la chaîne en elle-même mais plutôt la sécurité informatique).

Dans la pratique, ces deux aspects sont assez souvent conjugués. Le meilleur exemple est celui des communications militaires qui après chiffrement sont codées par un code convolutif.

Dans un contexte d'attaque, il s'agit alors d'accéder **illégitimement** à l'information transmise. D'abord interceptée, elle doit ensuite subir un traitement : décodage et/ou décryptement. Cette phase ne serait globalement pas différente de celle réalisée par le ou les destinataires légitimes si une différence notable ne distinguait pas les deux situations.

Là où le destinataire légitime connaît les transformations subies par l'information d'origine (codage et chiffrement), et donc également les transformations inverses à opérer pour accéder à cette information, l'attaquant, lui, les connaît rarement, ce qui lui complique singulièrement la tâche, voire la rend impossible. Autrement dit, l'un connaît les algorithmes (code et chiffre) et pas l'autre.

Dans le cas du codage, bien que par définition convention publique, le code utilisé est très souvent inconnu de l'attaquant. Il est possible de penser que cela est susceptible d'augmenter la sécurité de la chaîne de transmission en rendant tout décodage impossible. Il peut en effet sembler à première vue difficile de décoder sans connaitre le code utilisé.

Dans le cas du chiffrement, cette impression semble encore plus pertinente, bien que, selon les lois de Kerckhoffs [103], il soit toujours supposé lors de l'évaluation d'un système de chiffrement, que l'algorithme est connu de l'attaquant. On peut cependant accepter le sentiment qu'en l'absence de toute connaissance sur l'algorithme, l'idée de cryptanalyse spécifique n'a pas de sens et que dissimuler l'algorithme renforce la sécurité.

Or il faut reconnaître que cette situation est de loin la plus fréquente. En 1982, l'affaire Hans Bühler [81, 116, 160, 170], éclata. Les clients (et en particulier la Lybie, l'Iran, l'Iraq,...) de la célèbre firme suisse Crypto-AG [1] eurent la preuve [4] que les algorithmes vendus étaient totalement connus et contrôlés par la N.S.A. , et que des fonctionnalités cachées y avaient été introduites afin d'en permettre une attaque moins difficile. Cela eut comme conséquence que ces pays se mirent à fabriquer leurs propres machines de chiffrement, les algorithmes devenant inconnus des occidentaux.

Une approche alors possible pour l'attaquant est celle consistant à retrouver le code ou le chiffrement utilisé, à partir du seul matériau à lui disponible en relativement grande quantité : la transmission interceptée. Cette approche est elle viable et techniquement réalisable ?

Ce problème ainsi posé n'a pratiquement jamais fait l'objet de publications ou d'études ouvertes. Pour les codes, quelques ébauches de solutions

1. Crypto-AG est le fournisseur de plus de 130 pays en systèmes de chiffrement gouvernementaux, profitant de sa position officielle de neutralité, notamment vis à vis de l'OTAN

existent et se limitent à la détection de la nature du code [137] ou à la reconstruction d'instances simples de codes convolutifs dans le cas d'un canal sans bruit [145]. Plus récemment, le cas des codes en blocs a été envisagé plus en profondeur [165].

Pour la cryptologie, la seule mention d'une telle approche concerne la machine PURPLE , utilisée par les japonais pendant la seconde guerre mondiale et dont les cryptanalystes américains ont du reconstruire l'algorithme avant toute cryptanalyse, à partir du seul trafic chiffré intercepté. La technique employée n'a jamais été publiée [102].

L'objet de cette thèse est de présenter des techniques opérationnelles permettant de résoudre certaines parties de ce problème, dans des cas précis, démontrant ainsi que dissimuler les algorithmes (codes ou systèmes de chiffrement) n'ajoute en rien à la sécurité de la chaîne de transmission. L'approche générale est de partir de suppositions faites sur les systèmes utilisés. Elles sont généralement étayées par des renseignements complémentaires sur la chaîne de transmission considérée (d'origine ELINT [2], COMINT [3], HUMINT [4],...).

La seconde étape est de rechercher, puis d'exploiter une "signature" caractéristique du système et de ses paramètres. La présence de cette signature dans le matériau brut intercepté, permettra alors de conclure sur la nature interne de tout ou partie du système. Afin de garantir l'effectivité des méthodes, seul le message (codé et/ou chiffré) intercepté a été utilisée à l'exclusion de toute autre donnée (le message d'origine par exemple).

Dans une première partie, j'exposerai la solution que j'ai apportée dans le cas des systèmes de chiffrement par flot. Ces systèmes sont essentiellement employés par le monde gouvernemental (Défense, Intérieur, Affaires Etrangères,...) et industriel (commandes de satellites par exemple). Leurs primitives de base sont principalement des registres à décalage à rétroaction linéaire et des fonctions booléennes possédant des propriétés cryptographiques fortes. Le polynôme employé pour la rétroaction d'un registre à décalage fournit une première famille de signatures quand on considère les éléments de faibles poids de l'idéal (principal) polynomial qu'il génère. La signature des fonctions booléennes utilisées est déterminée par leur spectre de Walsh et certaines propriétés structurelles. En particulier, j'exposerai pour ces fonctions une nouvelle caractérisation de leur forme algébrique normale, par des objets combinatoires comme les designs et des familles intersectantes. J'ex-

2. *Electronic Intelligence* ou renseignement électronique
3. *Communication Intelligence* ou renseignement sur les communications
4. *Human Intelligence* ou renseignement de source humaine

poserai également une nouvelle attaque des systèmes de chiffrement par flot, basée sur une faiblesse structurelle des registres à décalage. Je définirai alors une nouveau critère de résistance.

Je présenterai également dans cette partie, une nouvelle attaque des systèmes de chiffrement à flot. Le principe est de considérer une sous-suite de la suite de sortie d'un registre à décalage, obtenue par décimation régulière. Quand cette dernière est convenablement choisie, la sous-suite est alors générée par un registre plus court, que l'on peut alors attaquer de façon moins coûteuse pour retrouver l'initialisation du registre. Je présenterai alors un nouveau critère de résistance pour cette attaque.

Dans la seconde partie, j'exposerai la solution que j'ai apportée dans le cas des codes convolutifs, d'abord simples, puis poinçonnés. Elle permet une reconstruction effective de ces codes, quel que soit leur taux et pour des communications présentant des niveaux de bruit relativement importants. Ces codes sont également très utilisés dans les transmissions militaires ou gouvernementales, dans les télécommunications par satellites et plus récemment dans la téléphonie mobile. Ils servent en particulier à protéger du bruit du canal, la plupart des transmissions chiffrées.

# Première partie

# Reconstruction en Cryptologie

# Chapitre 1

# Introduction à la cryptologie

La cryptologie (ou science du secret) regroupe deux branches : la *cryptographie* et la *cryptanalyse*.

La cryptographie s'occupe de l'aspect défense (la "cuirasse"). Son activité est l'étude des objets mathématiques utilisés pour construire des systèmes cryptologiques, leurs propriétés en vue d'une résistance maximale, vis-à-vis des attaques connues, leur mise en œuvre et leur gestion. La cryptanalyse, quant à elle, se préoccupe de l'aspect attaque (la "lance"). Elle a pour but de développer des techniques permettant de casser ou au minimum d'affaiblir suffisamment les systèmes étudiés. Ces techniques doivent, en final, être plus efficaces que l'essai de toutes les valeurs possibles des éléments secrets du système.

Cryptographie et cryptanalyse sont donc intimement liées et l'une ne peut être pratiquée sans la connaissance des résultats de l'autre. Toutefois, le point de vue du cryptanalyste est plus confortable que celui du cryptographe. En effet, il est facile de prouver que l'on a cassé tel ou tel système. Il suffit d'en apporter la preuve, de faire une démonstration. Le cryptographe, lui, quand il conçoit un système, ne peut apporter la preuve que son système est véritablement sûr que vis-à-vis de techniques de cryptanalyse connues. Or toutes ne sont pas publiées, loin s'en faut.

Je vais dresser dans ce chapitre un panorama succinct des fonctionnalités offertes par la cryptographie et aborder quand cela sera utile quelques aspects de la cryptanalyse. Une présentation plus générale et didactique pourra être trouvée dans l'ouvrage de G. Brassard [15] et une passionnante présentation historique de la cryptologie sera trouvée dans l'ouvrage de D. Kahn [102]. Des présentations plus techniques pourront être trouvées dans [150, 127].

## 1.1 Le chiffrement

A tout seigneur tout honneur, la première (historiquement et en importance) fonctionnalité cryptographique, est la *confidentialité* réalisée par le *chiffrement*. Un *système de chiffrement*, mécanisme assurant la confidentialité, est un système permettant de cacher le contenu d'un message clair par des opérations consistant soit à remplacer (*substitutions*) les symboles du messages, soit à les mélanger (*permutations*). La taille et la nature des alphabets dépendra des systèmes.

Ces opérations qui peuvent être combinées, doivent rendre toute connaissance du message impossible à toute personne ne disposant pas d'éléments secrets, les *clefs*, ayant servi à déterminer ces opérations. On parlera alors de chiffrement et de *déchiffrement* pour l'opération inverse, s'agissant des personnes légitimement autorisées à accéder au message. On parlera de *décryptement* s'agissant de l'obtention de ces clefs par l'attaque. Le système doit donc être suffisamment résistant à toute attaque permettant de retrouver ces clefs

Selon les lois de Kerckhoffs [103], il est toujours supposé que les opérations de transformation (chiffrement et déchiffrement) sont publiques et que la force du système doit uniquement reposer sur les éléments secrets. En effet, il est plus facile de changer une clé, lorsque compromise, qu'un système tout entier. Toutefois, cette hypothèse est rarement respectée. Pourquoi faciliter la tâche de l'ennemi ? Cacher la nature des opérations complique le travail du cryptanalyste, mais permet quelquefois, on peut le supposer, de cacher certaines fonctionnalités, permettant en retour une attaque, quand le système se retrouve "en face" (après avoir été vendu à l'adversaire par exemple). C'est ce problème qui motive la première partie de cette thèse.

Deux familles de systèmes existent selon que les clefs sont différentes ou non pour le chiffrement et le déchiffrement.

### 1.1.1 Systèmes symétriques

Ces systèmes sont encore appelés systèmes à clef secrète. Historiquement et en importance, ce sont les systèmes les plus répandus. D'abord systèmes manuels depuis l'aube des temps (substitutions simples, multiples, transpositions,... voir le livre du colonel M. Givierge [72]), ces systèmes sont maintenant devenus beaucoup plus complexes et reposent, pour leur principe et leur sécurité, sur une théorie mathématique importante : la théorie de l'information dont les fondements ont été posés par C. Shannon [152].

Le nom de symétrique vient du fait qu'une seule clef, partagée par

l'émetteur et le ou les destinataires, est nécessaire pour réaliser chiffrement et déchiffrement. Le nom de clef secrète exprime que toute la sécurité du message concerné repose sur cette clé. Deux familles de systèmes existent aujourd'hui [1] : les systèmes *par flot* et *par blocs*.

Les systèmes par flot sont des systèmes où les symboles du message clair (usuellement des bits) sont chiffrés un par un, à la volée. Le plus souvent, le message est combiné, par addition modulo 2, à une suite aléatoire. Le système le plus sûr, celui dont on peut (troisième théorème de Shannon [152] sur le secret parfait) prouver la sécurité absolue (lorsque correctement utilisé) est la *bande aléatoire une fois* (*one time pad* pour les anglais). Toute la sécurité repose sur le caractère parfaitement aléatoire de la suite utilisée. Mais le principal inconvénient vient de la nécessité de disposer d'une suite (le secret dans ce cas) aussi longue que le message, et de suites différentes pour chaque message différent. Cela pose d'énormes problèmes de gestion et de génération de clé. Ce système est réservé à des trafics de niveau stratégique (téléphone rouge par exemple).

Dans la pratique, on utilise plutôt des suites *pseudo-aléatoires*, générées de façon déterministe par un automate, à partir d'un secret commun court, qui lui peut être généré et échangé beaucoup plus facilement. Ils constituent l'importante famille des générateurs pseudo-aléatoires qui représentent la très grande majorité des systèmes opérationnels dans le monde. Ces systèmes seront présentés en détail dans le chapitre 2.

L'autre famille de systèmes symétriques est celle des systèmes de chiffrement par blocs. Ces systèmes sont nés en 1973 avec l'article de Feistel [53] et ont connu le succès avec le *Data Encryption Standard* ou *D.E.S.* [46]. Depuis leur succès n'a cessé d'augmenter, en particulier avec le concours A.E.S. ou *Advanced Encryption Standard* [1] visant à remplacer le D.E.S. Ces systèmes sont surtout utilisés dans le monde industriel et commercial. Leur emploi est encore très limité pour le chiffrement gouvernemental.

La raison en est que la "théorie des systèmes par blocs" est encore à naître. L'extrême difficulté à les décrire, ne serait-ce que d'un point de vue combinatoire, les rend suspects. La dissimulation de fonctions cachées permettant une cryptanalyse facile pour celui qui les connait a souvent été évoquée [150]. Leur sécurité ne repose sur aucune théorie mathématique, mais uniquement sur l'expérimentation et l'empirisme.

Le principe des schémas par blocs est le suivant : un bloc de message ($n = 64$ ou 128 bits) est chiffré d'un seul tenant par l'algorithme. En fait,

1. Les systèmes manuels d'autrefois, substitutions et permutations, sont maintenant considérés, en fait, comme un cas particulier de systèmes par blocs

chaque clef fixe une permutation sur l'ensemble de blocs de taille $n$. La taille de la clé est de 64 (D.E.S.) ou 128, 192 et 256 bits (A.E.S.). Autrement dit, le chiffrement par blocs, pour une clef donnée, n'est en fait qu'un simple système de substitution sur un alphabet de taille $2^n$.

D'un point de vue technique, il s'agit d'obtenir une suite inextricable de calculs, certes simples considérés isolément (permutations, substitutions, ...), mais empêchant toute vision globale de l'algorithme, lorsque finalement combinés.

Deux critères sont recherchés [153]. La *diffusion* qui fait dépendre le plus vite possible tout bit interne et tout bit du bloc de cryptogramme, des bits de clef et de message. Ainsi aucune redondance du message ne se retrouvera dans le chiffré. Cet effet est généralement obtenu grâce aux opérations de transposition [127, p. 20].La *confusion* qui doit rendre la relation clef/chiffré la plus complexe possible. Elle est généralement réalisée par des substitutions[127, p. 20].

Pour cela la plupart de ces systèmes (D.E.S., SAFER, IDEA, ...) utilisent la structure dite itérative. Pour chaque bloc de message, on itère $r$ fois une fonction interne $F$. Pour chaque tour $i$, la fonction $F$ est paramétrée par une sous-clef $K_i$ $(1 \leq i \leq r)$ de la clef de base $K$, obtenue par un algorithme de cadencement de clef initial. Le tour est alors noté $F_{K_i}$. La famille la plus

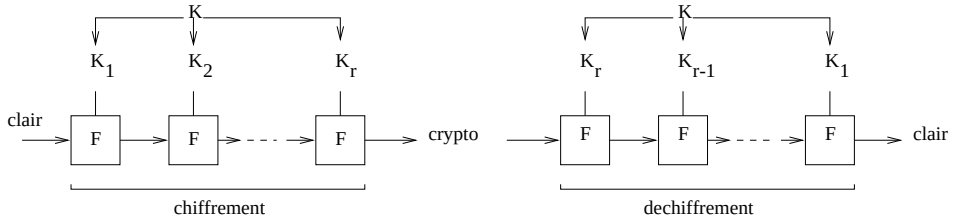


FIGURE 1.1 – Principe d'un chiffrement itératif par blocs

importante de ces systèmes est celle dite des *chiffrements de Feistel*.

**Définition 1** *Un* chiffrement de Feistel *est un chiffrement itératif opérant sur des blocs de* $2n$ *bits. La fonction itérée* $F$ *est définie par*

$$\begin{aligned} F \quad : \quad & \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n \times \mathbb{F}_2^n \\ & (L_{i-1}, R_{i-1}) \mapsto (L_i, R_i) \end{aligned}$$

*avec* $L_i = R_{i-1}$ *et* $R_i = L_{i-1} \odot f(R_{i-1}, K_i)$ *où* $\odot$ *est une loi de groupe. Quelle que soit la fonction* $f$ *utilisée, un chiffrement de Feistel est inversible. Pour*

*déchiffrer, il suffit d'utiliser le même processus à r tours en inversant l'ordre des sous-clefs $K_i$ (la fonction $F$ étant involutive).*

Le schéma 1.2 illustre cette définition. Quelles sont les attaques connues

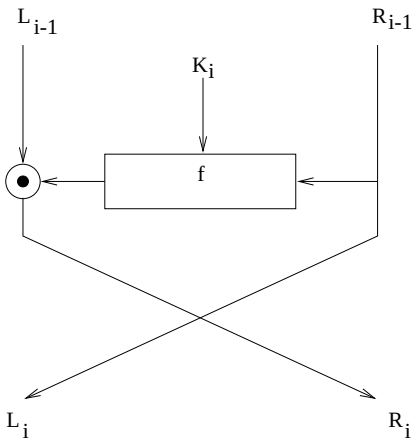


FIGURE 1.2 – Schéma de Feistel

pour les systèmes par blocs ? Il y en a essentiellement deux. Toutes reposent sur la connaissance d'un certain nombre de couples clair/cryptogramme.

### Cryptanalyse différentielle

Cette attaque a été introduite par E. Biham et A. Shamir en 1991 [10, 11, 12, 13]. C'est une attaque à clair choisi ou à clair connu.

Son principe général est le suivant : on soumet au chiffrement des couples de blocs de clair dont la différence $\alpha$, calculée en général par addition modulo 2 bit à bit des deux blocs (pour une généralisation à d'autres opérations voir [87]), est fixée. Cette attaque exploite alors le fait que la fonction itérée $F : (X, K) \mapsto Y = F_K(X)$ d'un chiffrement itératif par blocs présente généralement des faiblesses cryptographiques : la connaissance d'un couple de sorties $(Y, Y')$ de $F$ et de la différence $\Delta X$ de leur entrée donne une information sur quelques bits de la clef $K$. L'existence d'un biais (toujours présent de façon plus ou moins exploitable) dans la répartition des valeurs des différences de sorties $\Delta Y$ en fonction des différences en entrée $\Delta X$ permet d'accéder à cette information. Soit une clé est plus probable qu'une autre (existence d'un pic dans la distribution) soit elle est impossible (valeur nulle dans la distribution ; la différentielle est dite alors impossible).

On soumet donc des couples $(Y(0), Y'(0) = Y(0) \oplus \alpha)$ au chiffrement, et on estime la valeur $\Delta Y(r-1)$ des différences de leur chiffré après $r-1$ tours. Si $\Delta Y(r-1) = \beta$ avec une probabilité élevée (en tout cas différente de la répartition uniforme), c'est-à-dire que $F$ présente une faiblesse de distribution, alors il devient possible de retrouver quelques bits de la sous-clef $K_r$, utilisée au tour $r$.

**Définition 2** *Une* différentielle *au tour $i$ est un couple de différences $(\alpha, \beta)$, où*

$$\alpha = Y(0) \oplus Y'(0) \quad \text{et} \quad \beta = Y(i) \oplus Y'(i)$$

*On définit la probabilité de la différentielle par*

$$P[\Delta Y(i) = \beta | \Delta Y(0) = \alpha]$$

*où $Y(0)$ est une variable aléatoire uniformément distribuée.*

L'algorithme général de la cryptanalyse différentielle est le suivant : La com-

1. déterminer une différentielle $(\alpha, \beta)$ au tour $r-1$ dont le biais est élevé.
2. – choisir aléatoirement un bloc clair $X$ et soumettre $X$ et $X \oplus \alpha$ au chiffrement.
   – déterminer toutes les valeurs possibles $k$ de $K_r$ en supposant que $\Delta Y(r-1) = \beta$.
   – incrémenter le compteur correspondant aux apparitions de $k$.
3. itérer l'étape 2 jusqu'à ce que l'une des valeurs $k$ de $K_r$ apparaisse plus souvent que les autres.

TABLE 1.1 – Algorithme de la cryptanalyse différentielle

plexité de cette attaque est égale au nombre de messages clairs choisis qu'il faut soumettre au chiffrement pour trouver la sous-clef $K_r$.

**Proposition 1** *[107] Soit $P_{\max}$ la probabilité de la meilleure différentielle au tour $r-1$. On a*

$$P_{\max} = \max_{\alpha \neq 0, \beta \neq 0} (P[\Delta Y(r-1) = \beta | \Delta Y(0) = \alpha]$$

*Le nombre de blocs de clair choisis à soumettre au chiffrement pour retrouver $K_r$ c'est-à-dire la complexité de l'attaque, doit être au moins*

$$\frac{2}{P_{\max} - \frac{1}{2^n - 1}}$$

*où $n$ est le nombre de bits dans un bloc de clair.*

La cryptanalyse différentielle est impossible si toutes les différentielles au tour $r-1$ ont une probabilité proche de $\frac{1}{2^n-1}$. Pour cela, il faut que quel que soit $\alpha \in \mathbb{F}_2^n$ non nul, $\Delta Y(r-1)$ soit uniformément distribuée sur $\mathbb{F}_2^n - \{(0,0,\ldots,0)\}$.

### Cryptanalyse linéaire

Cette attaque a été introduite par M. Matsui [124] d'après les travaux de A. Tardy-Corfdir et H. Gilbert [163].

Son principe général est d'approcher la fonction itérée $F : (X,K) \mapsto F_K(X)$ par une fonction linéaire ; cela revient à trouver $\alpha, \beta$ et $\delta$ tels que

$$< \beta, F_K(X) >=< \alpha, X > \oplus < \delta, K >$$

pour de nombreux blocs clairs $X$.

En chaînant de telles équations, on aboutit à une approximation linéaire des $r$ tours de l'algorithme de chiffrement, de la forme :

$$< \alpha, Y(0) > \oplus < \beta, Y(r) > \oplus < \delta, K >= 0$$

En pratique, on utilise le plus souvent une approximation sur les $r-1$ premiers (ou derniers) tours afin d'obtenir une information sur la sous-clé du dernier (respectivement premier) tour.

La cryptanalyse linéaire a alors pour algorithme général, applicable à tous les systèmes itératifs par blocs, l'algorithme présenté en figure 1.2. Encore une fois, le taux de succès de cet algorithme dépend du nombre de couples clair/chiffré $N$.

**Proposition 2** *[124] Soit $N$ le nombre de couples clair/chiffré connus (on suppose que les blocs de clairs sont uniformément distribués). Soit $p$ la probabilité que $< \alpha, Y(0) > \oplus < \beta, Y(r) > \oplus < \delta, K >= 0$. Alors le taux de succès de l'algorithme est :*

$$\int_{-2\sqrt{N}|p-\frac{1}{2}|}^{+\infty} \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right) dx$$

Soit la relation $< \alpha, Y(0) > \oplus < \beta, Y(r) > \oplus < \delta, K >= 0$
satisfaite pour l'ensemble des blocs de clair avec une probabilité $p$
avec $|p - \frac{1}{2}|$ élevée.
Pour $N$ couples clairs/chiffrés $(Y(0), Y(r))$ :

- calculer $T = |\{Y(0), < \alpha, Y(0) > \oplus < \beta, Y(r) >= 0\}|$.
- si $T > \frac{N}{2}$ alors
  - $< \delta, K >= 0$ si $p - \frac{1}{2} > 0$.
  - $< \delta, K >= 1$ sinon
- si $T < \frac{N}{2}$ alors
  - $< \delta, K >= 1$ si $p - \frac{1}{2} > 0$.
  - $< \delta, K >= 0$ sinon

TABLE 1.2 – Algorithme de la cryptanalyse linéaire

### 1.1.2 Les systèmes asymétriques

Introduits en 1976 par W. Diffie et M. Hellman [47], ces systèmes utilisent deux clefs, l'une secrète (ou encore privée) servant à déchiffrer, l'autre publique pour le chiffrement. L'existence de cette clef publique, généralement disponible dans un annuaire [2] ou distribuée par le détenteur de la clé secrète à ses interlocuteurs sous la forme d'un certificat, permet de simplifier notablement la gestion des clefs et supprime la mise en place d'un secret partagé entre les deux protagonistes de la communication. C'est pourquoi le terme de cryptologie à clef publique est encore employé pour désigner cette nouvelle branche de la cryptologie.

Le principe général de ces systèmes est le suivant. L'émetteur chiffre son message avec la clef publique du destinataire, disponible pour tous dans un annuaire. Le destinataire, à l'aide de sa clef secrète, qui a été conçue en relation avec la clef publique, déchiffrera le message qui lui est adressé et que lui seul peut donc déchiffrer.

L'existence même de ce mécanisme asymétrique repose sur l'existence de fonctions spéciales, appelées *fonctions à sens unique avec trappes*. En effet, le calcul doit être facile dans un sens (chiffrement) et doit être impossible (du

2. Il est intéressant de noter, comme l'a mis en lumière la récente affaire concernant la sécurité de la clef publique de la carte bancaire française, que la manie inconsidérée du secret et la méconnaissance de la théorie, tend peu à peu à faire de cette clef publique une clef secrète, niant ainsi l'intérêt même de ces systèmes

moins en un temps raisonnable) à inverser (décrypter) sans une connaissance particulière rendant cette inversion facile (déchiffrement).

Ces fonctions sont fournies par des problèmes mathématiques réputés difficiles (factorisation, calcul du logarithme dans un corps fini, problèmes de la théorie des graphes,...). L'exemple le plus célèbre est celui du systèmes R.S.A. créé par D. Rivest, A. Shamir et L. Adleman en 1978 [144], basé sur la difficulté de la factorisation. Dans ce système, la clef secrète est constituée de deux grands nombres premiers (environ 512 bits chacun), et sa clef publique est constituée du produit de ces deux nombres. Le mécanisme général est présenté dans la figure 1.3. Le problème concernant ces systèmes est double :

L'émetteur $E$ veut chiffrer le message $m$ pour le destinataire $D$.
- **Chiffrement** : $E$ calcule avec la clef publique $(e, n)$ de $D$ $c = m^e \bmod n$.
- Le chiffré $c$ est envoyé.
- $D$ possède une clef secrète $d$ telle que $e.d \equiv 1 \bmod (p-1)(q-1)$ où $n = p.q$.
- $D$ déchiffre $c$ en calculant $c^d = m \bmod n$.

TABLE 1.3 – Chiffrement R.S.A.

- Ce sont des systèmes extrêmement lents. L'un des meilleurs de ces systèmes, R.S.A., reste encore mille fois moins rapide que le D.E.S. Leur usage est donc limité à d'autres fonctionnalités cryptologiques.
- Leur principe est basé sur la théorie de la complexité. Cette dernière suppose qu'il existe des problèmes que l'on ne peut résoudre en temps polynomial (la factorisation par exemple). Autrement dit, à une petite augmentation de la taille des données en entrée, correspondra une variation exponentielle des calculs nécessaires pour résoudre le problème utilisant ces données (pour une présentation exhaustive de cette théorie voir l'excellente monographie de Papadimitriou [135]).
  Or cette supposition (ou son contraire) n'a jamais été démontrée. Ce qui rend l'existence de ces fonctions hypothétique. De plus, toute résolution d'un problème difficile entraîne *ipso facto* celle de tous les autres. Toute la cryptologie asymétrique alors s'écroulerait.

## 1.2 L'authentification et la signature

La cryptologie à clef publique semble représenter un modèle idéal. En effet, elle permet théoriquement à deux protagonistes de communiquer sans qu'ils aient eu besoin de se rencontrer pour échanger des clefs. Malheureusement les choses ne sont pas aussi simples que cela. En particulier se pose le problème de l'*authentification*.

Lors du chiffrement R.S.A. (par exemple) le destinataire du message n'a aucun moyen de savoir qui lui envoie ce message. On définira cette fonctionnalité cryptologique de la manière suivante :

**Définition 3** *On dit que le destinataire $D$ authentifie l'émetteur $E$ si et seulement si*

1. *$E$ peut prouver à $D$ qu'il est bien $E$.*
2. *$E' \neq E$ ne peut prouver à $D$ qu'il est $E$.*

Il faut noter qu'authentifier $E$ revient à authentifier les messages qu'il envoie. Différents mécanismes, appelés *protocoles d'identification*, sont utilisés. Le plus simple consiste en un simple mot de passe (pour une connexion à une session Unix, pour utiliser sa carte bleue,...). Le problème majeur est que celui qui s'authentifie donne son secret. Une autre solution est de prouver de façon interactive son identité à un interlocuteur, en donnant la preuve de la détention d'un secret sans jamais communiquer ce dernier. Ce type de protocole est dit *à apport de connaissance nul (zero-knowledge)* [73]. Le plus célèbre de ces protocoles est celui de Fiat-Shamir [55]. Comme R.S.A., il repose sur un problème de factorisation.

Un autre problème peut survenir lors d'un chiffrement réalisé avec un système à clef publique : $E$ envoie un message à $D$ puis se rétracte et prétend qu'il ne l'a pas fait. Comment $D$ peut-il prouver que $E$ a bien envoyé ce message ? Un cas concret consisterait pour $E$ à passer un ordre de vente en bourse au banquier $D$, puis voyant que la transaction se révèle désavantageuse (le cours a subi une grande variation par exemple), il prétend que ce n'est pas lui qui a envoyé ce message, que sa *signature* a été contrefaite.

La signature électronique, permet d'éviter ce genre de problème.

**Définition 4** *On dit que le message $M$ (clair ou chiffré) est signé par $E$ si :*

1. *$E$ peut prouver à $D$ qu'il en est l'auteur.*
2. *$E' \neq E$ ne peut prouver à $D$ qu'il en est l'auteur.*

3. *D peut prouver à un tiers C que E est bien l'auteur (et le seul) du message M.*

*La signature cumule donc l'authentification avec le report de conviction. Si $C = E$, on parle de* non répudiation *par E.*

En pratique, la signature électronique consiste à adjoindre au texte clair ou chiffré, un petit nombre de bits qui dépendront simultanément du message lui-même et de son auteur (certificat). Ainsi sont garantis à la fois l'intégrité du message et l'identité de son auteur.

Le schéma 1.4 fournit un exemple de signature réalisée avec R.S.A. Afin

L'émetteur $E$ doit signer le message $m$ pour le destinataire $D$.
- $E$ possède une clef secrète $d$ telle que
  $e.d \equiv 1 \bmod (p-1)(q-1)$ où $n = p.q$.
  ($p$ et $q$ : deux nombres premiers d'au moins 512 bits chacun).
- Il calcule $s = m^d \bmod n$ et envoie $(m, s)$.
- $D$ vérifie la signature avec la clef publique $(e, n)$ de $E$
  en calculant $s^e = m \bmod n$.

TABLE 1.4 – Signature R.S.A.

de lutter contre la lenteur des systèmes de signature à clef publique, on réalise un condensé $h(m)$ du message (taille 160 bits) et on signe ce condensé.

## 1.3 L'intégrité

L'*intégrité* est une autre fonctionnalité importante de la cryptologie. Supposons qu'une banque envoie un message à une de ses succursales, demandant de virer sur le compte du client $C$ la somme de 10.000 francs. $C$, pirate à ses heures, intercepte le message en cours de communication et modifie le message en remplaçant 10.000 par 1.000.000.

Si le dispositif de signature est limité (pas de vérification de l'intégrité du message), la succursale ne détectera pas la modification. L'intégrité du message a été attaquée.

Pour remédier à ce genre d'attaque, on utilise les *fonctions de hachage* .

**Définition 5** *Une application $H : \mathbb{F}_2^\infty \mapsto \mathbb{F}_2^n$ est une fonction de hachage si*

1. *Calculer $H(x)$ à partir de $x$ est facile.*

2. *Calculer $x$ à partir de $H(x)$ est difficile.*
3. *$H$ est* sans-collision *: connaissant $x$ et $H(x)$, il est difficile de trouver $x' \neq x$ tel que $H(x) = H(x')$.*

Un algorithme de hachage est généralement composé d'une fonction de compression qui transformera un texte $x$ de longueur indéfinie en un bloc de longueur $l$ et d'une fontion de hachage proprement dite qui à un bloc de $l$ bits associera un condensé $H(x)$ de $n$ bits. Actuellement les deux algorithmes les plus utilisés, Ripemd160 [51] et SHA-I [63], fonctionnent avec une longueur de condensé (ou empreinte numérique) $n = 160$ et travaillent sur des blocs de message de $l = 512$ bits.

# Chapitre 2

# Le chiffrement par flot

## 2.1 Introduction

Le chiffrement par flot (*Stream encryption*) forme la classe la plus ancienne et la plus importante [1] du chiffrement moderne (depuis 1940 environ). Le chiffrement s'opère symbole par symbole (usuellement des bits) en combinant une suite de symboles de texte clair avec une suite aléatoire de symboles de même nature, généralement par addition modulo 2, dans le cas des bits. La définition suivante est généralement utilisée :

**Définition 6** *Soit $\mathcal{A}$ un alphabet de $q$ symboles et soit $E_e$ un système de chiffrement par substitution simple où $e \in \mathcal{K}$, l'espace clef du système. Soit $m_1 m_2 \ldots$ une suite de texte clair et soit $e_1 e_2 e_3 \ldots$ une suite clé de $\mathcal{K}$. Un* chiffrement par flot *remplace la suite de texte clair par une suite de texte chiffré $c_1 c_2 c_3 \ldots$ où $c_i = E_{e_i}(m_i)$. Si $d_i$ désigne l'inverse de $e_i$ alors $D_{d_i} = m_i$ assure la transformation inverse du chiffré vers le clair.*

En terme d'applications, c'est encore le type de chiffrement préférentiellement et quasi-exclusivement utilisé dans le monde industriel (en particulier dans les télécommunications) et gouvernemental.

A cela, plusieurs raisons peuvent être données. C'est un type de chiffrement beaucoup plus rapide que le chiffrement par blocs. Il permet des implémentation en hard (circuits ASIC) beaucoup plus faciles, économiques (complexité moindre), que ne le permettent les systèmes par blocs. Lorsque le traitement doit se faire symbole reçu par symbole reçu (cas des télécommu-

1. Pour le chiffrement commercial cependant, de plus en plus depuis 1977, les systèmes par bloc sont très utilisés.

nications) ou lorsque la mémoire tampon est trés limitée (cas des satellites), ce type de chiffrement devient incontournable.

Du fait de leur haute résistance aux erreurs et donc au bruit (ils ne propagent pratiquement pas les erreurs [127, p. 191 et suiv.]), ces systèmes sont utilisés de façon privilégiée dans le cas de communications susceptibles d'être fortement bruitées (champ de bataille, atmosphère terrestre,...).

Mais c'est surtout d'un point de vue sécurité que ces systèmes font l'unanimité parmi les gouvernementaux. Bien que l'art [2] de l'ingénieur ait ici une importance extrême, la sécurité de ces systèmes est maintenant étayée par un vaste et impresssionnant corpus de connaissances théoriques, initié par C.E. Shannon. Les primitives y sont peu nombreuses (registres divers et fonctions booléennes pour l'essentiel) et très bien connues. Nombre d'attaques ont été publiées et on peut considérer avec une certaine sérénité ces systèmes, lorsque conçus dans les règles de l'art, comme très sûrs. On dispose à présent d'un recul suffisamment conséquent.

Les systèmes par blocs, encore jeunes (le pemier date de 1975), ne disposent pas encore de ce corpus de connaissances théoriques et pour l'œil intéressé, ils font plutôt figure d'enfer combinatoire, où rien n'est finalement véritablement maîtrisé et où tout peut être caché.

Reposant pour une bonne part sur le savoir-faire de l'ingénieur, peu d'algorithmes de chiffrement par flot sont en fait connus. Ils sont le plus souvent propriétaires et secrets [150, p. 372 et 379]. Différentes variantes existent : systèmes à combinaison de registres, systèmes filtrés, systèmes décimés, systèmes à horloge contrôlée régulièrement ou non, systèmes multiplexés, ... Je vais montrer dans ce chapitre, que pour beaucoup d'entre eux, il est possible d'exhiber une équivalence avec une classe générique de systèmes, au point de faire de cette dernière le paradigme de ce type de chiffrement : les *systèmes à combinaison de registres*.

Grâce à cette équivalence, cette classe a permis de mettre en lumière le fait que seules deux composantes (*primitives*) sont véritablement importantes : les registres à décalage linéaires et les fonctions booléennes. Même les systèmes non équivalents à cette classe (registres décimés par exemple), utilisent presque exclusivement ces primitives, tant les connaissances que l'on en a (en terme de sécurité) sont bien assises.

Dans ce chapitre, je rappellerai d'abord la classification générale des systèmes de chiffrement par flot (section 2.2) : bande aléatoire une fois,

2. Ian Cassels, mathématicien à Cambridge et ancien cryptanalyste de Bletchley Park, a une formule plus imagée mais ô combien exacte : "*Cryptography is a mixture of mathematics and muddle, and without the muddle the mathematics can be used against you*".

systèmes synchrones et asynchrones. J'exposerai ensuite (section 2.3) les principales définitions et propriétés concernant les registres à décalage et les séquences qu'ils génèrent. Dans la section 2.4 je présenterai les fonctions booléennes et en particulier les propriétés essentielles que ces fonctions doivent avoir lorsqu'elles sont utilisées dans un système de chiffrement par flot. Enfin dans la section 2.5, je présenterai les principales variantes de ces systèmes et j'exhiberai leur équivalence avec la classe de référence constituée des systèmes à combinaison de registres.

## 2.2 Classification générale

### 2.2.1 La bande aléatoire une fois

L'origine de ce chiffrement provient du système de Vernam [167].

**Définition 7** *Le* chiffrement de Vernam *est un système de chiffrement par flot sur l'alphabet* $\mathcal{A} = \{0, 1\}$. *Un message binaire* $m_1 m_2 m_3 \dots m_t$ *est combiné avec une suite-clef binaire* $k_1 k_2 k_3 \dots k_t$ *de même longueur pour produire une suite chiffrée* $c_1 c_2 c_3 \dots c_t$ *où*

$$c_i = m_i \oplus k_i, \quad 1 \leq i \leq t$$

Quand la suite clef est générée indépendamment et aléatoirement, on parle alors de chiffrement par *bande aléatoire une fois* (*One time pad*). Ce type de chiffrement est alors prouvé parfaitement sûr contre toute attaque, même à chiffré seul. Plus précisément, si $M, C$ et $K$ sont des variables aléatoires désignant respectivement le texte clair, le texte chiffré et la suite clef (secrète), alors

$$H(M) = H(M|C)$$

où $H(.)$ désigne la fonction entropie [3]. De façon équivalente, cela revient à dire que la transinformation (ou information mutuelle ) est nulle. Autrement dit la connaissance du chiffré ne fournit aucune information sur le clair.

Shannon a prouvé [152] qu'une condition de sécurité parfaite était que

$$H(K) \geq H(M)$$

C'est-à-dire que l'incertitude sur la clef secrète doit être au moins aussi grande que celle sur le clair. Si la clef est de longueur $k$ et si les bits sont

3. Pour une variable aléatoire discrète $X$ prenant les valeurs $X_1, X_2, \dots, X_n$ avec les probabilités non nulles $p_1, p_2, \dots, p_n$, $H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$

choisis aléatoirement et indépendamment les uns des autres, alors $H(K) = k$ et la condition de Shannon devient $k \geq H(M)$. La bande aléatoire une fois est inconditionnellement sûre, quelle que soit la distribution statistique des bits de clair.

Le problème majeur de ce schéma est que la clef doit être de longueur aussi grande que celle du texte clair à chiffrer. Cela complique extrêmement la génération et la gestion des clefs. Il est important de rappeler que l'utilisation répétée d'une même bande (pour plusieurs messages différents, appelés alors messages parallèles) est catastrophique et permet en quelques secondes de retrouver la suite clef et les messages.

Ce problème a donc motivé la conception de chiffrement par flot où la suite clef est *pseudo-aléatoire* c'est-à-dire générée de façon déterministe à partir d'une clef beaucoup plus petite. Comme, dans ce cas, $H(K) << H(M)$, le système n'est plus à secret parfait. Toutefois, on s'assure lors de la conception que la sécurité soit suffisante pour l'usage voulu (grosso modo le temps de calcul nécessaire à l'attaquant doit être prohibitif).

Ces systèmes se répartissent en deux catégories : les systèmes synchrones et auto-synchronisés.

### 2.2.2 Systèmes synchrones

**Définition 8** *Un système de chiffrement par flot* synchrone *est un système de chiffrement dans lequel la suite pseudo-aléatoire est générée indépendamment du message clair et du message chiffré.*

L'opération de chiffrement peut alors être résumée par les équations suivantes :

$$\begin{aligned} \sigma_{i+1} &= f(\sigma_i, k) \\ z_i &= g(\sigma_i, k) \\ c_i &= h(z_i, m_i) \end{aligned}$$

où $\sigma_0$ est l'état initial et est généralement déterminé par les bits de la clef secrète $k$, $f$ est la fonction d'état [4], $g$ est la fonction produisant la suite pseudo-aléatoire $z_i$ et $h$ est la fonction de combinaison entre cette suite $z_i$ et le clair $m_i$ produisant le chiffré $c_i$. Les opérations de chiffrement et de déchiffrement sont décrites dans la figure 2.1. Dans un système synchrone,

4. Cette appellation vient du domaine des automates finis, avec lesquels on peut effectivement décrire les systèmes de chiffrement par flot. Il est à noter d'ailleurs que certains automates dits de Mealy et de Moore ont été conçus pour réaliser du chiffrement. Ce sont en quelque sorte les ancêtres, peu connus, des systèmes actuels.

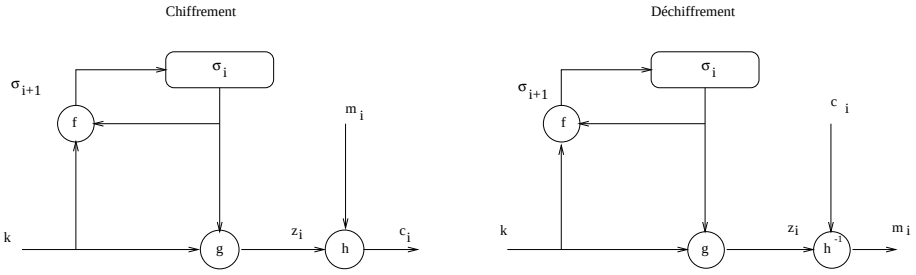


FIGURE 2.1 – Système de chiffrement synchrone

les émetteur et destinataire doivent être synchronisés (d'où le nom de synchrone), c'est-à-dire doivent utiliser à chaque temps $t$ le même bit de clef à la même position de texte clair et chiffré. Toute perte de synchronisation (insertion ou délétion d'un symbole) se traduit par un déchiffrement impossible. Des techniques de resynchronisation (réinitialisation, marqueurs,...) permettent de supprimer efficacement le problème. A noter que ces systèmes sont très résistants au bruit. Tout bit du texte chiffré modifié par le bruit sera mal déchiffré mais ce déchiffrement erroné ne perturbera en aucun cas celui des autres bits du texte chiffré.

La plupart des systèmes connus dans la littérature sont de ce type et plus exactement du type additif :

**Définition 9** *Un système de chiffrement* binaire additif *est un système par flot synchrone dans lequel la suite clef pseudo-aléatoire, la suite de texte clair et la suite de texte chiffré sont de nature binaire et où la fonction h est la fonction XOR.*

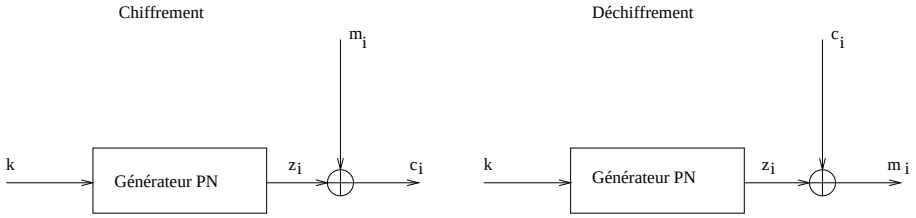


FIGURE 2.2 – Système de chiffrement additif

Le système est alors dénommé *générateur pseudo-aléatoire* ou générateur PN.

### 2.2.3 Systèmes auto-synchronisants

Ces systèmes sont encores appelés systèmes *asynchrones*.

**Définition 10** *Un système de chiffrement par flot* auto-synchronisant *ou* asynchrone *est un système dans lequel la suite pseudo-aléatoire est fonction de la clef et d'un nombre fixe de bits précédents du texte chiffré.*

Dans la littérature industrielle, le terme de système à autoclave sur le chiffré est également rencontré. L'opération de chiffrement peut alors être résumée par les équations suivantes :

$$\begin{aligned} \sigma_i &= (c_{i-t}, c_{i-t+1}, \ldots, c_{i-1}) \\ z_i &= g(\sigma_i, k) \\ c_i &= h(z_i, m_i) \end{aligned}$$

où $\sigma_0 = (c_{-t}, c_{-t+1}, \ldots, c_{-1})$ est l'état initial (non secret), $k$ est la clef, $g$ la fonction produisant la suite pseudo-aléatoire $z_i$ et $h$ est la fonction de combinaison entre cette suite $z_i$ et le clair $m_i$ produisant le chiffré $c_i$. La figure 2.3 décrit le principe général de ces systèmes. L'avantage de ces

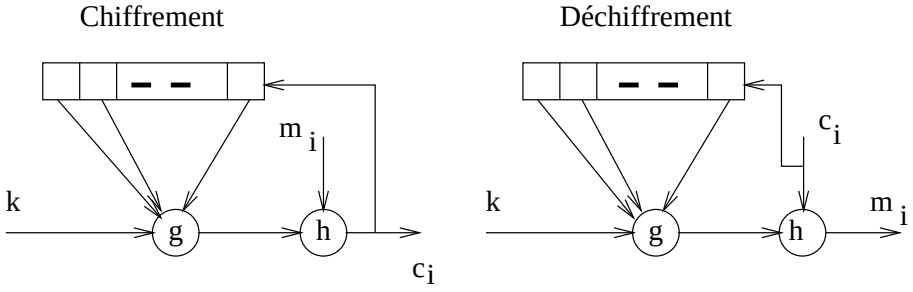


FIGURE 2.3 – Système de chiffrement asynchrone

systèmes est qu'en cas d'effacement ou d'insertion de bits dans le chiffré, l'auto-synchronisation reste possible étant donné que le déchiffrement ne dépend que d'un nombre limité de bits précédents de chiffré. Cela permet en cas de perte de synchronisation de la rétablir automatiquement. Un nombre restreint de bits sera alors perdu. La redondance de la langue permet alors de les retrouver.

La propagation des erreurs dues au bruit est de même limitée. Elle est limitée au nombre $t$ de bits de chiffré utilisés pour la génération de la suite pseudo-aléatoire. Enfin ces systèmes sont plus robustes que les systèmes

synchrones aux attaques utilisant les propriétés de redondance du texte clair, car chaque bit de clair influence la totalité du texte chiffré, assurant ainsi une très bonne diffusion des statistiques du clair.

## 2.3 Registres à décalage et séquences

La solution de la bande parfaitement aléatoire, utilisée de façon unique, n'est pas pratique. Il faut une clef différente pour chaque message, cette clef devant être de même longueur que le texte qu'elle chiffre. Cela pose évidemment des problèmes vite insurmontables de génération et de gestion des clefs.

L'idée est alors d'utiliser des suites dites *pseudo-aléatoires*, c'est à dire générées de façon déterministe mais en final présentant des propriétés aléatoires suffisantes. Pour générer de telles suites pseudo-aléatoires, quelques dizaines de bits, constituant la clef secrète (partagée par les protagonistes de la communication), initialisent les états internes d'un automate. De façon déterministe, l'automate alors génère une suite beaucoup plus longue possèdant les qualités statistiques requises.

Les éléments de base les plus utilisés dans ces automates sont les *registres à décalage à rétroaction linéaire*. Ils sont simples à implémenter en hardware, simples à programmer et très rapides.

Je vais maintenant rappeler les principales propriétés de ces objets dans le corps d'ordre deux $\mathbb{F}_2$. Elles sont généralisables à tout autre corps $\mathbb{F}_q$. Les démonstrations seront ici omises et le lecteur intéressé pourra les trouver dans [110] ou les articles cités.

### 2.3.1 Registres à décalage à rétroaction linéaire et suites à récurrence linéaire

Un registre à décalage à rétroaction linéaire de longueur $L$ est composé d'un registre à décalage contenant une suite de $L$ bits $(s_i, \ldots, s_{i+L-1})$ et d'une fonction de rétroaction linéaire (voir figure 2.4). A chaque impulsion d'horloge, le bit $s_i$ constitue la sortie du registre et les autres bits sont décalés vers la droite (ce sens est arbitraire et pourrait être inversé). La case de gauche à présent vide reçoit le bit $s_{i+L}$. Ce bit est produit par une fonction linéaire des bits $(s_i, \ldots, s_{i+L-1})$ :

$$s_{i+L} = c_1 s_{i+L-1} \oplus c_2 s_{i+L-2} \oplus \cdots \oplus c_L s_i$$

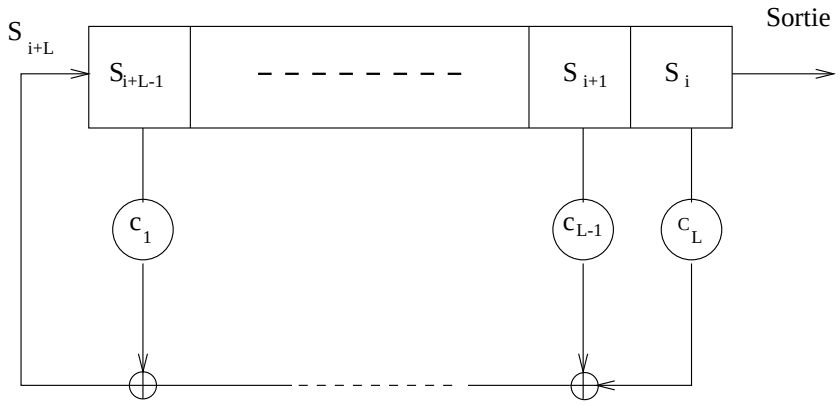


FIGURE 2.4 – Registres à décalage à rétroaction linéaire

où les coefficients de rétroaction $(c_i)_{1\leq i\leq L}$ sont à valeurs dans $\mathbb{F}_2$. Les bits $(s_0, \ldots, s_{L-1})$ qui déterminent complètement la suite, constituent l'*état initial du registre*.

Il est clair que la suite $(s_n)_{n\geq 0}$ produite par ce type de registre est une *suite à récurrence linéaire homogène d'ordre L*. Inversement, toute suite de ce type peut être générée par un registre de longueur $L$. On montre alors qu'une telle suite est *ultimement périodique*, c'est-à-dire qu'il existe un indice $n_0$ tel que la suite $(s_n)_{n\geq n_0}$ est périodique.

**Proposition 3** *[119] Toute suite à récurrence linéaire homogène d'ordre L sur $\mathbb{F}_2$ est ultimement périodique et sa plus petite période T est inférieure ou égale à $2^L - 1$. De plus, si le coefficient de rétroaction $c_L$ est non nul, alors la suite est périodique.*

En cryptographie, il est essentiel de disposer de suites ayant la plus grande période possible et idéalement maximale. Sinon, le cryptanalyste sera capable de prévoir la nature relative de bits disposés à intervalles réguliers (la période) et disposera alors d'équations à second membre connu, dont la résolution lui donnera une partie du secret. L'obtention d'une période maximale est subordonnée à une propriété sur le *polynôme caractéristique* du registre. En particulier le résultat suivant est intéressant (voir [90, 110, 148] pour plus de détails) :

**Définition 11** *(**Polynôme caractéristique d'un registre**) Soit un registre à décalage de rétroaction linéaire donnée par*

$$\forall i \geq 0,\ s_{i+L} = c_1 s_{i+L-1} \oplus c_2 s_{i+L-2} \oplus \cdots \oplus c_L s_i$$

*Son polynôme caractéristique est le polynôme $f$ de degré $L$ à coefficients binaires*

$$f(X) = X^L \oplus c_1 X^{L-1} \oplus c_2 X^{L-2} \oplus \cdots \oplus c_L$$

**Définition 12** *(***Ordre d'un polynôme***) Soir $f(X)$ un polynômes binaire. Son ordre, noté ord($f$), est le plus petit entier $t$ tel que $X^t \equiv 1$ mod $f(X)$.*

La notion d'ordre d'un polynôme permet alors de définir le résultat important suivant :

**Proposition 4** *Soit un registre à décalage à rétroaction linéaire de polynôme caractéristique $f$. Alors la plus petite période de la suite $(s_n)_{n\geq 0}$ produite par ce registre divise l'ordre de $f$.*

### 2.3.2 Complexité linéaire d'un registre à décalage

L'utilisation des séries génératrices permet de décrire plus finement la suite à récurrence linéaire et le mécanisme du registre la produisant. Cette approche est décrite en détail dans [110, 177]. La suite $(s_n)_{n\geq 0}$ est exprimée au moyen de la série formelle $s(X) = \sum_{n\geq 0} s_n X^n$ en faisant intervenir le *polynôme de rétroaction du registre.*

**Définition 13** *(***Polynôme de rétroaction d'un registre***) Soir un registre à décalage à rétroaction linéaire dont la fonction de rétroaction linéaire est donnée par la relation de récurrence :*

$$\forall i \geq 0,\ s_{i+L} = c_1 s_{i+L-1} \oplus c_2 s_{i+L-2} \oplus \cdots \oplus c_L s_i$$

*Son polynôme de rétroaction $f^*$ est le polynôme réciproque de son polynôme caractéristique, c'est-à-dire*

$$f^*(X) = 1 \oplus c_1 X \oplus c_2 X^2 \oplus \cdots \oplus c_L X^L = X^L f(\frac{1}{X})$$

Cela permet alors l'écriture de la suite $(s_n)_{n\geq 0}$ sous forme d'une fraction rationnelle décrivant le mécanisme de fonctionnement du registre :

**Proposition 5** *[110] La suite $(s_n)_{n\geq 0}$ est produite par un registre à décalage à rétroaction linéaire dont le polynôme de rétroaction est $f^*(X) = 1 \oplus c_1 X \oplus c_2 X^2 \oplus \cdots \oplus c_L X^L = X^L f(\frac{1}{X})$ si et seulement si son développement en série formelle $s(X) = \sum_{n=0}^{\infty} s_n X^n$ s'écrit :*

$$s(X) = \frac{g(X)}{f^*(X)}$$

*où $g$ est un polynôme tel que $deg(g) < deg(f^*)$. De plus, le polynôme $g$ est entièrement déterminé par l'état initial de la suite :*

$$g(X) = \sum_{j=0}^{L-1} \sum_{i=0}^{j} c_{i+k-j} s_i x^j \tag{2.1}$$

En fait le polynôme caractéristique du registre (donc le polynôme de rétroaction) peut ne pas être irréductible. Pour disposer d'une représentation la plus "minimale" possible de la suite, on définit la notion de polynôme caractéristique *minimal* de la suite $(s_n)_{n\geq 0}$. C'est le plus petit polynôme (en terme de degré) de tous les polynômes engendrant la suite $(s_n)_{n\geq 0}$.

**Définition 14** *(*Complexité linéaire*) Soit $(s_n)_{n\geq 0}$ une suite récurrente linéaire d'ordre $L$ sur $\mathbb{F}_2$ dont l'état initial est non nul*[5]*. Son* polynôme caractéristique minimal *est l'unique polynôme unitaire $f_0 \in \mathbb{F}_2[X]$ tel qu'il existe un polynôme $g_0 \in \mathbb{F}_2[X]$, avec $deg(g_0) < L$ et $\gcd(g_0, f_0^*) = 1$, vérifiant*

$$s(X) = \frac{g_0(X)}{f_0^*(X)}$$

*où $f_0^*(X)$ est le polynôme réciproque de $f_0$. La* complexité linéaire *du registre produisant la suite $(s_n)_{n\geq 0}$, notée $\Lambda(s)$, est alors égale au degré du polynôme caractéristique minimal de $(s_n)_{n\geq 0}$.*

La complexité linéaire d'un registre produisant une suite donnée $(s_n)_{n\geq 0}$ est donc définie comme la longueur du plus petit registre permettant de la générer. Cette complexité linéaire, à partir de la suite $(s_n)_{n\geq 0}$ peut être facilement déterminée à l'aide de l'algorithme de Massey-Berlekamp [7, 122]. A l'aide de seulement $2\Lambda(s)$ bits de la suite, le polynôme caractéristique minimal de la suite est entièrement retrouvé.

Cela pose évidemment un problème pour une suite qui se veut le plus aléatoire possible. $2\Lambda(s)$ bits suffisent pour retrouver (et donc prévoir) toute la suite. La première approche pour tenter de limiter cet inconvénient est d'utiliser des registres dont la complexité linéaire est égale à la longueur. Cela est possible si le polynôme caractéristique est irréductible [110]. La suite ne peut alors être engendrée par un registre plus court.

**Exemple 1** *Cet exemple est emprunté à [21]. Considérons la suite binaire $(s_n)_{n\geq 0}$ produite par le registre de longueur 10 de la figure 2.5. Le polynôme caractéristique est $f(X) = X^{10} \oplus X^9 \oplus X^7 \oplus X^6 \oplus X^3 \oplus 1$. Le polyôme de*

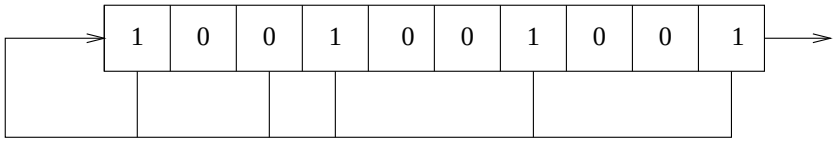


FIGURE 2.5 – Exemple de registre de longueur 10

*rétroaction est donc* $f^*(X) = X^{10} f(\frac{1}{X}) = X^{10} \oplus X^7 \oplus X^4 \oplus X^3 \oplus X \oplus 1$. *Soit* $g$ *le polynôme dont les coefficients sont donnés par l'état initial du registre* $(s_9, s_8, \ldots, s_0)$. *Avec la proposition 5 on obtient alors*

$$g(X) = X^7 \oplus X \oplus 1$$

*An appliquant toujours la même proposition, on peut donc calculer la série génératrice de la suite* $(s_n)_{n\geq 0}$ *:*

$$\sum_{n=0}^{\infty} s_n X^n = \frac{g(X)}{f^*(X)} = \frac{X^7 \oplus X \oplus 1}{X^{10} \oplus X^7 \oplus X^4 \oplus X^3 \oplus X \oplus 1} = \frac{1}{X^3 \oplus 1}$$

*Le polynôme caractéristique minimal du registre est donc*

$$f_0(X) = X^3 \oplus 1$$

*Il est facile de vérifier que les registres des figures 2.5 et 2.6 produisent*

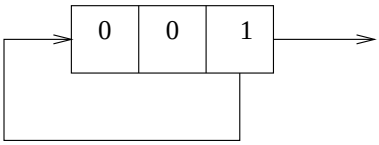


FIGURE 2.6 – Registre de longueur 3 produisant la même suite que le registre de longueur 10

*la même suite* $(s_n)_{n\geq 0}$. *Dans un contexte cryptologique, si un concepteur de système de chiffrement utilise un registre de longueur 10 mal choisi, un attaquant pourra déterminer toute la suite produite par ce registre avec seulement les six bits* $(s_0, s_1, s_2, \ldots, s_5)$ *de la suite.*

5. En fait, on peut considérer en pratique un état initial nul en considérant le rebouclage $1 \oplus s_{i+L}$. L'état tout à 1 est alors interdit. Certains schémas utilisent cette convention. Pour plus de détails, voir [111, page 190]

Le polynôme caractéristique minimal du registre sert également à déterminer la plus petite période de la suite engendrée.

**Proposition 6** *[110](*Période d'une suite à récurrence linéaire*)*
*Soit $(s_n)_{n\geq 0}$ un suite à récurrence linéaire de polynôme caractéristique minimal $f_0$ et dont l'état initial est non nul. Sa plus petite période est égale à l'ordre de $f_0$.*

Il est particulièrement important d'utiliser des suites ayant la plus grande période possible, pour une complexité linéaire donnée. Ces suites, dites *suites ML* (de longueur maximale) sont obtenues à partir de registres dont le polynôme caractéristique minimal est primitif (voir [110, Chap. 3]. On a alors le résultat suivant, très important :

**Théorème 1** *Si le polynôme de rétroaction minimal d'un registre est primitif et que son état initial est non nul, alors la suite $(s_n)_{n\geq 0}$ produite par ce registre est de période maximale égale à $2^{\Lambda(s)} - 1$.*

Ce théorème explicite le lien entre la complexité linéaire du registre et la taille de la période. Plus grande sera cette complexité, plus grande sera la période.

### 2.3.3 Combinaison de suites à récurrence linéaire : le générateur à combinaison de registres

Toutefois, même lorsque le polynôme caractéristique du registre est choisi avec soin (*i.e.* primitif), la complexité linéaire de la suite (et par là sa période) n'est pas suffisante pour contrer certaines attaques (et en tout premier lieu celle menée avec l'algorithme de Massey-Berlekamp).

De plus, la suite $(s_n)_{n\geq 0}$ issue du registre ne peut en elle-même être utilisée comme suite aléatoire servant au chiffrement. Même si elle présente des propriétés statistiques assez intéressantes (pour plus de détails voir l'ouvrage de référence de S.W. Golomb [76]), cette suite possède des propriétés de linéarité extrêmement fortes. La combiner telle quelle au texte clair ne fournirait aucune sécurité et des attaques très efficaces (sous une forme légèrement modifiée) sont possibles [175, 176]. Il est donc nécessaire de faire disparaitre au maximum ces propriétés de linéarité, de les briser suffisamment pour les rendre inexploitables par le cryptanalyste.

La solution à ces deux préoccupations, briser la linéarité et augmenter la complexité linéaire de la suite produite, consiste à combiner $n$ registres en parallèle à l'aide d'une fonction booléenne $f : \mathbb{F}_2^n \mapsto \mathbb{F}_2$, appelée *fonction de combinaison*. Ce système constitue le système de chiffrement à flot le plus

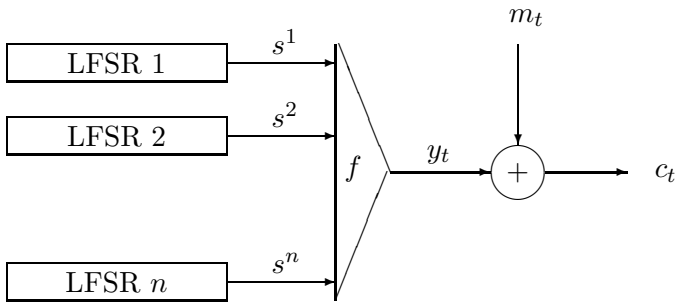


FIGURE 2.7 – Générateur à combinaison de registres

simple qui soit : les générateurs à combinaison de registres. Dans ce type de systèmes, la suite $(y_t)_{t\geq 0}$ issue de la fonction $f$ est combinée par addition modulo 2 à la suite $(m_t)_{t\geq 0}$ de texte clair produisant la suite $(c_t)_{t\geq 0}$ de texte chiffré (voir figure 2.7). Nous verrons un peu plus loin que la fonction booléenne doit être choisie très soigneusement pour offrir une résistance aux attaques.

L'avantage de ce schéma est d'augmenter considérablement l'espace des clefs. En effet, la clef secrète est alors constituée non plus de l'initialisation d'un seul registre mais de plusieurs.

Il est important de connaitre les propriétés de la nouvelle suite $(y_t)_{t\geq 0}$ afin d'en évaluer la sécurité. Quelle est en particulier sa complexité linéaire et de là sa période ? La suite, de façon évidente, est générée par au moins un registre de polynôme caractéristique $f(X) = X^N \oplus 1$, si cette suite est de longueur $N$ et la longueur de cette suite sera un majorant de sa période. Il n'est donc pas absurde de continuer à parler de complexité linéaire. Il faut juste s'assurer qu'il n'exite pas de polynôme caractéristique de degré inférieur pouvant également l'engendrer.

Pour étudier cette nouvelle complexité linéaire, on utilise la *Forme Algébrique Normale* de la fonction $f$ qui sera définie plus précisément dans le chapitre 5. Cette forme peut-être définie comme une représentation polynomiale multivariée. Plus précisément, c'est l'unique polynôme

$$Q_f \in \mathbb{F}_2[x_1, x_2, \ldots, x_n]/(x_1^2 - x_1, \ldots, x_n^2 - x_n)$$

tel que pour tout vecteur $(u_1, u_2, \ldots, u_n)$ de $\mathbb{F}_2^n$ on ait

$$f(u_1, u_2, \ldots, u_n) = Q_f(u_1, u_2, \ldots, u_n).$$

Le degré de $f$ est le degré total du polynôme $Q_f$.

Utilisant cette représentation polynomiale, il suffit juste de définir la somme de deux suites à récurrence linéaire et leur produit [6]. On a donc les résultats suivants [74, 88, 89, 90, 149, 151, 178]

**Proposition 7** *Soit $(u_n)_{n\geq 0}$ et $(v_n)_{n\geq 0}$ deux suites à récurrence linéaire de polynômes de rétroaction minimaux respectifs $f_0$ et $g_0$. Alors la complexité linéaire de leur somme vérifie*

$$\Lambda(u \oplus v) \leq \Lambda(u) + \Lambda(v)$$

*avec égalité si et seulement si $pgcd(f_0, g_0) = 1$. De plus la période de leur somme est égale au ppcm des périodes de $(u_n)_{n\geq 0}$ et $(v_n)_{n\geq 0}$.*

La situation est donc idéale quand les deux polynômes de rétroaction sont primitifs et de périodes relativement première.

**Proposition 8** *Soient $(u_n)_{n\geq 0}$ et $(v_n)_{n\geq 0}$ deux suites à récurrence linéaire de polynômes de rétroaction minimaux respectifs $f_0$ et $g_0$. Alors la complexité linéaire de leur produit vérifie*

$$\Lambda(u.v) \leq \Lambda(u)\Lambda(v)$$

*avec égalité si et seulement si les polynômes $f_0$ et $g_0$ sont primitifs et si $pgcd(f_0, g_0) = 1$. Dans ce cas la période de $(uv)_{n\geq 0}$ est égale au produit des périodes de $(u_n)_{n\geq 0}$ et $(v_n)_{n\geq 0}$.*

Les propositions 7 et 8 se généralisent immédiatement à $k$ suites. Le théorème général suivant permet d'évaluer la complexité et la période dans le cas d'un système à combinaison de registres.

**Théorème 2** *Soient $n$ registres à décalage à rétroaction linéaire dont les polynômes de rétroaction sont primitifs et de degrés $L_1, L_2, \ldots, L_n$ deux à deux premiers entre eux. Alors la complexité linéaire de la suite produite en combinant ces registres par la fonction booléenne $f$ à $n$ variables est égale à*

$$L = f(L_1, L_2, \ldots, L_n)$$

*où $f$ est vue comme un polynôme de $\mathbb{F}_2[x_1, x_2, \ldots, x_n]$ et est évalué sur les entiers.*

6. Pour la généralisation à des corps $\mathbb{F}_q$ avec $q > 2$, il faudrait également définir la complexité linéaire de la puissance d'une suite à récurrence linéaire.

**Exemple 2** *Considérons le générateur de Geffe [71]. Il est défini par trois registres dont les polynômes de rétroaction sont primitifs et de degrés $L_1, L_2$ et $L_3$ deux à deux premiers entre eux. Ces registres sont assemblés par la fonction booléenne suivante*

$$f(x_1, x_2, x_3) = x_1 \oplus x_1x_2 \oplus x_2x_3$$

*La complexité linéaire de la suite produite par ce générateur est donc $L_1 + L_1L_2 + L_2L_3$.*

Une question intéressante touche à la complexité linéaire d'une suite binaire prise au hasard. Soit $u_n$ une suite prise aléatoirement au hasard parmi toutes les suites binaires de longueur $n$ (donc en particulier de l'aléa vrai fourni, par exemple, par une résistance thermique échantillonnée) et soit $\Lambda(u_n)$ sa complexité linéaire. Si $par(x)$ représente la parité de $x$ alors on a le résultat suivant

**Proposition 9** *[148, Chap. 4] Une suite $u_n$ de longueur $n$ aléatoirement choisie a une complexité linéaire*

1. *d'espérance mathématique*

$$E(\Lambda(u^n)) = \frac{n}{2} + \frac{4 + par(n)}{18} - \frac{1}{2^n}\left(\frac{n}{3} + \frac{2}{9}\right)$$

2. *de variance*

$$\begin{aligned} Var(\Lambda(u^n)) &= \frac{86}{81} - \frac{1}{2^n}\left(\frac{14 - par(n)}{27} + \frac{82 - 2par(n)}{81}\right) \\ &\quad - \frac{1}{2^{2n}}\left(\frac{n^2}{9} + \frac{4n}{27} + \frac{4}{81}\right) \end{aligned}$$

La complexité linéaire d'un suite prise au hasard est donc en moyenne égale à la moitié de sa longueur.

Les propositions 7 et 8 peuvent s'énoncer sous une forme différente, qui sera utilisée dans le chapitre suivant. Notons $\mathcal{S}(P)$ l'ensemble de toutes les séquences produites par un registre de polynôme de rétroaction $P$. Alors

**Proposition 10** *[90, 149, 151] Soit $P$ et $Q$ deux polynômes non constants sur $\mathbb{F}_2$. Alors nous avons*

- $\{(u_t + v_t)_{t>0},\ u \in \mathcal{S}(P), v \in \mathcal{S}(Q)\} = \mathcal{S}(R)$ *où $R$ est le plus petit commun multiple de $P$ et $Q$.*

– $\{(u_t v_t)_{t>0},\ u \in \mathcal{S}(P), v \in \mathcal{S}(Q)\} = \mathcal{S}(R)$ *où* $\deg(R) \leq \deg(P)\deg(Q)$. *L'égalité est réalisée si et seulement si au moins un des polynômes* $P$ *et* $Q$ *possède seulement des racines simples et si tous les produits* $\alpha\beta$ *sont distincts quels que soient* $\alpha$ *et* $\beta$ *tels que* $P(\alpha) = 0$ *et* $Q(\beta) = 0$ *dans un corps de décomposition commun. Cette condition est notamment satisfaite si* $P$ *et* $Q$ *sont d'ordres premiers entre eux.*

Une borne inférieure sur le degré de $R$ peut également se déduire de la multiplicité des racines de $P$ et $Q$ et du nombre des produits distincts $\alpha\beta$ [80].

**Proposition 11** [110, Th. 8.53] *Soit* $P$ *et* $Q$ *deux polynômes non constants sur* $\mathbb{F}_2$. *Alors* $\mathcal{S}(P)$ *est un sous-ensemble de* $\mathcal{S}(Q)$ *si et seulement si* $P$ *divise* $Q$.

Cette proposition implique que si une séquence $s$ est générée par un registre à décalage de polynôme $P$, alors il satisfait les relations de récurrence (également appelées équations de parité ) correspondant à $PQ$ pour un quelconque $Q \in \mathbb{F}_2[X]$.

Pour tout polynôme de rebouclage fixé, $P$ de degré $L$, je ne considérerai dans le chapitre suivant, que tous les multiples de $P$ de poids $d$ où $d$ est petit. Cette approche est similaire aux techniques d'attaque par corrélation rapide [25, 100, 125]. La formule suivante (voir par exemple [25]) fournit une excellente approximation du nombre moyen $m(d)$ de multiples $Q$ de $P$ ayant un poids $d$ et de degré au plus $D$ $Q(X) = 1 + \sum_{j=1}^{d-1} X^{i_j}$ :

$$m(d) \simeq \frac{D^{d-1}}{(d-1)!2^L} \ . \tag{2.2}$$

### 2.3.4 Registres à décalage à rebouclage non linéaire

Si on se limite, pour la fonction de rétroaction d'un registre, aux fonctions booléennes à $n$ variables, linéaires, le nombre de ces objets est extrêmement limité dès que $n$ augmente un peu (il y en a $2(2^n - 1)$). Par contre si on considère toutes les fonctions booléennes à $n$ variables, leur nombre est bien plus important : $2^{2^n}$. Cela permet peut-être d'augmenter le nombre de registres capables de fournir des suites convenables pour un usage cryptographique, voire peut-être de meilleure qualité.

**Définition 15** *(***Registre à décalage à rebouclage non linéaire***)*
*Un registre à décalage à rebouclage non linéaire de longueur* $L$ *est un registre dans le lequel la fonction de rétroaction est une fonction booléenne non linéaire.*

En fait, le domaine des séquences générées par de tels registres a très peu été étudié et la plupart des problèmes sont encore ouverts. De fait, c'est la raison pour laquelle peu de systèmes de chiffrement par flot les utilisent. Le principal problème, de taille, est qu'aucun résultat connu ne permet de prédire ou d'encadrer la valeur de la période de telles suites [7]. Le seul résultat pertinent est le suivant.

**Définition 16** *Un registre à décalage à rebouclage non linéaire est dit* non singulier *si et seulement si chacune des séquences qu'il engendre est périodique.*

**Proposition 12** *[76, 106] Un registre à décalage à rebouclage non linéaire de fonction de rétroaction $f(x_1, x_2, \ldots, x_n)$ est non singulier si et seulement si $f$ est de la forme $f = x_n \oplus g(x_1, x_2, \ldots, x_{n-1})$ où $g$ est une fonction booléenne à $n-1$ variables.*

X. Lai a prouvé que cette proposition n'était valable que dans le cas binaire et a généralisé les conditions de non singularité dans le cas des autres corps finis [106].

## 2.4 Fonctions booléennes et chiffrement par flot

Les fonctions booléennes constituent des primitives cryptographiques essentielles dans le chiffrement à flot (et en cryptographie d'une manière générale). On a vu dans la section précédente que dans le cas de la combinaison de plusieurs registres, le degré de la forme algébrique normale de la fonction avait une incidence sur la complexité linéaire de la suite produite. De plus, elles sont indispensables pour briser les propriétés de linéarité inhérentes aux suites produites directement par les registres.

En fait, la plus grande partie de la sécurité d'un système de ce type repose sur ces fonctions qui le composent. C'est la raison pour laquelle la recherche sur les fonctions booléennes est très active et surtout capitale. L'enjeu est de trouver des fonctions qui possèdent le plus grand nombre, simultanément, de propriétés cryptographiques fortes : équilibre, haute non linéarité, immunité aux corrélations, critère de propagation,....Malheureusement depuis les travaux de Meier et Staffelbach [126] on sait que certains critères sont incompatibles, ce qui oblige à rechercher des compromis. Ces derniers seront en fait surtout dictés par l'implémentation, c'est-à-dire l'usage pratique de ces fonctions dans un schéma réel (c'est là qu'intervient l'art de l'ingénieur).

7. Il est cependant évident que la période est inférieure ou égale à $2^L - 1$

### 2.4.1 Définitions

Une *fonction booléenne à n variables* est une application de $\mathbb{F}_2^n$ dans $\mathbb{F}_2$. Une première façon de décrire une telle fonction est de donner sa *table de vérité*. C'est une table constituée de toutes les valeurs possibles de $\mathbb{F}_2^n$ avec la valeur de la fonction $f$ en chacune d'elles.

**Définition 17** *On appelle* support *de la fonction booléenne $f$ l'ensemble des éléments $u$ tels que $f(u) \neq 0$ et on le note $supp(f)$. On appelle* poids *de $f$ le cardinal de son support et on le note $wt(f)$. On définit la* distance *séparant deux fonctions booléennes $f$ et $g$ par $d(f,g) = wt(f \oplus g)$.*

Si on considère la table de vérité comme un mot binaire de longueur $2^n$ alors cette distance est simplement la distance de Hamming entre deux mots. Cette représentation par sa table de vérité a permis de caractériser combinatoirement [20, 79] les fonctions booléennes sans corrélation à l'aide de tableaux orthogonaux, objets introduits en 1947 par Rao [142]. Cependant, cette représentation n'est pas pratique à manipuler. Il faut en effet une complexité mémoire (stockage) et calcul (génération et lecture) en $\mathcal{O}(2^n)$. Le problème reste du même ordre si on travaille uniquement sur le support. Il suffit de considérer le cas d'une fonction de poids $2^{n-1}$.

C'est la raison pour laquelle on utilise plus souvent une autre représentation, plus compacte : la forme algébrique normale (FAN).

**Définition 18** *La* Forme Algébrique Normale d'une fonction booléenne $f$ *à $n$ variables est l'unique polynôme $Q_f$ de $\mathbb{F}_2[x_1, x_2, \ldots, x_n]/(x_1^2 - x_1, \ldots, x_n^2 - x_n)$ tel que*

$$\forall (u_1, u_2, \ldots, u_n) \in \mathbb{F}_2^n,\ f(u_1, u_2, \ldots, u_n) = Q_f(u_1, u_2, \ldots, u_n).$$

*On appelle degré de $f$, noté $deg(f)$ le degré du polynôme $Q_f$. Une fonction de degré 1 est dite* affine *et si de plus $f(0, 0, \ldots, 0) = 0$ alors elle est dite* linéaire

J'utiliserai la notation suivante pour ce polynôme de la FAN :

$$f(x_1, x_2, \ldots, x_n) = \sum_{x \in \mathbb{F}_2^n} a_u x^u, \quad a_u \in \mathbb{F}_2$$

où $u = (u_1, u_2, \ldots, u_n)$ et $x^u = x_1^{u_1} x_2^{u_2} \ldots x_n^{u_n}$. Les $(a_u)_{u \in \mathbb{F}_2^n}$ sont les coefficients de la FAN. On les calcule au moyen de la transformée de Möbius [114] (voir section 5.2) mais ce calcul est encore de complexité exponentielle. Pour la suite, la fonction $f$ sera confondue avec sa forme algébrique normale.

Récemment [36, 37], une nouvelle forme normale a été définie par C. Carlet et P. Guillot : la *Forme Numérique Normale* (NNF). Ses coefficients ne sont plus binaires mais entiers. Elle est établie par la transformée de Möbius dans laquelle l'addition n'est plus faite modulo 2 mais dans l'anneau des entiers. Cette nouvelle vision a d'ores et déja permis des caractérisations simples des fonctions courbes et des fonctions résilientes [37].

Il existe de nombreuses autres représentations et caractérisations de fonctions booléennes ; un exposé détaillé pourra être trouvé dans [21, Chap. 6]. Toutes se répartissent toutefois en deux catégories : la fonction est vue comme un polynôme. C'est alors la FAN que l'on considère. Ou bien la fonction (sa table de vérité) est vue comme un mot de code. Les outils de la théorie algébrique des codes peuvent alors être utilisés.

En fait, la théorie des codes, depuis près de 50 ans, travaille plus généralement sur les objets des mathématiques discrètes. L'accumulation des résultats concernant ces objets est énorme tant en qualité qu'en quantité. Depuis quelques années, la cryptologie (en l'occurrence ici les fonctions booléennes considérées comme des primitives cryptologiques) s'intéresse à ce gigantesque corpus de connaissances pour accroître ses domaines de compréhension et d'action. Pour une présentation détaillée de cette vision, le lecteur pourra consulter la thèse de Caroline Fontaine [65], [22] ou [139, Chap. 1, 8 ,11 et 14].

**Définition 19** *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.*

**Proposition 13** *(*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 plus ou moins bonnes

de la fonction, d'un point de vue cryptographique. Cette transformée de Walsh-Hadamard est extrêmement utile pour étudier les fonctions booléennes. Mentionnons en particulier la caractérisation de l'immunité aux corrélations et de la non linéarité, que nous aborderons plus tard.

J'aurai besoin dans le chapitre 4 de calculer les coefficients $a_u$ de la forme algébrique normale d'une fonction à partir de ses coefficients de Walsh $\widehat{\chi_f}(u)$. J'utiliserai la formule donnée par la proposition suivante.

**Proposition 14** *Soient $f$ une fonction booléenne à $n$ variables et $(a_u)_{u\in\mathbb{F}_2^n}$ les coefficients de sa forme algébrique normale. Alors nous avons pour tout $u \in \mathbb{F}_2^n$*

$$a_u = 2^{wt(u)-1}\left(1 - \frac{1}{2^n}\sum_{v\preceq\bar{u}} \widehat{\chi}_f(v)\right) \bmod 2$$

*où $\bar{u}$ représente la complétion à un, bit à bit, $wt(u)$ représente le poids de Hamming de $u$ et $\preceq$ représente l'ordre partiel sur le treillis booléen défini par*

$$\alpha \preceq \beta \Leftrightarrow \alpha_i \leq \beta_i \ \ \forall\, 1 \leq i \leq n.$$

*Preuve.*
En utilisant la transformée de Möbius de $f$ :

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

nous avons quel que soit $u \in \mathbb{F}_2^n$

$$\begin{aligned} a_u &= \sum_{x\preceq u} f(x) \bmod 2 = \sum_{x\preceq u} \frac{1}{2}\left(1 - (-1)^{f(x)}\right) \bmod 2 \\ &= 2^{wt(u)-1} - \frac{1}{2}\sum_{x\preceq u}(-1)^{f(x)} \bmod 2 \end{aligned}$$

Comme la transformée de Fourier normalisée est involutive, nous avons

$$\forall x \in \mathbb{F}_2^n,\ \ (-1)^{f(x)} = 2^{-n}\sum_{v\in\mathbb{F}_2^n} \widehat{\chi}_f(v)(-1)^{v\cdot x}\ .$$

En combinant ces équations, nous en déduisons que

$$\begin{aligned} a_u &= 2^{wt(u)-1} - 2^{-n-1}\sum_{x\preceq u}\sum_{v\in\mathbb{F}_2^n} \widehat{\chi}_f(v)(-1)^{v\cdot x} \bmod 2 \\ &= 2^{wt(u)-1} - 2^{-n-1}\sum_{v\in\mathbb{F}_2^n} \widehat{\chi}_f(v)\left(\sum_{x\preceq u}(-1)^{v\cdot x}\right) \bmod 2\ . \end{aligned}$$

L'ensemble $E_u = \{x \in \mathbb{F}_2^n,\ x \preceq u\}$ est sous-espace vectoriel de $\mathbb{F}_2^n$ de dimension $wt(u)$. Son orthogonal, $E_u^{\perp}$, satisfait $E_u^{\perp} = E_{\bar{u}}$. Il s'ensuit que

$$\sum_{x \preceq u} (-1)^{v \cdot x} = \begin{cases} 2^{wt(u)} & \text{si } v \in E_{\bar{u}}, \\ 0 & \text{sinon.} \end{cases}$$

Nous obtenons donc que, $\forall u \in \mathbb{F}_2^n$,

$$a_u = 2^{wt(u)-1} - 2^{-n-1+wt(u)} \sum_{v \preceq \bar{u}} \widehat{\chi}_f(v) \bmod 2 \ .$$

■

J'utiliserai ce résultat lors de la phase de reconstruction de la fonction de combinaison (section 4.3).

### 2.4.2 Propriété d'équilibre

**Définition 20** *Une fonction booléenne $f$ à $n$ variables est dite* équilibrée *lorsqu'elle prend autant de fois la valeur 1 que la valeur 0. Autrement dit, $f$ est équilibrée si et seulement si $wt(f) = 2^{n-1}$.*

En cryptographie cette propriété est importante. Elle fait de la sortie de la fonction une variable aléatoire de répartition uniforme. Dans le cas contraire, le biais en faveur du 1 ou du 0 pourrait fournir des informations utilisables par le cryptanalyste et lui donner un certain degré de prédictabilité sur la suite.

Par définition (voir définition 19), il est clair que :

**Proposition 15** *Une fonction booléenne est équilibrée si et seulement si $\widehat{\chi_f}(0) = 0$.*

Il en résulte que déterminer si une fonction est équilibrée, dans le cas général est de complexité exponentielle. Je montrerai plus tard (chapitre 5) que dans certains cas, ce problème devient de complexité polynomiale. Cela me permettra de construire une classe non triviale de fonctions équilibrées.

### 2.4.3 Propriété d'immunité aux corrélations

En 1985, Siegenthaler [155] proposa une des premières attaques publiées, pour les systèmes de chiffrement par flot. Je rappellerai cette attaque de façon détaillée dans le chapitre 3. Elle consiste à retrouver le contenu des

registres (la clef secrète) par une approche du type "diviser pour régner", indépendamment les uns des autres. Elle n'est possible que s'il existe une *corrélation* entre la sortie de la fonction de combinaison et l'une de ses entrées. Autrement dit, si la probabilité $P[f(x_1, x_2, \ldots, x_n) = x_i]$ pour un $i$ donné est différente de $\frac{1}{2}$. Cette probabilité peut s'exprimer en fonction de la transformée de Walsh-Hadamard de la fonction $f$ :

$$P[f(x_1, x_2, \ldots, x_n) = x_i] = \frac{1}{2}\left(1 + \frac{\widehat{\chi_f}(e_i)}{2^n}\right) \tag{2.3}$$

où $e_i$ est le vecteur binaire dont la seule composante non nulle est en $i$ème position.

Pour lutter contre cette attaque, le critère d'immunité aux corrélations a été défini et étudié [154].

**Définition 21** *Une fonction booléenne à n variables est dite* sans corrélation d'ordre *t si sa distribution de valeurs ne change pas lorsque l'on fixe au plus t entrées. Une fonction équilibrée sans corrélation d'ordre t est dite* résiliente d'ordre *t ou encore t-résiliente.*

En 1988, G.-Z. Xiao et J.L. Massey [173] ont explicité ce critère en termes de transformée de Walsh-Hadamard.

**Proposition 16** *[173] La fonction booléenne f à n variables est sans corrélation d'ordre t si et seulement si elle vérifie*

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

*Elle est résiliente d'ordre t si de plus* $\widehat{\chi_f}(0) = 0$.

On notera une fonction sans corrélation d'ordre $t$, une fonction CI(t). De façon équivalente, la fonction est dite corrélée à l'ordre $t + 1$.

D'un point de vue cryptanalytique, il est clair, en considérant l'équation 2.3, que lorsque le coefficient de Walsh-Hadamard est nul pour une valeur $u \in \mathbb{F}_2^n$, alors il n'existe pas de corrélation entre la sortie de la fonction et la fonction affine $< u, x >$ pour tous les $x \in \mathbb{F}_2^n$.

Quand ce coefficient est non nul, au contraire une telle corrélation existe. Son importance dépendra alors de la valeur de $\widehat{\chi_f(u)}$. En pratique, pour mener une attaque de type Siegenthaler, il faudra donc considérer un nombre de registres au moins égal à $wt(u)$ simultanément, pour exploiter une telle corrélation. Les attaques seront donc d'autant plus coûteuses que l'ordre de non corrélation est élevé. C'est là précisément que réside l'attrait des fonctions ayant un ordre élevé d'immunité aux corrélations.

Notons qu'il existe toujours une corrélation et plus son ordre est élevé plus elle risque d'être importante, en vertu de l'équation de Parseval. Diverses constructions ont été proposées pour construire des fonctions CI(t) avec $t$ grand (voir [58] et [65, p. 45 et suiv.] pour un exposé détaillé).

C. Fontaine [65] a exhibé des fonctions offrant à la fois un ordre de corrélation élevé et une répartition homogène des valeurs non nulles du spectre de Walsh-Hadamard. Ces fonctions sont actuellement les meilleures connues. D'autres constructions récentes ont été également proposées à la suite de ses travaux [117, 118].

Siegenthaler a donné [154] une borne sur le degré des fonctions sans corrélation et résilientes. Cela préfigure la notion de compromis à trouver pour concilier plusieurs critères (ici l'ordre d'immunité aux corrélations et le degré ; ce dernier influence directement la complexité linéaire ; voir section 2.3.3).

**Proposition 17** *Soit une fonction booléenne à $n$ variables. Alors :*
- *Si $f$ est sans corrélation d'ordre $t$, alors $deg(f) \leq n - t$.*
- *Si $f$ est $t$-résiliente, alors :*
  - *si $t \leq n - 2$, on a $deg(f) \leq n - t - 1$.*
  - *si $t = n - 1$, on a $deg(f) = 1$.*

### 2.4.4 Propriété de non-linéarité

L'importance de ce critère a déja été montré pour la complexité linéaire d'une suite provenant de la combinaison de plusieurs registres. Le problème, comme l'a souligné J.L. Massey [123] est de bien définir ce que l'on appelle non-linéarité. La première approche (historiquement) a été de la définir par le degré de la forme algébrique normale de la fonction. Cette approche est utilisée pour évaluer la complexité linéaire.

Mais considérons par exemple la fonction

$$f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3 \oplus x_1x_2x_3.$$

Elle est de degré 3, degré maximal pour une fonction sur $\mathbb{F}_2^3$. Toutefois, il est facile de voir que cette fonction, en fait est très proche d'une fonction linéaire (dont le degré est 1). Cela montre que cette approche de la non linéarité d'une fonction booléenne n'est pas suffisante.

La seconde approche consiste alors à définir la non-linéarité comme étant sa distance minimale de toute fonction affine, c'est à dire utilisant la distance de Hamming de la fonction au code de Reed-Müller d'ordre 1, $\mathcal{R}(1, n)$ (voir pour plus de détails [22]). De ce point de vue, le mot de code représentant

la fonction doit être le plus loin possible de tous mots du code $\mathcal{R}(1,n)$. Les mots de ce code représentent les fonction linéaires ou affines.

**Définition 22** *On appelle* non-linéarité d'une fonction booléenne *à $n$ variables , notée $nl(f)$, la distance qui la sépare de l'ensemble des fonctions affines à $n$ variables :*

$$nl(f) = \min_{g\, affine}(d(f,g))$$

On a alors

**Proposition 18** *[65, page 46] Soit une fonction booléenne à $n$ variables. Sa non-linéarité est égale à*

$$nl(f) = 2^{n-1} - \frac{1}{2}\sup_{u\in\mathbb{F}_2^n}\left(|\widehat{\chi_f}(u)|\right).$$

Dans le contexte du chiffrement à flot, l'importance de ce critère n'a jamais vraiment été soulignée. Tout au plus était-il jugé comme un aspect plus global de l'immunité aux corrélations, critère, lui, plutôt local (puisque relatif à telle ou telle entrée de la fonction). Si cette vision n'est pas fausse, bien au contraire, l'importance de la non-linéarité a été mise un peu plus en lumière dans [65, page 47].

**Théorème 3** *[65] Soit $f$ une fonction booléenne à $n$ variables, utilisée comme fonction de combinaison de $n$ registres à décalage à rétroaction linéaire. Supposons que la fonction est $t$-résiliente et non $(t+1)$-résiliente. Alors la fonction booléenne $g$ dépendant de $t+1$ variables qui se rapproche le plus de $f$ est une fonction affine de la forme*

$$\epsilon \oplus \bigoplus_{i\in T} x_i$$

*où $\epsilon \in \mathbb{F}_2$ et où $T$ désigne l'ensemble des indices des variables dont dépend $g$, autrement dit les numéros des registres attaqués.*

Ainsi, puisque la meilleure approximation de $f$ par une fonction dépendant de $t+1$ variables est affine, l'attaque par corrélation sur $t+1$ registres sera d'autant plus coûteuse que la fonction $f$ sera loin des fonctions affines, c'est-à-dire que $f$ aura une grande non-linéarité.

Idéalement, les fonctions réalisant totalement cette situation sont les fonctions dites *courbes* .

**Définition 23** *Les fonctions booléennes $f$ à $n$ variables vérifiant $|\widehat{\chi_f}(u)| = 2^{\frac{n}{2}}$ pour tout $u \in \mathbb{F}_2^n$, sont appelées* fonctions courbes.

Malheureusement, les fonctions courbes ne sont pas des fonctions équilibrées. Elles ne peuvent pas, par conséquent, être également résilientes (voir section 2.4.5).

Ces fonctions ont été caractérisées par O.S. Rothaus [146] qui a classé toutes les fonction courbes à six variables. Parallèlement, J.F. Dillon [48] les a largement étudiées. Depuis de nombreux travaux ont été menés sur les fonctions courbes [26, 28, 29, 30, 31, 34, 35, 37, 33, 83, 93, 94].

**Proposition 19** *[146] Les fonctions booléennes courbes n'existent que pour un nombre pair, $n$, de variables. De plus, si $f$ est une telle fonction, on a*
- *si $m \geq 4$ alors $deg(f) \leq \frac{n}{2}$.*
- *si $m = 2$, alors $deg(f) = 1$*

**Proposition 20** *[48, Th. 6.2.2, page 74] Soit $f$ une fonction booléenne à $n$ variables. Alors $f$ est courbe si et seulement si la fonction*

$$\begin{array}{rcl} \mathbb{F}_2^n & \rightarrow & \mathbb{F}_2 \\ x & \mapsto & f(x \oplus a) \oplus f(x) \end{array}$$

*est équilibrée pour tout vecteur non nul $a$ de $\mathbb{F}_2^n$.*

En fait, concernant les fonctions courbes, beaucoup de problèmes restent encore non résolus. Peu de choses sont connues sur celles de degré 3. D'une manière générale, on ne connait pas le nombre total de fonctions courbes. Plusieurs constructions ont été proposées. Le lecteur intéressé pourra se référer à [65, p. 49] et sa bibliographie ou encore lire l'article récapitulatif de C. Carlet [32].

### 2.4.5 Compromis sur les propriétés

Idéalement, une fonction utilisée pour un système de chiffrement à flot doit donc cumuler plusieurs propriétés si elle veut conférer au schéma dans lequel elle est utilisée (un système à combinaison de registres en général), une bonne résistance aux attaques connues : degré élevé, équilibre, immunité aux corrélations la plus grande possible et haute non-linéarité. Malheureusement ces propriétés sont le plus souvent incompatibles ce qui oblige le cryptographe à rechercher des compromis. Le plus souvent, lors de la phase de conception de l'algorithme, l'ingénieur compensera ces compromis par des subtilités dans l'implémentation de ces primitives.

### Degré et immunité aux corrélations

La proposition 17 montre qu'une fonction à $n$ variables ne peut à la fois avoir un haut degré et un ordre d'immunité aux corrélations $t$ élevé, puisque son degré est majoré par $n - t$.

### Haute non-linéarité et immunité aux corrélations

Ce compromis a été explicité par C. Fontaine dans sa thèse [65]. Je le reprends ici.

Selon la proposition 16, une fonction est sans corrélation d'ordre $t$ si et seulement si

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

Avec l'équation de Parseval (proposition 13) nous avons alors

$$\sum_{u=0 \text{ ou } wt(u)>t} (\widehat{\chi_f}(u))^2 = 2^{2n} \tag{2.4}$$

et la non-linéarité de la fonction $f$ est

$$nl(f) = 2^{n-1} - \frac{1}{2} \sup_{u=0 \text{ ou } wt(u)>t} (|\widehat{\chi_f}(u)|)\,.$$

Une fonction sans corrélation d'ordre $t$ possède donc une non-linéarité idéale si elle est telle que

$$\begin{aligned} \widehat{\chi_f}(u) &= 0 \text{ pour } 1 \leq wt(u) \leq t \\ \text{et } |\widehat{\chi_f}(u)| &= c \text{ pour } u = 0 \text{ ou } wt(u) > t \end{aligned}$$

où $c$ est une constante. On a par l'équation 2.4 :

$$c^2 = \frac{2^{2n}}{2^n - \sum_{i=1}^{t} \binom{n}{i}}.$$

Pour que la non-linéarité idéale soit atteinte, il faut donc que cette quantité soit un carré parfait, ce qui, $n$ étant fixé, dépend de $t$, l'ordre d'immunité aux corrélations. Dans ce cas, la non-linéarité est égale à la partie entière de $2^{n-1} - \frac{c}{2}$. Si cette quantité n'est pas un carré parfait, on ne peut évaluer la meilleure non-linéarité que l'on peut réellement espérer : on peut seulement avancer $2^{n-1} - \lfloor \frac{c}{2} \rfloor$ comme borne supérieure (borne assez mauvaise). C. Fontaine a tabulé pour quelques valeurs de $n$ et de $t$ les non-linéarités

idéales dans le cas des fonctions sans corrélation d'ordre $t$ et les fonctions $t$-résilientes.

Plus récemment, A. Canteaut, P. Charpin, C. Carlet et C. Fontaine [22, Section 3] ont montré que lorsque la non-linéarité d'une fonction est idéale, son spectre de Fourier est à trois valeurs. Des résultats nouveaux [117, 118, 162] ont également permis d'amélorier les bornes connues sur la non-linéarité dans le cadre d'un compromis avec l'immunité aux corrélations.

#### Fonctions courbes et équilibre

Une fonction booléenne $f$ à $n$ variables est équilibrée si et seulement si $\widehat{\chi_f}(0) = 0$. Or $f$ est courbe si et seulement si $\widehat{\chi_f}(u) = \pm 2^{\frac{n}{2}}$ pour tout $u \in \mathbb{F}_2^n$. Ces deux conditions sont incompatibles.

#### Fonctions courbes et degré

La proposition 19 montre qu'une fonction courbe à $n$ variables est de degré au plus égal à $\frac{n}{2}$.

## 2.5 Variantes et équivalence des systèmes

D'un point de vue général, un système de chiffrement à flot se décompose en trois parties :

- Un moteur, constitué essentiellement de registres à décalage en période maximale. Ces registres peuvent être conventionnels, décimés, contrôlés par une horloge régulière ou irrégulière,... Le but de ce moteur est de fournir une ou plusieurs suites, possédant déja de bonnes propriétés statistiques.
- Les séquences produites par le moteur ayant des propriétés de linéarité plus ou moins fortes, il est indispensable de les gommer. La seconde partie est donc un module dont le rôle est de briser cette linéarité, d'assurer des effets de diffusion (chaque bit de clef influence le plus de bit de sortie possible) et de confusion (on ne peut isoler un "fil" ou une partie du schéma). Ce module (que j'appelerai module NDPC [8]) est en général composé de fonctions booléennes ayant les meilleures propriétés possibles, des permutateurs, des multiplexeurs,...Il peut se réduire, comme dans les systèmes à combinaison de registres, à une seule fonction booléenne.

8. Nonlinéarité, Diffusion, Confusion, Propagation

– Un module de combinaison de la suite chiffrante avec le texte clair. Le plus commun se réduit à une addition modulo 2 (Xor) mais encore une fois des modules plus complexes peuvent exister.

Il est alors possible, pour certains schémas, d'exhiber, par une transformation plus ou moins importante du système, une équivalence avec le système à combinaison de registre. L'approche générale consiste alors à décrire les modules NDPC et de combinaison par une unique fonction booléenne, qui prend en entrée, les suites produites par le moteur.

Je vais présenter quelques uns des systèmes génériques qui peuvent ainsi être ramenés à un système à combinaison de registres. Toutefois, dans la pratique une analyse fine d'un schéma permet assez souvent de se ramener au moins à un équivalent bruité, c'est-à-dire que les parties du schéma qui "résistent" à cette réduction sont souvent assimilables à un bruit de paramètre $p \neq \frac{1}{2}$ que l'on incorpore alors au paramètre du texte clair $p_0$.

### 2.5.1 Systèmes multiplexés

Dans ce type de système, un dispositif appelé *multiplexeur* choisit ses entrées parmi plusieurs disponibles (dans notre cas les sorties de plusieurs registres). Le premier système publié est celui de Geffe [71] (voir figure 2.8). Le choix s'effectue au niveau du registre 2 et la fonction booléenne opérant

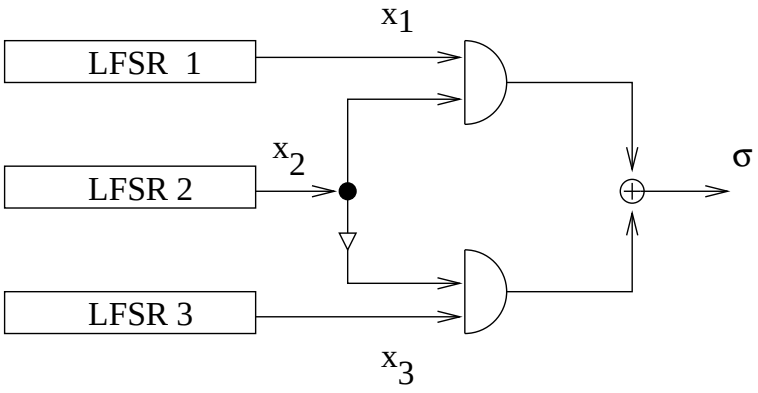


FIGURE 2.8 – Générateur de Geffe

ce choix est donnée sous forme logique par

$$\sigma = x_1.x_2 \oplus x_3.\overline{x_2}$$

ce qui en terme de pseudo-code donne

$$\textbf{if } x_2 \textbf{ then } sigma = x_1 \textbf{ else } sigma = x_3$$

Il est alors possible de se ramener à un système à combinaison de 3 registres en prenant la fonction booléenne

$$f(x_1, x_2, x_3) = x_1x_2 \oplus x_2x_3 \oplus x_3$$

Cette transformation est dans ce cas très facile mais elle illustre parfaitement le type de transformation à faire. A noter que ce générateur a été cassé par Siegenthaler en utilisant précisément cette équivalence. D'autres systèmes multiplexés connus peuvent ainsi être transformés en systèmes à combinaison de registres : le générateur de Pless [138] , cryptanalysé par Rubin [147], le générateur de Jennings [96], cryptanalysable par la méthode présentée dans [176], le générateur de Bruer [16], cassé par Siegenthaler.

### 2.5.2 Registre filtré non linéairement

Ce schéma a été proposé par Siegenthaler [156] et est décrit par la figure 2.9. Un unique registre est utilisé, de longueur $L$, produisant une suite en

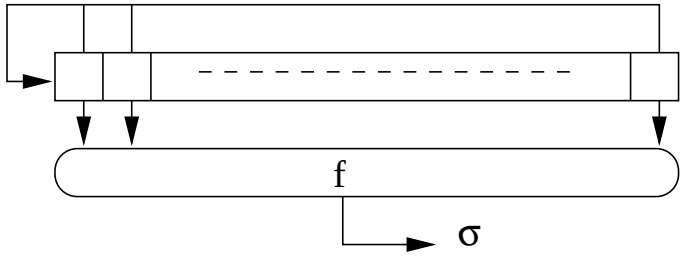


FIGURE 2.9 – Registre filtré

période maximale. Certaines cellules de ce registre sont prises en entrée par une fonction $f$ à $m$ variables. C'est la sortie de la fonction $f$ qui va constituer la suite pseudo-aléatoire, combinée avec le texte clair. Les résultats principaux concernant ce type de systèmes sont résumés par ces deux propositions.

**Proposition 21** *[104](***Borne de Key***)*
*Soit un registre de longueur $L$, de période maximale, filtré par une fonction $f$ de degré $m$. Alors la complexité linéaire de ce système est au plus*

$$\Lambda(s) = \sum_{i=1}^{m} \binom{L}{i}$$

*$\Lambda(s)$ est appelée la borne de Key.*

**Proposition 22** *[148] Pour un registre en période maximale de longueur première L, la fraction des fonctions booléennes f de degré m produisant des séquences de complexité linéaire maximale $\Lambda_m$ est*

$$P_m \approx \exp(\frac{-\Lambda_m}{L.2^L}) > e^{-\frac{1}{L}}$$

*Donc pour des grandes valeurs de L, un générateur filtré produit en général des séquences dont la complexité linéaire atteint la borne de Key.*

Golic [75], en 1996, a donné une cryptanalyse de ces systèmes, baptisée *attaque par inversion* ainsi que certains critères pour la conception de ces systèmes.

T. Siegenthaler a montré [156] que ce schéma peut toujours se ramener à un schéma à combinaison de registres utilisant une fonction de combinaison $g$ à $n$ variables (avec $1 \leq n \leq m$). Les registres ont alors tous le même polynôme de rétroaction. Seuls les états initiaux diffèrent.

**Exemple 3** *[156] Soit $L = 13$ et le système représenté par la figure 2.10. La fonction originale est donnée par*

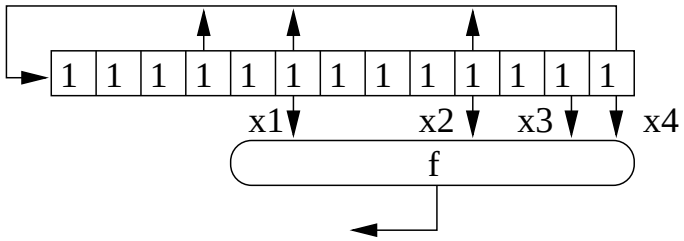


FIGURE 2.10 – Registre filtré de longueur 13

$$f = x_1 \oplus x_4 \oplus x_1x_3 \oplus x_2x_3 \oplus x_1x_4 \oplus x_2x_4.$$

*L'initialisation considérée pour l'exemple de la figure 2.10 est le vecteur tout à 1. Avec seulement 800 bits de texte chiffré, il est possible d'obtenir le système équivalent à combinaison de 3 registres. La nouvelle fonction est à trois variables et est donnée par*

$$g = y_1 \oplus y_2 \oplus y_1y_3 \oplus y_1y_2 \oplus y_2y_3$$

*et les initialisations des trois registres sont :*

- $(0,0,0,0,0,0,1,1,1,1,0,0,0)$ *pour le registre en entrée sur* $y_1$.
- $(0,0,0,0,0,0,1,1,1,1,0,0,1)$ *pour le registre en entrée sur* $y_2$.
- $(0,0,0,0,0,0,0,0,0,0,1,1,0)$ *pour le registre en entrée sur* $y_3$.

R. Forré [68] a modifié l'approche de Siegenthaler pour traiter le cas des registres longs, en appliquant une attaque par corrélation rapide. Notons que pour certains types de registre filtré, E. L. Key a donné des équivalents encore plus simple à un seul registre [104]

### 2.5.3 Registres contrôlés régulièrement

Ces systèmes seront traités en détail dans le chapitre suivant. L'équivalence est rendue possible par le fait que changer la vitesse d'horloge permet de simuler des registres souvent plus courts. Ce qui est équivalent à considérer une décimation de la suite originelle. Cela m'a permis de mettre au point une nouvelle attaque sur les systèmes de chiffrement à flot et de définir un nouveau critère de résistance pour les registres. Cette attaque permet pratiquement, de déterminer les valeurs de degré des polynômes de rebouclage qui rendent vulnérables à cette attaque les schémas existants.

# Chapitre 3

# Attaque par décimation

Lorsque l'on veut cryptanalyser un système de chiffrement par flot, deux approches sont possibles. La première consiste à étudier le schéma tel quel, à rechercher des faiblesses locales (fonction booléenne présentant des corrélations, registres à décalage en période non maximale, clefs faibles,...) et à les exploiter pour obtenir une information sur les bits de clef secrète (l'initialisation des registres), de façon moins coûteuse que la recherche exhaustive. Cette approche est pratiquement la seule considérée dans la littérature. La plus connue en est l'attaque par corrélation de Siegenthaler que je présenterai dans ce chapitre.

La seconde approche consiste à considérer le système non pas tel quel, mais à opérer une *transformation de schéma*. Autrement dit on va modifier le schéma pour obtenir un schéma équivalent sur tout ou partie des données dont on dispose (le texte chiffré) et que l'on cherche (les bits de clef). Cette transformation est opérée de sorte que le travail à fournir pour opérer une cryptanalyse totale soit beaucoup moins important que si l'on considérait l'algorithme original.

Cette approche a été peu utilisée, en tout cas peu publiée. Rares sont les publications traitant et utilisant effectivement la transformation de schéma. On ne peut guère citer que l'attaque des registres filtrés par Siegenthaler [156] ou celle de E. Key [104].

Dans ce chapitre, je vais présenter une attaque que j'ai mise au point utilisant cette approche. Elle exploite une propriété intéressante des registres à décalage. Si l'on effectue une décimation (sélection de bits à intervalles réguliers) de la suite de sortie d'un registre, la sous-suite obtenue peut souvent être directement générée par une registre plus court. Sur ce registre, une attaque de type Siegenthaler devient alors moins coûteuse.

Je vais donc d'abord rappeler en détail l'attaque de Siegenthaler (section 3.1) et succinctement les attaques par corrélation rapides. Puis je présenterai ensuite cette nouvelle attaque que j'ai appelée *attaque par décimation*. La proposition 23 rappellera les résultats concernant la décimation d'une suite générée par un registre à décalage. L'algorithme d'attaque proprement dit sera ensuite exposé. Je présenterai ensuite (proposition 25) le nouveau critère de résistance que j'ai défini permettant de se prémunir contre cette nouvelle attaque. Je comparerai enfin les performances de l'attaque par décimation à celles d'attaques précédentes.

Les résultats de ce chapitre ont été publiés dans [62].

## 3.1 L'attaque par corrélation de Siegenthaler

Cette attaque a été développée par T. Siegenthaler [155]. C'est la première attaque publiée, sur les systèmes de chiffrement par flot. C'est une attaque à chiffré seul, en supposant que le message clair $(m_t)_{t\geq 0}$ est issu d'une source sans mémoire, binaire, dont la sortie vaut 0 avec une probabilité $p_0 = P[m_t = 0]$ (avec en pratique $0.6 \leq p_0 \leq 0.7$). Une attaque à clair connu devient alors un cas particulier en prenant $p_0 = 1$.

Cette attaque exploite l'existence d'une corrélation entre la sortie de la fonction de combinaison $f$ [1] et une fonction linéaire de ses entrées. Une telle corrélation existe toujours, plus ou moins exploitable comme il a été montré dans le chapitre précédent.

**Théorème 4** *(***Cas d'une fonction 1-corrélée***) Soit $f$ une fonction booléenne de combinaison à $n$ variables. Pour $1 \leq i \leq n$, on note*

$$q_i = P[f(x_1, x_2, \ldots, x_n) = x_i]$$

*où les $x_i$ sont $n$ variables aléatoires indépendantes uniformément distribuées dans $\mathbb{F}_2$.*

*Soient $c_1, c_2, \ldots, c_N$, $N$ bits du texte chiffré. Alors l'estimateur $H_i$ entre le chiffré et la sortie $s^i$ du $i$ème registre, défini par*

$$H = hamming((s_t^i)_{t\geq 1}, (c_t)_{t\geq 1}) = \sum_{t=1}^{N} (s_t^i \oplus c_t)$$

*est une variable aléatoire gaussienne suivant une loi $\mathcal{N}(N, p_i)$. où $p_i = 1 - p_0 - q_i + 2p_0q_i$.*

1. Je considèrerai uniquement les systèmes à combinaison de registres. Les résultats de ce chapitre peuvent être appliqués à tous les autres systèmes de chiffrement par flot.

Supposons que le registre $i$ soit de longueur $L_i$. Il faut donc tester les $2^{L_i} - 1$ initialisations possibles. Deux cas se présentent alors :

- L'initialisation est la bonne (hypothèse $\mathcal{H}_0$, celle ayant effectivement généré la suite chiffrée). Alors la variable aléatoire $H$ suit une loi normale $\mathcal{N}(N, p_i)$ avec $p_i \neq \frac{1}{2}$.
- L'initialisation n'est pas la bonne (hypothèse $\mathcal{H}_1$). Alors la suite générée est indépendante de la suite chiffrée et la variable aléatoire $H$ suit une loi $\mathcal{N}(N, \frac{1}{2})$.

Un simple test d'hypothèse [54] permet de choisir entre les deux hypothèses. Un seuil de décision $T$ est fixé et on se fixe comme règle de décision (on suppose que $p_i < \frac{1}{2}$) :

- si $H < T$ alors on est sous $\mathcal{H}_0$ et le candidat est retenu.
- si $H > T$ alors on est sous $\mathcal{H}_1$ et le candidat est rejeté.

Deux risques d'erreurs sont attachés à cette règle de décision :

- Le risque de ne pas détecter le bon candidat (probabilité de non détection $pnd$). Elle est donnée par

$$pnd = P[H > T|\mathcal{H}_0]$$

- Le risque de détecter indûment un candidat (probabilité de fausse alarme $pfa$). Elle est donnée par

$$pfa = P[H < T|\mathcal{H}_1]$$

Ces probabilités vont bien sûr évoluer avec la position de $T$. Il faut donc choisir le seuil de manière à les minimiser (ou à défaut minimiser au moins la $pnd$ car on ne veut pas perdre la bonne initialisation). On peut donc jouer sur le nombre $N$ de bits nécessaires pour mener l'attaque.

En utilisant le théorème central limite, on sait que la variable aléatoire centrée réduite de $H$ sous les deux hypothèses tend vers une loi normale centrée réduite $\mathcal{N}(0, 1)$ de fonction de répartition $\Phi(x)$ donnée par

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp -\frac{x^2}{2}\, dx.$$

Cette approximation est valide dans la mesure où $N$ est grand. On a donc

$$P[\mathcal{N}(0,1) < \frac{T - \frac{N}{2}}{\sqrt{\frac{N}{4}}}] \leq pfa$$

$$P[\mathcal{N}(0,1) > \frac{T - Np_i}{\sqrt{Np_i(1-p_i)}}] \leq pnd$$

Posons $a = \Phi^{-1}(pfa)$, l'abscisse de la loi $\mathcal{N}(0,1)$ pour une $pfa$ fixée et $b = \Phi^{-1}(1-pnd)$, l'abscisse pour cette même loi, pour une $pnd$ fixée. Alors nous obtenons deux équations à deux inconnues :

$$\begin{aligned} -a &= \frac{T - \frac{N}{2}}{\sqrt{\frac{N}{4}}} \\ b &= \frac{T - Np_i}{\sqrt{Np_i(1-p_i)}} \end{aligned}$$

Ce qui finalement donne :

$$\begin{aligned} T &= \frac{N}{2} - a\sqrt{\frac{N}{4}} \\ N &= \left( \frac{2b\sqrt{p_i(1-p_i)} + a}{1-2p_i} \right)^2 \end{aligned}$$

L'attaque par corrélation de Siegenthaler permet donc de retrouver l'initialisation des registres en

$$\prod_{i,p_i=\frac{1}{2}} (2^{L_i-1}) + \sum_{i,p_i\neq\frac{1}{2}} (2^{L_i-1})$$

essais systématiques alors qu'une attaque exhaustive en aurait nécessité $\prod_{i=1}^{n}(2^{L_i}-1)$.

Il est facile de voir que cette attaque est très vite limitée par la taille du registre auquel on s'attaque. En pratique, dès que $L > 64$, cela devient difficile. D'autres attaques ont été développées, beaucoup plus efficaces, les *attaques par corrélation rapides*.

## 3.2 Les attaques par corrélation rapide

Je présente ici très succinctement les attaques par corrélation rapide. Ma présentation est celle de A. Canteaut et M. Trabbia (voir [25] pour une présentation plus détaillée).

Ce type d'attaque a été introduit par Meier and Staffelbach [125] et fonctionne sur le même principe que l'attaque de Siegenthaler. Elle permet cependant d'éviter de passer en revue toutes les initialisations possibles des registres étudiés. Si l'on considère $k$ registres simultanément (parce que la fonction est corrélée d'ordre $k$ ou encore CI(k-1)), on a vu dans le chapitre

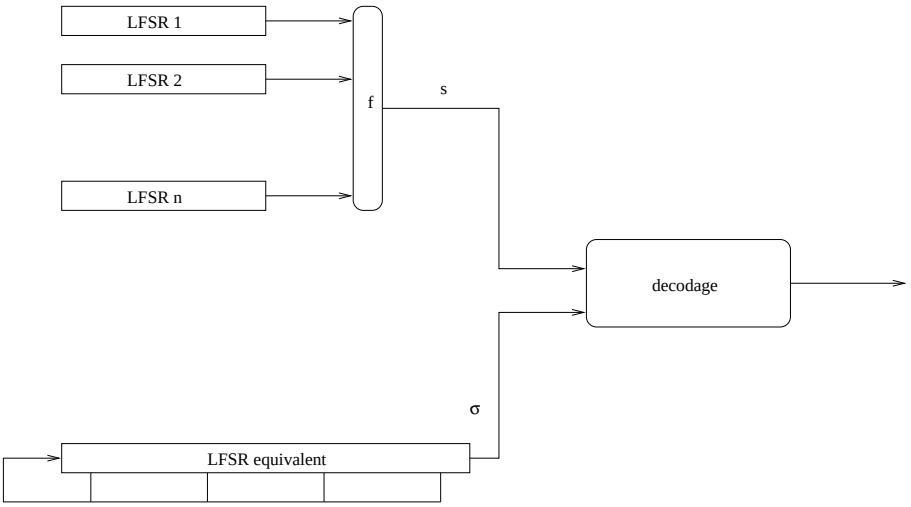


FIGURE 3.1 – Principe de l'attaque par corrélation rapide

précédent que l'on peut assimiler la suite $\sigma$ produite à une suite générée par un unique registre de longueur $L$ avec

$$L = g(L_{i_1}, L_{i_2}, \ldots, L_{i_k})$$

et dont on connait le polynôme de rétroaction $P$ (calculé à partir des $P_{i_1}, \ldots, P_{i_k}$). On note $d$ le nombre de coefficients de rétroaction non nuls de ce registre. On écrit alors

$$P(X) = 1 \oplus \bigoplus_{i=1}^{d} X^{l_i}.$$

Pour déterminer entièrement le générateur engendrant la suite $\sigma$, il suffit donc de déterminer $L$ bits consécutifs de $\sigma$. Pour une attaque à clair connu [2] on dispose de $N$ bits de la suite $s$ correspondant à la suite $\sigma$ bruitée (voir figure 3.1) :

$$\forall t \geq 0, \quad \sigma_t = s_t \oplus e_t$$

où $e_t$ représente le bit de bruit à l'instant $t$. Cela revient donc à considérer un code de longueur $N$ et de dimension $L$.

La probabilité d'erreur $p$ est donnée par

$$p = P[e_t = 1] = P[f(X_1, X_2, \ldots, X_n) \neq g(X_{i_1}, \ldots, X_{i_k})] = p_g$$

2. Cette technique se généralise sans problème à une attaque à chiffré seul. On utilise seulement en plus la probabilité $p_0$ pour le clair de valoir 0.

L'idée de l'attaque consiste à utiliser un algorithme de décodage rapide afin de retrouver $\sigma$ à partir de $s$. Chaque bit $\sigma_t$ de la suite $\sigma$ vérifie par définition l'équation de parité définie par le polynôme de rétroaction du registre :

$$\forall L \leq t \leq N, \;\; \sigma_n \oplus \bigoplus_{i=1}^{d} \sigma_{t-l_i} = 0.$$

Diverses méthodes permettent d'augmenter le nombre d'équations : élévation à une puissance $j$ de 2 (des équations de parité supplémentaires de longueur $2^j L$ sont alors obtenues), en recherchant des multiples de petits poids $d$, ..., le but étant d'obtenir le plus d'équations possibles, c'est-à-dire le plus d'information possible pour un bit donné.

Différents algorithmes de décodage sont alors utilisés selon les auteurs mais le principe reste le même. Pour chaque bit $\sigma_t$ de la suite $\sigma$, le nombre d'équations de parité vérifiées par la suite $s$ est calculé. Si ce nombre est insuffisant, le bit $s_n$ est probablement faux. Cela donne alors l'algorithme suivant (table 3.1) : Le nombre d'équations de parité détermine directe-

Pour $t$ variant de 0 à $N-1$
- Calcul du nombre $D$ d'équations de parité satisfaites parmi celles concernant le bit $t$.
- Si $D \leq T$, alors $s\prime = s_t \oplus 1$
  Sinon $s\prime = s_t$.
- $s \leftarrow s\prime$

TABLE 3.1 – Une itération de l'algorithme de décodage

ment la convergence de l'algorithme vers la solution recherchée. Elle dépend également de l'algorithme de décodage. Pour un algorithme de décodage itératif de Gallager [70] modifié, on a par exemple le résultat suivant

**Théorème 5** *[25](*Condition de convergence*) L'algorithme de décodage converge si et seulement si le nombre $m(d)$ d'équations de poids $d$ pour chaque bit vérifie*

$$m(d) > \frac{K_d}{C_{d-2}(p)}$$

*où $C_{d-2}(p)$ est la capacité du canal binaire symétrique de probabilité d'erreur $p_{d-2} = \frac{1}{2}(1-(1-2p)^{d-2})$. Donc*

$$C_{d-2}(p) = 1 + p_{d-2}\log_2(p_{d-2}) + (1-p_{d-2})\log_2(1-p_{d-2})$$

$K_d = \simeq 1$ *si* $d \geq 4$ *et* $K_3 \simeq 2$.

*Le nombre de bits* $N$ *de la suite* $s$ *nécessaires pour obtenir ce nombre d'équations est alors de l'ordre de*

$$N = 2^{\alpha_d(p)+\frac{L}{d-1}} \text{ avec } \alpha_d(p) = \frac{1}{d-1}\log_2[(d-1)!\frac{K_d}{C_{d-2}(p)}]$$

Ces attaques ont été introduites par Meier et Staffelbach [125]. Différentes améliorations ont ensuite été publiées [42, 129, 136]. Récemment T. Johansson et F. Jönsson ont proposé d'utiliser les codes convolutifs ou des turbo-codes pour améliorer sensiblement cest attaques [100, 101]. Plus récemment, une autre attaque très efficace utilise des multiples polynomiaux de poids $d = 3, 4$ ou 5 et le décodage itératif de Gallager [25]. Enfin, une nette amélioration de ces attaques [41] passe par la construction et l'utilisation d'un code de dimension plus petite que celle du code de départ afin de jouer sur la complexité du décodage.

## 3.3 Décimation de registres à décalage

Je vais maintenant rappeler les résultats théoriques que je vais utiliser pour mon attaque. Je considère une séquence $\sigma = \sigma_1, \sigma_2, \ldots,$ produite par une registre de longueur $L$ dont le polynôme de rétroaction est irréductible dans $GF(q)[X]$.

Supposons maintenant que l'on opère un échantillonnage sur $\sigma$ à une fréquence $d$ (cela correspond à un cadencement $d$ fois plus lent, ou encore à un contrôle d'horloge tous les $d$ cycles) produisant une sous-séquence $\rho = \rho_1, \rho_2, \ldots$. En d'autres termes, cela est équivalent à une $d$-décimation de la séquence originelle $\sigma$. Nous avons donc $\rho_i = \sigma_{dj}$ pour $j = 0, 1, 2, \ldots$ ou, en notation de séquence $\rho = \sigma[d]$.

**Définition 24** *On appelle* décimation *d'ordre* $d$ *d'une séquence* $(\sigma_t)_{t\geq 0}$ *(ou encore* $d$*-décimation) l'opération consistant à prélever un bit de cette séquence tous les* $d$ *bits. La suite obtenue, notée* $\rho = \sigma[d]$ *est donnée par :*

$$(\rho_i)_{i\geq 0} = \sigma[d] = (\sigma_{dj})_{j\geq 0}$$

Avec un contrôle d'horloge à intervalle $d$, le registre de départ se comportera comme un registre différent, appelé *registre simulé* [148]. La question intéressante est de déterminer ses propriétés, spécialement relativement au registre de départ. Elles sont résumées dans la proposition suivante

**Proposition 23** *[148, page 146] Soit $\sigma$ une séquence produite par un registre dont le polynôme de rétroaction $P(x)$ est irréducible dans $GF(q)$, de degré $L$. Soient $\alpha$ une racine de $P(x)$ et $T$ la période de $P(x)$. Soit $\rho$ la séquence résultant d'une d décimation de $\sigma$, i.e. $\rho = \sigma[d]$. Alors le registre simulé, c'est-à-dire le registre produisant directement $\rho$ possède les propriétés suivantes :*

1. *Le polynôme de rétroaction $P^*(x)$ du registre simulé est le polynôme minimal de $\alpha^d$ dans $GF(q^L)$.*
2. *La période $T^*$ de $P^*(x)$ est égale à $\frac{T}{\gcd(d,T)}$.*
3. *Le degré $L^*$ de $P^*(x)$ est égal à l'ordre multiplicatif de $q$ dans $\mathbf{Z}_{T^*}$.*

*De plus tout $d$ appartenant à $C_k$, où $C_k = \{k, kq, kq^2, \ldots, \} \bmod T$ désigne la classe cyclotomique de $k$ modulo $T$, fournit le même registre simulé, sauf pour certaines initialisations. Finalement, toute séquence produite par le registre simulé est égale à $\sigma[d]$.*

Dans les cas pratiques, $P(x)$ est primitif et donc $T = 2^L - 1$. Alors quand $T$ n'est pas premier, par un choix judicieux de $d$ tel que $\gcd(2^L - 1, d) \neq 1$, on peut espérer obtenir un registre simulé plus court que le registre de départ.

**Exemple 4** *[148] Considérons un registre avec $P(x) = X^4 + X + 1$, $P$ est primitif, $T = 15$ et $L = 4$. Soit $\alpha$ une racine primitive de $P$.*

- *$d \in C_3 = \{3, 6, 9, 12\}$, $T^* = 5$ et $L^* = 4$ puisque l'ordre multiplicatif de 2 dans $\mathbf{Z}_5$ est 4 ($\alpha^3$ appartient à un sous-groupe d'ordre 5).*
- *$d \in C_5 = \{5, 10\}$, $T^* = 3$ et $L^* = 2$ puisque l'ordre multiplicatif de 2 dans $\mathbf{Z}_3$ est 2 (et $P^*(x) = X^2+X+1$) ($\alpha^5$ appartient à un sous-groupe d'ordre 4).*

Le polynôme de rétroaction $P^*(x)$ du registre simulé peut être obtenu en appliquant l'algorithme de Berlekamp-Massey [122] sur la séquence $\rho$.

## 3.4 L'attaque par décimation

### 3.4.1 Description de l'algorithme

Une décimation d'ordre $d$ considère un bit sur $d$ ce qui a pour effet d'augmenter la taille de séquence de départ si l'on veut disposer d'une sous-suite de longueur suffisante. Afin de disposer d'une attaque plus réaliste et opérationnelle, je considère uniquement une attaque à chiffré seul. Cela limitera la relative augmentation de la suite nécessaire.

J'utiliserai le modèle du canal binaire symétrique (BSC) avec une probabilité d'erreur $p$ qui modélisera simultanément la corrélation de la fonction booléenne $P[x_t = \sigma] = p_\sigma$ et la probabilité du clair $p_0 = P[m_t = 0]$. Je prendrai $p_0 = 0.65$ ce qui correspond à la plupart des cas pratiques. On a alors $p = p_0 + p_\sigma - 2.p_0 p_\sigma$.

Soit un registre cible de longueur $L$ tel qu'il existe $d|2^L - 1$ satisfaisant la proposition 23. Je montrerai (section 3.5) que cela est le cas quand $L$ n'est pas premier. Alors, dans ce cas, le registre simulé est de longueur au plus $\frac{L}{2}$ (j'ai effectué un calcul exhaustif de toutes les valeurs utiles $d$ pour $L < 256$. Certaines sont données en annexe A. Les autres sont disponibles dans [56].)

Les étapes principales de l'algorithme sont alors les suivantes (je me place pour le moment dans le cas d'une fonction 1-corrélée dans un but de simplicité ; le cas des fonctions d'ordre de corrélation plus élevé sera présenté plus loin) :

### Extraction de la suite décimée

À partir de la suite chiffrée $\sigma$, j'extrais la sous-suite de chiffré $\rho = \sigma[d]$ obtenue par décimation tous les $d$ bits. Je noterai $\sigma_i[d]$ la sous-suite obtenue par $d$-décimation à partir de l'index de temps $i$. Sans autre précision, $\rho = \sigma[d]$ correspond dans la suite à $\sigma_0[d]$.

Le choix du paramètre de décimation $d$ est dicté par le compromis entre la taille du registre simulé $L^*$ la plus petite possible et la longueur résultante de la taille de chiffré nécessaire. Les différentes simulations ont montré qu'en règle générale, la valeur de $d$ réalisant ce meilleur compromis est celle donnant $L^* = \frac{L}{2}$ ou $L^* = \frac{L}{3}$ selon la parité de $L$, $L$ non premier.

### Calcul du polynôme $P^*(X)$

Connaissant le polynôme de départ $P(X)$, je choisis une initialisation aléatoire, génère une suite de sortie $x$ et la décime selon le même facteur $d$ obtenant la sous-suite $x[d]$. L'algorithme de Massey-Berlekamp appliqué sur la suite $x[d]$ permet alors de retrouver $P^*(X)$, polynôme de rebouclage du registre simulé. Il n'est en fait que le polynôme minimal de $\alpha^d$ où $\alpha$ est une racine primitive de $P$. La proposition 23 assure que ce polynôme est identique quelle que soit l'initialisation de $L$ bits choisie au départ pour le registre d'origine.

### Attaque du registre simulé

A l'aide de la séquence décimée $\rho = \sigma[d]$ je vais tenter de retrouver la séquence de sortie correspondante $y$ du registre simulé, qui est également la séquence décimée $x[d]$ du registre (cible) de départ. J'utilise alors une simple attaque de Siegenthaler 3.1 appliquée sur le registre simulé.

Cela ne requiert que $2^{L^*}$ essais systématiques. Je calcule pour chacune des $2^{L^*}$ initialisations possibles la valeur de l'estimateur

$$E = \sum_t (x^t[d] \oplus \sigma^t[d] \oplus 1)$$

Pour l'initialisation correcte de $L^*$ bits, $E$ suit une loi normale de valeur moyenne $N.(1-p)$ et de variance $\sigma^2 = N.p.(1-p)$ (hypothèse $\mathcal{H}_0$) tandis que pour les mauvaises initialisations, $E$ suit une loi normale de valeur moyenne $N.\frac{1}{2}$ et de variance $\sigma^2 = N.\frac{1}{4}$ (hypothèse $\mathcal{H}_1$) où $N$ est le nombre minimum de bits de chiffré nécessaires de $\sigma[d]$.

Pour discriminer $\mathcal{H}_0$ de $\mathcal{H}_1$, comme pour l'attaque de Siegenthaler, je fixe un seuil de décision $\mathcal{T}$. Si $|E| > \mathcal{T}$ alors $\mathcal{H}_0$ est acceptée et l'initialisation est conservée sinon $\mathcal{H}_1$ est choisie et l'initialisation est rejetée.

Le nombre minimum $N$ de bits de $\sigma[d]$ dépend du nombre de fausses décisions que j'accepte. Ce nombre (ainsi que le seuil $\mathcal{T}$) est déterminé par la probabilité de fausse alarme $pfa$ et la probabilité de non déctection $pnd$ (voir section 3.1). On a alors, en utilisant les notations de la section 3.1 :

$$N = \left( \frac{2b\sqrt{p(1-p)} - a}{1-2p)} \right)^2 \tag{3.1}$$

$$\mathcal{T} = \frac{1}{2}\left(a\sqrt{N} + N\right) \tag{3.2}$$

En terme de suite chiffrée, il faut donc considérer $N.d$ bits. Les quelques candidats retenus sont mémorisés avec la valeur correpondante de l'estimateur $E$, dans l'ordre décroissant de ces valeurs. En pratique, pour les erreurs retenues, le candidat de $L^*$ bits est dans les tout premiers scores.

### Détermination des bits de clef restants

Chaque candidat de $L^*$ bits retenu à l'étape précédente est utilisé pour générer à l'aide du registre simulé, les $L$ premiers bits (supposés) de $x[d]$.

Comme chaque bit de la séquence $x[d]$ est aussi un bit de la séquence $x$, il peut être décrit par deux équations différentes. L'une est en $L^*$ variables (le bit est vu comme un bit de la séquence $x[d]$). L'autre est en $L$ variables

(le bit est vu comme appartenant à la séquence $x$). L'exemple 5 illustre cela. On peut donc avec les $L$ bits de $x[d]$ établir un système de $L$ équations à $L$ inconnues. Ce système est de façon évidente de rang $L^*$. En considérant $L^*$ variables principales, on peut inverser le système et exprimer ces $L^*$ variables principales en fonction des $L - L^*$ autres variables (considérées comme paramètres). Une recherche supplémentaire exhaustive sur ces $L-L^*$ paramètres permet de retrouver une initialisation de $L$ bits.

Chaque initialisation de $L$ bits dans l'ordre décroissant de leur score (valeur de l'estimateur $E$) est testée en calculant

$$E' = \sum_t (x^t \oplus \sigma^t \oplus 1)$$

Celle donnant le plus grand score est retenue comme l'initialisation cherchée. L'attaque est achevée.

#### Abaissement de la probabilité de fausse alarme (pfa)

Afin d'obtenir une probabilité de fausse alarme (pfa) finale proche de zéro (c'est-à-dire une seule solution de $L^*$ bits) et diminuer la complexité de la deuxième phase de recherche exhaustive, j'effectue un travail similaire sur quelques autres séquences décalées et décimées

$$\sigma_0[d], \sigma_1[d], \ldots, \sigma_k[d]$$

où $\sigma_i[d]$ indique que l'on décime $\sigma$ à partir de la position $i$ ($0 \leq i < d$). Pour chaque position $i$, je conserve les meilleures initialisations et calcule l'initialisation correspondant à la position $i = 0$. Les deux initialisations de $L^*$ bits $I_0$ et $I_i$ générant respectivement les suites décimées $\sigma_0[d]$ et $\sigma_i[d]$ sont liées par la relation [110, p. 400, lemme 8.2]

$$I_0 = A \cdot I_i$$

où $A$ est une matrice inversible de rang $L^*$ [110]. La bonne initialisation de $L^*$ bits recherchée figurera simultanément dans plusieurs listes et sera ainsi facilement détectée par des techniques simples de tri. J'ai pu constater que $2 \leq k \leq 8$ convenait la plupart du temps.

### 3.4.2 Résultats de simulation

Je vais présenter quelques résultats de simulation afin d'illustrer la présentation faite dans la section précédente.

**Exemple 5** Fonction CI(0).
*Considérons un registre de longueur* $L = 40$ *de polynôme de rétroaction :*

$$\begin{aligned} P(x) &= 1 \oplus x^2 \oplus x^3 \oplus x^4 \oplus x^5 \oplus x^6 \oplus x^8 \oplus x^{11} \oplus x^{12} \oplus x^{15} \oplus x^{17} \oplus x^{19} \oplus \\ &\quad x^{22} \oplus x^{24} \oplus x^{25} \oplus x^{26} \oplus x^{27} \oplus x^{28} \oplus x^{29} \oplus x^{35} \oplus x^{40} \end{aligned}$$

*Considérons un BSC de probabilité* $p = 0.49$ ($P_0 = 0.65$ *et* $P[x_t = y_t] = 0.534$) *modélisant une fonction booléenne CI(0) et le texte clair en même temps. Comme* $2^{40} - 1 = 3.5^2.11.17.31.41.61681$, *en prenant* $d = 1,048,577$ *pour facteur de décimation pour la séquence* $\sigma$ *produite par le registre, j'obtiens un registre simulé plus court de longueur* $L^* = 20$ *ayant pour polynôme de rétroaction :*

$$P^*(x) = 1 \oplus x \oplus x^2 \oplus x^3 \oplus x^6 \oplus x^7 \oplus x^8 \oplus x^{11} \oplus x^{12} \oplus x^{13} \oplus x^{15} \oplus x^{18} \oplus x^{20}$$

*Avec une* $pnd = 0.1$ *et une* $pfa = 0.1$, *j'ai donc besoin (équation 3.2), de* $N = 13050$ *bits de séquence décimée* $\rho = \sigma[d]$ *c'est-à-dire* $N.d = 2^{33}$ *bits de chiffré. Pour retrouver la bonne initialisation, 15 minutes sur un PII 400 Mhz avec moins de 1 Mo de mémoire ont été nécessaires.*

*Notons que le 20,971,541ème bit (e.g.), lorsque vu comme appartenant à la séquence* $x$ *est décrit par l'équation à 40 variables suivante (où* $x_i$ $0 \leq i \leq 39$ *représente le* $i$*ème bit inconnu de l'initialisation de 40 bits) :*

$$\begin{aligned} b_{20,971,541} &= x_2 \oplus x_4 \oplus x_6 \oplus x_7 \oplus x_{14} \oplus x_{16} \oplus x_{17} \oplus x_{19} \oplus x_{20} \oplus x_{22} \oplus x_{23} \\ &\quad \oplus x_{25} \oplus x_{26} \oplus x_{28} \oplus x_{29} \oplus x_{30} \oplus x_{33} \oplus x_{36} \oplus x_{37} \end{aligned}$$

*tandis, que lorsque vu comme le 21ème bit de* $x[d]$, *nous avons l'équation suivante (20 variables) (où* $y_i$ $0 \leq i \leq 19$ *représente le* $i$*-ème bit de l'initialisation de* $L^*$ *bits) :*

$$b_{20,971,541} = y_0 \oplus y_1 \oplus y_2 \oplus y_3 \oplus y_6 \oplus y_7 \oplus y_8 \oplus y_{11} \oplus y_{12} \oplus y_{13} \oplus y_{15} \oplus y_{18}$$

**Exemple 6** Fonction CI(1).
*Considérons maintenant deux registres* $P_1$ *de longueur* $L_1 = 34$ *et* $P_2$ *de longueur* $L_2 = 39$ *combinés par une fonction booléenne CI(1).*

$$\begin{aligned} P_1(x) &= 1 \oplus x^4 \oplus x^5 \oplus x^6 \oplus x^7 \oplus x^{12} \oplus x^{13} \oplus x^{14} \oplus x^{15} \oplus x^{16} \oplus x^{18} \oplus x^{20} \\ &\quad \oplus x^{21} \oplus x^{22} \oplus x^{24} \oplus x^{25} \oplus x^{29} \oplus x^{30} \oplus x^{31} \oplus x^{33} \oplus x^{34} \\ P_2(x) &= 1 \oplus x \oplus x^2 \oplus x^3 \oplus x^4 \oplus x^5 \oplus x^{14} \oplus x^{15} \oplus x^{16} \oplus x^{17} \oplus x^{19} \oplus x^{21} \oplus \\ &\quad x^{24} \oplus x^{21} \oplus x^{24} \oplus x^{27} \oplus x^{28} \oplus x^{29} \oplus x^{31} \oplus x^{32} \oplus x^{36} \oplus x^{37} \oplus x^{39} \end{aligned}$$

*Prenons le même modèle de BSC que dans l'exemple 5 mais pour une fonction CI(1). Cela implique que nous devons considérer toutes les initialisations d'au moins deux registres simultanément. Le meilleur facteur de décimation d dans ce cas est alors* $d = 131,073$ *qui divise* $2^{34} - 1$*. Alors, nous obtenons deux nouveaux registres simulés :*

$$\begin{aligned} P_1^*(x) &= 1 \oplus x \oplus x^2 \oplus x^5 \oplus x^6 \oplus x^7 \oplus x^9 \oplus x^{10} \oplus x^{11} \oplus x^{12} \oplus x^{13} \oplus x^{15} \oplus x^{17} \\ P_2^*(x) &= 1 \oplus x \oplus x^4 \oplus x^6 \oplus x^7 \oplus x^{14} \oplus x^{16} \oplus x^{17} \oplus x^{18} \oplus x^{20} \oplus x^{22} \oplus x^{24} \\ &\quad \oplus x^{28} \oplus x^{30} \oplus x^{32} \oplus x^{34} \oplus x^{39} \end{aligned}$$

*Avec* $pnd = 0.1$ *et* $pfa = 0.1$*, nous avons besoin de (équation 3.2),* $N = 13050$ *bits de séquence décimée* $\rho = \sigma[d]$ *c'est-à-dire* $N.d = 2^{30}$ *bits de chiffré. La complexité est alors en* $\mathcal{O}(2^{56})$ *avec moins de 1 Mo de mémoire. Des simulations partielles (*$k = 8$*) m'ont permis, sur un PII 400 Mhz, de vérifier que la* $pfa$ *finale était effectivement proche de 0.*

## 3.5 Critère de résistance

Je vais maintenant définir un critère permettant de préciser quand cette attaque est inopérante malgré la présence d'une corrélation exploitable.

Par une application directe de la proposition 23, on peut en première approche, définir ce critère ainsi :

**Proposition 24** *Soit* $L \in \mathbb{N}$*. Soit un registre* $R$ *quelconque de polynôme de rétroaction de degré* $L$ *et une fonction* $f$ *corrélée à ce registre.* $R$ *résiste à l'attaque par décimation si et seulement si* $\forall d < 2^L - 1$ *tel que* $d|(2^L - 1)$*, l'ordre multiplicatif de 2 dans* $\mathbb{Z}_{T^*}$ *est égal à* $L$ *où* $T^* = \frac{2^L - 1}{\gcd(2^L - 1, d)}$*.*

De façon évidente, on a alors

**Corollaire 1** *Si* $T = 2^L - 1$ *est premier alors tout registre de longueur* $L$ *résistera à l'attaque par décimation.*

En fait, la théorie des corps finis permet de préciser plus simplement ce critère.

**Théorème 6** *(Critère de sous-corps) [110, p. 49] Soit* $\mathbb{F}_q$ *un corps fini à* $q = p^n$ *éléments. Alors tout sous-corps de* $\mathbb{F}_q$ *est d'ordre* $p^m$*, avec* $m$ *un diviseur positif de* $n$*. Inversement, si* $m$ *est un diviseur positif de* $n$*, alors il existe exactement un sous-corps de* $\mathbb{F}_q$ *à* $p^m$ *éléments.*

Le critère de résistance peut alors être défini ainsi :

**Proposition 25** *Tout registre de longueur $L$ résiste à l'attaque par décimation si et seulement si $L$ est premier.*

*Preuve.*
immédiate par application du théorème 6 et de la proposition 24. ■

En d'autres termes, la résistance est assurée dès que le corps $\mathbb{F}_{2^L}$ ne contient aucun sous-corps autre que $\{0, 1\}$. Il devient alors évident que, lorsque cette attaque est possible, il existe une valeur de $L^*$ au plus égale à $\frac{L}{2}$.

Un critère de *primalité relative* entre les longueurs de registres, lorsque ceux-ci sont sommés, était déja connu :

**Théorème 7** *[110, p. 426] Pour tout $i = 1, 2, \ldots, h$, soit $\sigma_i$ une séquence en période maximale dans $\mathbb{F}_q$ de période $r_i$. Si $r_1, r_2, \ldots, r_h$ sont premiers entre eux deux à deux, alors la période de la somme $\sigma_1 + \sigma_2 + \cdots + \sigma_h$ est égale au produit $r_1 r_2 \ldots r_h$.*

Ce nouveau critère de primalité absolue permet donc de déterminer les paramètres qui affaiblissent un schéma utilisant des registres à décalage.

Lors de la conception d'un système de chiffrement par flot, il faut soigneusement choisir le degré des polynômes de rétroaction (en liaison avec l'ordre de corrélation des fonctions utilisées) si l'on veut résister à cette attaque. La table 3.2 donnes quelques valeurs de $L$ satisfaisant ce critère de résistance ("premier" se rapporte au corollaire 1 autrement dit $2^L - 1$ est premier ; "ordre" concerne la proposition 24, c'est-à-dire que $2^L - 1$ est composite avec $L$ premier).

| Commentaires | $L$ |
|---|---|
| Premier | 5 - 7 - 13 - 17 - 19 - 31 - 61 - 89 - 107 - 127 |
| Ordre | 11 - 23 - 29 - 37 - 41 - 43 - 47 - 53 - 59 - 67 - 71 - 73<br>83 - 97 - 101 - 109 - 113 - 131 - 139 - 149 - 151 - 157<br>163 - 167 - 173 - 178 - 179 - 181 - 191 - 193 - 197 - 199<br>211 - 223 - 227 - 229 - 233 - 239 - 241 - 251 |

TABLE 3.2 – Valeurs de $L < 256$ résistant à l'attaque par décimation

J'ai établi une liste complète des paramètres $d$, $T^*$ et $L^*$ jusqu'à $L = 256$ que l'on peut consulter dans [56].

## 3.6 Comparaison avec les attaques récentes

### 3.6.1 Attaque Chepyzhov/Johansson/Smeets (CJS)

Ces auteurs ont proposé dans [41] une idée nouvelle permettant d'améliorer les attaques par corrélation rapides. L'idée de base est de combiner des paires d'équations de parité décrivant le $[N, L]$ code dont les mots sont constitués de toutes les suites de longueur $N$ produites par un registre de longueur $L$ donnée, code qui passe dans un canal binaire symétrique de paramètre $p = \frac{1}{2} - \epsilon$. Le but est d'obtenir une nouvelle équation (par addition modulo 2 des deux équations formant la paire) faisant intervenir seulement les $K$ premiers bits de l'initialisation. Autrement dit, on cherche à obtenir de nouvelles équations de parité modélisant un $[n_2, K]$ code linéaire passant dans un canal de paramètre $p_2$.

La conséquence est que la suite nécessaire $n_2$ sera plus longue c'est-à-dire $n_2 > N$ et un facteur additionnel de bruit devra être considéré ce qui implique que $p_2 = 2p(1-p) = \frac{1}{2} - 2\epsilon^2 > p$. Un compromis doit donc être recherché puisque diminuer $K$ (pour aboutir à une complexité moindre) provoque une augmentation de $n_2$ et du paramètre d'erreur $p_2$.

Si on considère un registre de longueur $L$ et un canal binaire symétrique de paramètre $p = \frac{1}{2} - \epsilon$. Pour $K$ fixé la séquence de suite chiffrante (les auteurs travaillent à clair connu) nécessaire doit être de longueur $N$ donnée par :

$$N \approx \frac{1}{2}\sqrt{K(\log(2)}\epsilon^{-2}2^{\frac{L-K}{2}}.$$

La complexité de la phase de décodage est alors d'ordre $\mathcal{O}(2^k \cdot n_2)$ où $n_2$ a une espérance mathématique donnée par

$$E(n_2) = \frac{N(N-1)}{2}2^{-(L-K)}. \tag{3.3}$$

La complexité de la phase de précalcul est négligeable (du moins dans le cas de paires d'équations de parité). La mémoire nécessaire est au plus de $n_2(K + 2\log_2 N)$ bits (essentiellement il s'agit de la matrice génératrice $G_2$ du nouveau code $[n_2, K]$). Ainsi en prenant $K$ petit, le facteur en $2^K$ dans l'expression de la complexité est réduit mais en contrepartie cela se fait au prix d'une augmentation de la taille de séquence nécessaire $n_2$ et donc en final $N$.

La comparaison de cette attaque avec l'attaque par décimation n'est pas aisée, en particulier car certaines données, notamment celles concernant la détermination du taux de succès de l'attaque CJS ne sont pas explicitées. Les éléments de comparaison suivant peuvent toutefois être donnés.

– Le grand avantage de l'attaque CJS est de fonctionner même pour des registres de longueur première, ce qui n'est pas le cas de l'attaque par décimation. Toutefois, le précalcul (pour obtenir la matrice génératrice $G_2$) doit être refait pour tout nouveau polynôme. Ces données nécessitent beaucoup plus de mémoire que n'en requiert l'attaque par décimation.
– L'absence de données sur le taux d'erreur dans [41] ne permet pas de comparaison autrement que sur les exemples donnés dans cet article. En particulier, les auteurs travaillent à clair connu ce qui constitue une hypothèse moins réaliste que l'hypothèse à chiffré seul, retenue pour l'attaque par décimation. Sur l'exemple présenté dans [41], l'attaque par décimation a été convertie dans l'hypothèse de clair connu. Les résultats sont donnés dans le tableau 3.3.

| | Attaque par Décimation | Attaque CJS |
|---|---|---|
| Probabilité d'erreur du BSC | 0.45 | 0.45 |
| Longueur de séquence $N$ | $2^{29}$ | $2^{23}$ |
| Probabilités de succès | **0.9** | **0.5** |
| Complexité | $2^{30}$ | $2^{25}$ |

TABLE 3.3 – Comparaison de l'attaque par décimation avec l'attaque CJS

– En l'absence de données supplémentaires, il semble que l'attaque par décimation réussit plus souvent que l'attaque CJS, pour des séquences de même ordre de longueur. L'analyse, notamment en terme de longueur de séquence, indique (équation 3.3) que l'attaque CJS est beaucoup plus sensible à de petites variations du paramètre d'erreur du canal (de plus l'attaque de Siegenthaler est optimale car elle consiste en un décodage à maximum de vraisemblance [41]). Cela implique que pour des taux de bruit sensiblement plus élevés (par exemple lorsque à clair inconnu) l'attaque par décimation est plus efficace.
– Enfin, il faut remarquer que l'attaque CJS ne retrouve qu'un petit nombre $K$ de bits avec une probabilité de succès relativement faible. La faible valeur de $K$ est précisément là le facteur de gain de cette attaque. Cela signifie qu'il faut encore trouver $L-K$ bits. Cette valeur est d'autant plus grande que $K$ est petit. La gain en complexité est donc reporté sur cette phase additionnelle. Les auteurs n'évoquent que très rapidement ce problème. Ils suggèrent de répéter autant de fois que nécessaire la phase de décodage pour les autres groupes de $K$ bits. Or la probabilité de succès finale sera le produit des probabilités de

succès de ces décodages successifs. En moyenne, elle sera donc très faible.

En final, si l'attaque CJS reste une méthode élégante, l'attaque par décimation semble plus efficace, en particulier par son taux de succès. Rappelons que cette comparaison n'a de sens que lorsque $L$, la longueur du registre, n'est pas premier. Sinon, l'attaque CJS est la seule efficace.

### 3.6.2 Attaque Canteaut/Trabbia

A. Canteaut et M. Trabbia (CT) dans [25] ont récemment présenté l'une des meilleures attaques connues pour les systèmes de chiffrement par flot. Ils ont considéré des équations de parité de poids $\delta = 4, 5$ en utilisant le décodage itératif de Gallager.

Toutefois, le principal inconvénient de leur approche, demeure l'énorme quantité de mémoire requise à la fois pour la phase de précalcul (génération des équations de parité) et la phase de décodage. Cette dernière, quoique très efficace, requiert en particulier une complexité encore trop importante pour des attaques fréquentes et /ou des registres relativement longs.

Je vais maintenant établir une comparaison de l'attaque par décimation (DA), quand elle est possible (voir section 3.5) avec l'attaque CT. Supposons que par l'application de la proposition 23 une réduction de $\Delta L$ bits sur la recherche exhaustive soit possible ($\Delta L \geq \frac{L}{2}$). Alors, l'attaque par décimation a une complexité de

$$\mathcal{C}_{DA} = N \cdot \mathcal{O}(2^{L-\Delta L+1})$$

où $N$ est donnée par l'équation 3.2, et nécessite une quantité négligeable de mémoire. Calculons le gain par rapport à l'attaque CT. Dans [25] la complexité est donnée par la formule suivante :

$$\mathcal{C}_{CT} = 5.(\delta - 1).\frac{K_\delta}{C_{\delta-2}(p)}.2^{\alpha_\delta(p)+\frac{L}{\delta-1}}$$

où $\delta$ est le poids de l'équation de parité ($3 \leq \delta \leq 5$) et $C_{\delta-2}(p)$ est la capacité du BSC, de probabilité générale

$$p_{\delta-2} = \frac{1}{2}(1 - (1 - 2p)^{\delta-2})$$

*i.e.*

$$C_{\delta-2}(p) = 1 + p_{\delta-2}\log(p_{\delta-2}) + (1 - p_{\delta-2})\log(1 - p_{\delta-2}).$$

La valeur $K_\delta \approx 1$ si $\delta \geq 4$ et $K_3 \approx 2$. Enfin

$$\alpha_\delta(p) = \frac{1}{\delta - 1}\log_2[(\delta - 1)!\frac{K_\delta}{C_{\delta-2}(p)}].$$

Le gain de complexité de l'attaque par décimation est donc

$$\frac{\mathcal{C}_{CT}}{\mathcal{C}_{DA}} = 5 \cdot (\delta - 1) \cdot \frac{K_\delta}{N \cdot C_{\delta-2}(p)} \cdot 2^{\alpha_\delta(p) + \frac{L}{\delta-1} + \Delta L - L - 1}$$

Il est facile de voir que le gain augmente avec $\delta$. Or utiliser des poids $\delta$ plus grand que 3 précisément le point central de l'attaque CT. Cela renforce l'intérêt de l'attaque par décimation. De plus, le gain augmente avec la probabilité d'erreur $p$. Ce fait assure une efficacité plus grande qu'avec l'attaque CT. En ce qui concerne la mémoire requise, l'attaque CT a besoin

| | Attaque DA | Attaque CT ($\delta = 5$) |
|---|---|---|
| Bits de chiffré | $2^{33}$ | $2^{20}$ |
| Complexité | $2^{30}$ | $2^{59}$ |
| Mémoire (bytes) | $< 1024$ | $2^{38}$ |

TABLE 3.4 – Comparaison sur l'exemple 5 ($p = 0.49$)

de

$$(\delta - 1).\frac{K_\delta}{C_{\delta-2}(p)} + 2^{\alpha_\delta(p) + \frac{L}{\delta-1} + 1}$$

mots mémoire. Elle augmente, une fois encore avec $\delta$ et $p$. Pour illustrer ce

| | DA | CT ($\delta = 5$) (Décodage) | CT ($\delta = 5$) (ppm) | CT ($\delta = 5$) (ppwm) |
|---|---|---|---|---|
| Bits chiffré | $2^{30}$ | $2^{29}$ | - | - |
| Complexité | $2^{56}$ | $2^{67}$ | $2^{56}$ | $2^{81}$ |
| Mém. (bytes) | $< 1024$ | $2^{39}$ | $2^{57}$ | - |

TABLE 3.5 – Comparaison sur l'exemple 6 ($p = 0.49$)

gain, les tables 3.4 et 3.5 comparent les résultats de simulation précédents (voir section 3.4.2) avec ceux de l'attaque CT [25].

Malgré une suite nécessaire plus grande, le gain de complexité est conséquent ($2^{39}$) avec aucune contrainte mémoire. Toutefois pour être juste, l'attaque CT reste meilleure en terme de taille de chiffré pour des registres relativement courts et doit être préférée. Ce n'est plus le cas quand $L$ augmente comme l'illustre l'exemple 6 ("ppm" signifie précalcul avec mémoire et "ppwm" signifie précalcul sans mémoire).

Il faut noter que le précalcul dans l'attaque CT est fait une fois pour toutes, mais doit être refait chaque fois que le registre change. L'attaque par décimation ne requiert aucun précalcul.

# Chapitre 4

# Reconstruction de systèmes par flot

## 4.1 Introduction

Dans ce chapitre, je vais présenter la technique de reconstruction que j'ai développée dans le cas des systèmes de chiffrement à flot. Je ne considèrerai que le cas des générateurs à combinaison de registres, dans la mesure où (voir section 2.5) beaucoup de systèmes y sont équivalents et peuvent s'y ramener par transformation de schéma.

Le problème peut être formulé ainsi. Face à un adversaire qui utilise une machine de chiffrement à flot, non connue, comment exploiter les interceptions de ses transmissions et cryptanalyser son système. Notons que ce problème est très fréquemment rencontré. A moins de très grossières erreurs des chiffreurs, il n'est pas possible de mener une cryptanalyse sans la connaissance de la machine, c'est-à-dire de la logique de chiffrement.

Un précédent historique existe : celui de la machine PURPLE , durant la seconde guerre mondiale, utilisée par les japonais [102]. La cryptanalyse n'a été possible qu'après avoir reconstruit la machine, ou un équivalent. Les cryptanalystes américains n'ont jamais su d'ailleurs si la logique qu'ils ont reconstituée était celle réellement utilisée par le Japonais, ou simplement un équivalent. Les japonais, quant à eux, après la guerre, n'ont jamais voulu croire à cette reconstruction. Ils ont toujours pensé que l'algorithme lui-même avait été compromis (par l'espionnage humain).

Ce chapitre présente une approche similaire pour les systèmes à flots. A l'aide de résultats algébriques et statistiques, je montrerai comment retrouver toutes les primitives constituant ce genre de schéma : les registres (et

plus précisément les polynômes de rebouclage) et la fonction booléenne de combinaison.

La reconstruction des registres à décalage est en soi déjà intéressante. Une attaque de système à combinaison de registres [134] dûe à S. Palit et B. Roy, en 1999, a été montrée possible quand la fonction de combinaison est inconnue. Toutefois, elle demeure de complexité exponentielle car utilise l'attaque de Siegenthaler [154, 155].

Pour mener cette reconstruction, j'ai fixé la base de travail suivante :

- Je ne peux utiliser que du matériel chiffré. C'est en effet le seul matériel disponible en quantité relativement abondante et facile à obtenir (interceptions). La connaissance du clair correspondant n'est pas possible car trop irréaliste. Toutefois, on admet que les messages chiffrés obtenus aient pu être générés à partir de clefs de chiffrement différentes. La technique de reconstruction fonctionne en effet très bien avec cette hypothèse. Chaque message sera de longueur réaliste, c'est-à-dire de quelques milliers de caractères tout au plus. Toutes ces conditions ont pour but de permettre une reconstruction opérationnelle, ne dépendant pas de prérequis fantaisistes.
- Le temps de calcul pourra être long. Le travail de reconstruction est en effet effectué une seule fois (pour toutes). La durée de vie d'un algorithme de ce type est en général très longue : dix à vingt ans. La raison en est que la conception de tels systèmes (la plupart gouvernementaux ou industriels) est très longue et coûteuse (développement, tests, implémentation en hard,...).
- Je connais le code de représentation du clair (ASCII, CCITTx, Murray,...) ou à défaut, au minimum, j'ai une connaissance statistique minimale (probabilité $p_0$ pour un bit de clair d'être nul), d'ailleurs pratiquement commune à tous ces codes ($0.6 \leq p_0 \leq 0.7$). Je connais également au moins le groupe linguistique auquel appartient la langue utilisée par le chiffreur. La phase d'interception est toujours en mesure de fournir cette connaissance.
- Le système est de type générateur à combinaison de registres. La plupart des systèmes réels utilisent des fonctions de combinaisons de 9 variables au plus (donc de registres).

Ce chapitre est organisé de la manière suivante : dans une première section, je traiterai de la reconstruction des registres et présenterai l'algorithme la permettant (algorithme 4.1). Ce problème consiste à retrouver quels polynômes de rebouclage sont utilisés dans le système de chiffrement étudié. Ces polynômes seront détectables grâce à certains de leurs multiples présentant la propriété d'avoir peu de coefficients non nuls (proposition 26 et son corol-

laire, théorème 8 et son corollaire).

Je présenterai ensuite une analyse de complexité. Elle permettra de prouver que le gain en complexité est très important par rapport à une recherche exhaustive de tous les polynômes possibles d'un degré maximal donné.

Je donnerai ensuite quelques résultats numériques sur un cas totalement traité. Enfin je présenterai comment reconstruire la fonction booléenne de combinaison utilisée. Il s'agit grâce aux informations obtenues lors de la reconstruction des polynômes, de déterminer les coefficients de Walsh de la fonction. La proposition 14, démontrée au chapitre 2, permet alors de retouver les coefficients de la forme algébrique normale de la fonction.

Ces résultats ont été publiés avec Anne Canteaut dans [23, 24].

## 4.2 Reconstruction des registres

Je vais maintenant montrer comment il est possible de retrouver totalement les primitives constituant le générateur pseudo-aléatoire décrit par la figure 4.1, uniquement avec des bits de chiffré. La première étape de la reconstruction passe par la reconstitution des registres (les polynômes de rebouclage) constituant le système utilisé.

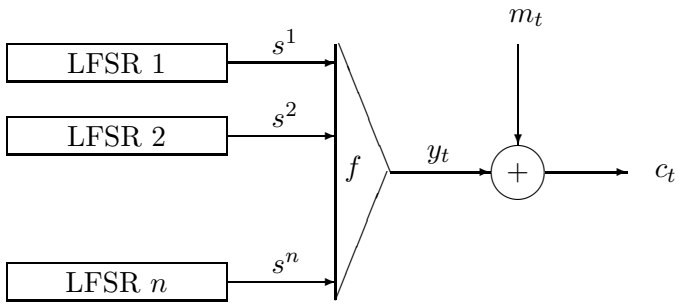


FIGURE 4.1 – Système générique de chiffrement par flot : générateur à combinaison de registres

### 4.2.1 Notations

J'utiliserai dorénavant les notations suivantes (voir figure 4.1 pour certaines d'entre elles) :

– $n$ désignera le nombre de registres du générateur.

- $P_i$ et $L_i$ désignent respectivement le polynôme de rebouclage et le degré de ce polynôme (qui désigne également la longueur du registre, voir définition 11 du chapitre 2) du ième registre. Dans tout ce qui suit degré du polynôme et longueur du registre seront interchangeables.
- $s^i$ représentera la séquence générée par le ième registre. Les séquences $y$, $m$ et $c$ respectivement désigneront la suite pseudo-aléatoire (avant combinaison avec le clair), la suite de texte clair et la suite de texte chiffré. $s$ désignera une séquence binaire quelconque.
- Pour désigner l'opération bit à bit, l'index de temps $t$ sera utilisé pour préciser le bit considéré.
- $1_T$ désigne le $n$-uplet défini par la fonction indicatrice relative à l'ensemble $T$, support (ensemble des indices non nuls) de ce $n$-uplet. La ième composante de $1_T$ vaut 1 si et seulement si $i \in T$.
- $\mathcal{S}(P)$ désigne toutes les séquences produites par un registre ayant $P$ pour polynôme de rebouclage.

J'utiliserai la notion de *densité* d'un polynôme

**Définition 25** *On appelle densité d'un polynôme $P$, notée lac($P$), le nombre de ses coefficients non nuls. $P$ s'écrit alors*

$$P(X) = 1 + \sum_{j=1}^{lac(P)-1} X^{i_j} \text{ avec } i_1 < i_2 < \ldots < i_{lac(P)-1} .$$

*On parle également du poids du polynôme $P$.*

Le texte clair est supposé produit par une source binaire sans mémoire telle que $Pr[m_t = 0] = p_0 \neq \frac{1}{2}$ . Dans la pratique, cette hypothèse constitue une excellente approximation et tous les codes en usage dans le monde (ASCII, Murray, CCITTx ...) la satisfont très largement. La valeur $p_0$ sera prise entre 0.6 et 0.7, ce qui correspond à la réalité.

### 4.2.2 Présentation de la méthode

La méthode utilisée pour retrouver les polynômes de rebouclage utilisés dans le système de chiffrement considéré est la suivante : tous les polynômes $Q$ de degré au plus $D$ et de densité fixée $\text{lac}(Q) = d$ vont être testés par l'algorithme. Sachant que si un polynôme $P$ de degré $L$ donné par

$$P = X^L \oplus c_1 X^{L-1} \oplus c_2 X^{L-2} \oplus \cdots \oplus c_L X \oplus 1$$

a été utilisé, toute suite $s = (s_t)_{t \geq 0}$ de $\mathcal{S}(P)$ satisfait la relation de récurrence

$$s_{i+L} \oplus c_1 s_{i+L-1} \oplus c_2 s_{i+L-2} \oplus \cdots \oplus c_L s_i = 0$$

Tout multiple $Q$ de $P$ vérifie alors une relation de récurrence similaire.

Autrement dit, tout polynôme $Q$ detecté, parce qu'il satisfait (au moins statistiquement, puisque l'on travaillera sur une version bruitée de la séquence $s$) une certaine relation de récurrence sur cette suite, sera considéré comme un multiple du polynôme $P$ cherché, que l'on obtiendra par une simple factorisation.

Comme on travaille sur une suite bruitée (le bruit provient de la fonction de combinaison et de l'addition du texte clair), il faudra exploiter les corrélations éventuelles entre la sortie de chaque registre (ou groupe de registres) et la suite chiffrée. Plus la densité du polynôme de rebouclage du registre considéré est élevée plus cette corrélation sera difficile à exploiter. Cela explique pourquoi il est plus intéressant de travailler sur des multiples de faible densité.

Je vais maintenant formaliser les choses et présenter rigoureusement le modèle statistique utilisé.

### 4.2.3 Modèle statistique

Il faut d'abord, dans le cas général, montrer que la connaissance d'une séquence $s$ corrélée avec la séquence de texte chiffré (voir figure 3.1) fournit des informations sur le polynôme de rebouclage des registres utilisés. On connait la séquence $c = (c_t)_{t\geq 0}$. Elle correspond au texte chiffré.

**Proposition 26** *Soit $s = (s_t)_{t\geq 0}$ une séquence binaire quelconque et $s^1, s^2, \ldots, s^n$ les sorties respectives de $n$ registres combinés par une fonction $f$.*

*Si $Pr[c_t = s_t] \neq 1/2$ alors il existe une fonction booléenne $g$ à $n$ variables telle que $s = g(s^1, \ldots, s^n)$. De plus*

$$Pr[c_t = s_t] = 1 - p_0 - p_g + 2p_0p_g$$

*où $p_g = Pr[f(x_1, \ldots, x_n) = g(x_1, \ldots, x_n)]$.*

*Preuve.*
Sachant que $m_t, y_t$ désignent respectivement les bit de message clair et de suite chiffrante au temps $t$ et que $f$ désigne la fonction de combinaison (voir figures 3.1 et 4.1), de façon évidente, nous avons :

$$\begin{aligned} Pr[c_t = s_t] &= Pr[y_t = s_t]Pr[m_t = 0] + Pr[y_t = s_t \oplus 1]Pr[m_t = 1] \\ &= 1 - p_0 - Pr[y_t = s_t] + 2p_0Pr[y_t = s_t] \ . \end{aligned}$$

Par hypothèse, $p_0 \neq 1/2$. Alors $Pr[c_t = s_t] \neq 1/2$ implique que $Pr[y_t = s_t] \neq 1/2$. Comme $y = f(s^1, \ldots, s^n)$, les séquences $y$ et $s$ sont statistiquement indépendantes si $s$ est statistiquement indépendante de $(s^1, \ldots, s^n)$.

Il s'ensuit que $Pr[y_t = s_t] = 1/2$ à moins que $s = g(s^1, \ldots, s^n)$ pour une fonction booléenne $g$. Dans ce cas, nous avons

$$Pr[y_t = c_t] = Pr[f(x_1, \ldots, x_n) = g(x_1, \ldots, x_n)] \ .$$

On peut construire la fonction $g$ de la manière suivante.

$$\forall\, x \in \mathbb{F}_2^n \quad \text{soit} \quad T_x = \{t | x_t = x\}$$

Autrement dit pour $x$ fixé, $T_x$ est l'ensemble des temps $t$ pour lesquels la valeur $x$ est en entrée de $f$. On a de façon évidente

$$T_x = T_x^1 \cup T_x^0 \quad \text{et} \quad T_x^1 \cap T_x^0 = \emptyset$$

où

$$T_x^0 = \{t \in T_x | s_t = y_t\} \quad \text{et} \quad T_x^1 = \{t \in T_x | s_t \neq y_t\}.$$

Si les suites $s$ et $y$ sont statistiquement dépendante, on a

$$|T_x^0| \neq \frac{|T_x|}{2} \quad \text{et } |T_x^1| \neq \frac{|T_x|}{2}.$$

Alors si $|T_x^0| > |T_x^1|$ on choisit $g(x) = f(x)$ sinon $g(x) = 1 \oplus f(x)$. Il est alors facile de voir que $g$ satisfait à la proposition par construction. ■

Il faut remarquer que certaines variables peuvent très bien ne pas apparaitre dans la forme algébrique normale de $g$ (dégénérescence). Si $s$ est telle que $P[c_t = s_t] \neq 1/2$, nous déduisons de la proposition précédente et de la proposition 10 que le polynôme de rebouclage de $s$ est lié aux polynômes $P_1$,...,$P_n$.

Le cas le plus intéressant, et celui qui va permettre la reconstruction, est celui où la fonction $f$ est $\tau$-corrélée, pour une certaine valeur $\tau$, c'est-à-dire lorsque

$$P[f(s^{i_1}, s^{i_2}, \ldots, s^{i_\tau}) = s^{i_1} \oplus s^{i_2} \oplus \ldots \oplus s^{i_\tau}] \neq \frac{1}{2}.$$

La fonction $g$ est alors de façon évidente donnée par $g(x) = u.x$ (où . désigne le produit scalaire sur $\mathbb{F}_2$) où $supp(u) = \{i_1, i_2, \ldots, i_\tau\}$. Dans ce cas là, la corrélation entre la suite chiffrée $c = (c_t)_{t \geq 0}$ et la séquence $s = s^{i_1} \oplus s^{i_2} \oplus \ldots \oplus s^{i_\tau}$ va permettre d'obtenir une information sur $P_{i_1}, P_{i_2}, \ldots, P_{i_\tau}$.

**Corollaire 2** *Soit $Q \in \mathbb{F}_2[X]$ et soit $\mathcal{S}(Q)$ tel que défini dans la proposition 10. On se donne une suite $c = (c_t)_{t \geq 0}$. S'il existe $s \in \mathcal{S}(Q)$ tel que $P[c_t = s_t] \neq 1/2$, alors*

$$\exists g \quad \textit{une fonction}\ , \exists Q' \in \mathbb{F}_2[X] \quad Q'|Q \ \textit{et}\ Q' \ \textit{dépend de}\ P_1, \ldots, P_n \ \textit{et de}\ g.$$

Ce résultat me permet d'utiliser la méthode présentée précédemment, pour retrouver des informations concernant $P_1, \ldots, P_n$. Soit $\mathcal{Q}$ un sous-ensemble de $\mathbb{F}_2[X]$. Pour tout $Q \in \mathcal{Q}$, il suffit de déterminer si $\mathcal{S}(Q)$ contient une séquence corrélée au texte chiffré représentant la séquence $c = (c_t)_{t\geq 0}$. Si une telle séquence existe, $Q$ fournit des informations sur $P_1, \ldots, P_n$.

Je choisis ici pour $\mathcal{Q}$, l'ensemble de tous les polynômes $Q$ de $\mathbb{F}_2[X]$ de densité $d$ et de degré au plus $D$ ayant la forme suivante

$$Q(X) = 1 + \sum_{j=1}^{d-1} X^{i_j} \qquad i_j \in \mathbb{N} \ .$$

Les paramètres $d$ et $D$ comme je le montrerai plus loin détermineront la quantité de texte chiffré nécessaire. Selon leurs valeurs, comme je le montrerai lors de l'analyse de complexité, la complexité de la méthode sera plus ou moins importante.

Le corollaire 2 se place dans le cas général d'une fonction $g$ sans autre précision sur cette fonction. Il faut s'assurer que sa forme sera suffisamment simple pour permettre effectivement d'extraire pratiquement l'information sur les polynômes $P_1, \ldots, P_n$ et si possible les polynômes eux-mêmes. Cela impose d'ores et déja la condition initiale suivante sur le paramètre $D$, en utilisant l'équation 2.2 :

- Il est d'abord important de rappeler la propriété suivante (proposition 10) : le degré du polynôme de rebouclage d'une séquence correspondant au produit de deux séquences $s^i$ et $s^j$, de polynôme de rebouclage respectif $P_i$ et $P_j$ est borné par le produit $L_iL_j$ des degrés respectifs de ces polynômes. Il est bien plus élevé que le degré du polynôme de rebouclage qui aurait généré la suite $s^i \oplus s^j$. Par conséquent (équation 2.2), le paramètre $D$ est borné comme suit

$$D \leq \left( \sqrt{m(d)(d-1)!2^{L_iL_j}} \right)^{\frac{1}{d-1}} .$$

- Si la limite supérieure $D$ sur le degré des polynômes testés est convenablement choisie, les polynômes $Q$ détectés par l'algorithme correspondent au cas où la fonction de combinaison $g$ est linéaire. Il suffit de considérer la définition de la complexité linéaire d'un registre (section 2.3.2) et l'équation 2.2 donnant le nombre moyen de multiples en fonction de $D$. Pour $g(x) = u \cdot x$, tout polynôme de rebouclage générant $s = g(s^1, \ldots, s^n)$ est un multiple de $\mathrm{lcm}_{i \in supp(u)} P_i$ où $supp(u) = \{i, u_i = 1\}$. Comme tous les polynômes de rebouclage

utilisés dans la pratique sont primitifs, nous avons $\mathrm{lcm}_{i\in supp(u)}P_i = \prod_{i\in supp(u)} P_i$ dans la plupart des situations.
De plus nous avons

$$Pr[c_t = s_t] = \frac{1}{2} + \frac{(2p_0 - 1)}{2^{n+1}}\widehat{\chi}_f(u) \ . \tag{4.1}$$

Autrement dit, par un choix adapté du paramètre $D$, la fonction $g$ sera linéaire. L'opération pour retrouver alors les polynômes $P_1,\ldots,P_n$ sera une simple factorisation. Toujours avec l'équation 2.2 on a

$$D \geq \left(\sqrt{m(d)(d-1)!2^{L_i+L_j}}\right)^{\frac{1}{d-1}} .$$

L'exemple suivant illustre la méthode présentée.

**Exemple 7** *Considérons le générateur à combinaison de registres décrit par Geffe [71]. Ce générateur est constitué de trois registres combinés par la fonction booléenne $f$ :*

$$f(x_1, x_2, x_3) = x_1x_2 \oplus x_2x_3 \oplus x_1 \ .$$

*Supposons que les polynômes de rebouclage des registres constituant le générateur soient choisis aléatoirement primitifs et que leur degré soit respectivement $L_1 = 15$, $L_2 = 17$ et $L_3 = 23$. Soit $c$ la suite de texte chiffré obtenue par addition bit à bit avec la séquence produite par le générateur de Geffe à une suite de texte clair telle que $p_0 \neq 0.5$.*

*Soit $\mathcal{Q}$ l'ensemble de tous les polynômes de poids 4 et de degré au plus égal à 10000. Pour tout $Q \in \mathcal{Q}$, on détermine si $\mathcal{S}(Q)$ contient une séquence corrélée avec $c$. Soit $P$ un polynôme aléatoire de degré $L$. On déduit de la formule (2.2) que $\mathcal{Q}$ contient un multiple de $P$ de poids 4 si $L \leq 37$. La méthode présentée permettra de détecter les multiples de $P_1$, $P_3$ et $P_1P_2$. Notons que $P_2$ ne peut être détecté par l'algorithme puisque le coefficient de Walsh $\widehat{\chi}_f(010)$ s'annule.*

D'un point de vue pratique, pour déterminer si $\mathcal{S}(Q)$ contient une séquence corrélée avec $c$, il suffit de calculer l'équation de parité correspondant à $Q$ pour les bits de texte chiffré. L'efficacité de cette procédure dépend fortement de la densité de $Q$ comme le précise le théorème suivant.

**Théorème 8** *Soit $Q$ un polynôme dans $\mathbb{F}_2[X]$ de densité $d$ de la forme suivante*

$$Q(X) = 1 + \sum_{j=1}^{d-1} X^{i_j} \ \textit{avec}\ i_1 < i_2 < \ldots < i_{d-1} \ .$$

*Pour une suite de texte chiffré donnée $(c_t)_{t<N}$, on considère la séquence binaire $(z_t)_{i_{d-1}\leq t<N}$ définie par $z_t = c_t \oplus \bigoplus_{j=1}^{d-1} c_{t-i_j}$ . Alors la variable aléatoire $Z = \sum_{t=i_{d-1}}^{N-1} (-1)^{z_t}$ suit une loi normale de moyenne*

$$M = \pm(N - i_{d-1})(2\varepsilon)^d$$

*et de variance*

$$\sigma^2 = (N - i_{d-1})(1 - (2\varepsilon)^{2d})$$

*où $\varepsilon = \max_{s\in\mathcal{S}(Q)} |Pr[c_t = s_t] - \frac{1}{2}|$.*

*Preuve.*
Soit $s \in \mathcal{S}(Q)$ telle que $|Pr[c_t = s_t] - \frac{1}{2}|$ soit maximale. Soit $p = Pr[c_t = s_t]$. Quel que soit $t$, $c_t$ se décompose en $c_t = s_t \oplus e_t$ où $Pr[e_t = 1] = 1 - p$. Alors nous avons

$$Pr[z_t = 1] = Pr[c_t \oplus \bigoplus_{j=1}^{d-1} c_{t-i_j} = 1] = Pr[e_t \oplus \bigoplus_{j=1}^{d-1} e_{t-i_j} = 1]$$

comme $s$ satisfait l'équation de parité car $s \in \mathcal{S}(Q)$. Ceci implique que $z_t = 1$ si et seulement si le nombre de positions $i \in \{t, t - i_1, \ldots, t - i_{d-1}\}$ telles que $e_i = 1$ est impair. Pour cela nous avons

$$\begin{aligned} Pr[z_t = 1] &= \sum_{\ell=0,\ell \text{ impair}}^{d} \binom{d}{\ell} (1-p)^\ell p^{d-\ell} \\ &= \frac{1}{2}\left[\sum_{\ell=0}^{d} \binom{d}{\ell}(1-p)^\ell p^{d-\ell} - \sum_{\ell=0}^{d} \binom{d}{\ell}(p-1)^\ell p^{d-\ell}\right] \\ &= \frac{1}{2}\left[1 - (2p-1)^d\right] \ . \end{aligned}$$

La variable aléatoire $Z$ peut maintenant être explicitée par

$$Z = (N - i_{d-1}) - 2\sum_{t=i_{d-1}}^{N} z_t \ .$$

Toutes les variables aléatoires $z_t$ sont indépendantes et identiquement distribuées. Par application du théorème central limite [54], la variable aléatoire

$$\sum_{t=i_{d-1}}^{N} z_t$$

pour de grandes valeurs de $(N - i_{d-1})$ peut être supposée suivre une distribution normale de moyenne $(N - i_{d-1})Pr[z_t = 1]$ et de variance $(N - i_{d-1})Pr[z_t = 1]Pr[z_t = 0]$. Il s'ensuit que $Z$ suit une loi normale de moyenne

$$M = (N - i_{d-1})(1 - 2Pr[z_t = 1]) = (N - i_{d-1})(2p - 1)^d$$

et de variance

$$\begin{aligned} \sigma^2 &= 4(N - i_{d-1})Pr[z_t = 1]Pr[z_t = 0] \\ &= (N - i_{d-1})(1 - (2p - 1)^d)(1 + (2p - 1)^d) \\ &= (N - i_{d-1})(1 - (2p - 1)^{2d}) \ . \end{aligned}$$

■

Si toutes les séquences de $\mathcal{S}(Q)$ sont statistiquement indépendantes de $c$, $Z$ suit une loi normale de moyenne 0 et de variance $(N - i_{d-1})$ comme $\varepsilon = 0$ dans ce cas.

Il faut maintenant discriminer deux hypothèses :

– $\mathcal{H}_0$ : pour tout $s \in \mathcal{S}(Q)$, $Pr[c_t = s_t] = \frac{1}{2}$.

– $\mathcal{H}_1$ : il existe $s \in \mathcal{S}(Q)$ telle que $Pr[s_t = c_t] \neq \frac{1}{2}$.

Du théorème précédent, nous avons :

$$\begin{aligned} Pr[Z = x \mid \mathcal{H}_0] &= \frac{1}{\sqrt{2\pi(N - i_{d-1})}} \exp\left(-\frac{x^2}{2(N - i_{d-1})}\right) \\ Pr[Z = x \mid \mathcal{H}_1] &= \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - M)^2}{2\sigma^2}\right) \end{aligned}$$

On utilise un seuil de décision $T$, $T > 0$, pour discriminer les hypothèses $\mathcal{H}_0$ et $\mathcal{H}_1$. Si $|Z| < T$, $\mathcal{H}_0$ est conservée ; si $|Z| \geq T$, $\mathcal{H}_1$ est acceptée. Le nombre minimum de bits de texte chiffré nécessaires, $N$, dépend du nombre de décisions fausses (fausses alarmes ou encore candidats indûment conservés) qui sont autorisées. Ce nombre correspond à la probabilité de fausse alarme, $P_f = Pr[|Z| \geq T \mid \mathcal{H}_0]$. Le seuil de décision est déterminé par la probabilité de non détection, $P_n = Pr[|Z| < T \mid \mathcal{H}_1]$. Soit $\Phi^*$ la fonction de répartition de la loi normale centrée réduite,

$$\Phi^*(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{x^2}{2}\right) dx \ .$$

Alors nous avons

$$\begin{aligned} P_f &= Pr[|Z| \geq T \mid \mathcal{H}_0] = 2\int_{-\infty}^{-T} Pr[Z = x \mid \mathcal{H}_0]dx \\ &= \frac{2}{\sqrt{2\pi(N - i_{d-1})}} \int_{-\infty}^{-T} \exp\left(-\frac{x^2}{2(N - i_{d-1})}\right) dx \\ &= 2\Phi^*\left(\frac{-T}{\sqrt{N - i_{d-1}}}\right) \end{aligned}$$

De la même manière, la probabilité de non détection est donnée par

$$\begin{aligned} P_n &= Pr[|Z| < T \mid \mathcal{H}_1] = \frac{1}{\sqrt{2\pi}\sigma} \int_{-T}^{T} \exp\left(-\frac{(x - M)^2}{2\sigma^2}\right) dx \\ &= \frac{1}{\sqrt{2\pi}} \int_{\frac{-T-M}{\sigma}}^{\frac{T-M}{\sigma}} \exp\left(-\frac{x^2}{2}\right) dx = \Phi^*\left(\frac{T - M}{\sigma}\right) - \Phi^*\left(\frac{-T - M}{\sigma}\right) \\ &= \Phi^*\left(\frac{T - |M|}{\sigma}\right) - \Phi^*\left(\frac{-T - |M|}{\sigma}\right) \end{aligned}$$

comme $M$ n'est pas nécessairement positif.

Dans la plupart des cas, $\Phi^*(\frac{-T-|M|}{\sigma})$ est beaucoup plus petit que $P_n$ et que $\Phi^*(\frac{T-|M|}{\sigma})$. Ce dernier approximera donc $P_n$. La valeur de $P_n$, déterminée à l'avance, fixe la valeur du seuil :

$$\begin{aligned} T &= |M| + \Phi^{*-1}(P_n)\sigma \\ &= (N - i_{d-1})(2\varepsilon)^d + \Phi^{*-1}(P_n)\sqrt{(N - i_{d-1})(1 - (2\varepsilon)^{2d})}\ . \end{aligned}$$

De la même manière, la valeur fixée à l'avance pour la probabilité de fausse alarme fournit la valeur minimale de $(N - i_{d-1})$ :

$$N - i_{d-1} = \left(\frac{T}{\Phi^{-1}(1 - \frac{P_f}{2})}\right)^2\ .$$

Après différents essais pour déterminer les valeurs de $P_f$ and $P_n$ les plus appropriées et les plus efficaces, j'ai choisi de prendre $P_f = 2^{-20}$ et $P_n = 10^{-3}$.

Dans la pratique, la séquence de texte chiffré dont on dispose n'est pas d'une longueur excessive. Le cryptanalyste ne dispose que de messages, longs de quelques milliers de bits consécutifs, tout au plus. Ces messages peuvent

raisonnablement être supposés avoir été produits à partir de clefs différentes, c'est-à-dire avec des initialisations de registres différents.

Or cette technique de reconstruction, à l'inverse de la cryptanalyse, ne se préoccupe pas de retrouver l'initialisation des registres. L'information qui est recherchée est liée aux polynômes (on considère des polynômes et leurs multiples décrivant des relations de récurrence linéaires ) et non à telle ou telle initialisation (les relations de récurrence sont valables quelles que soient les suites produites par un registre de polynôme de rebouclage donné, c'est-à-dire quelles que soient les initialisations). Cela permet de s'affranchir de cette apparente limitation.

Le théorème 8 peut s'adapter sans problème à cette situation plus réaliste. Les différentes expérimentations que j'ai pu mener, ont totalement confirmé sa validité.

**Corollaire 3** *Soit $Q$ un polynôme dans $\mathbb{F}_2[X]$ de densité $d$ de la forme suivante*

$$Q(X) = 1 + \sum_{j=1}^{d-1} X^{i_j} \text{ avec } i_1 < i_2 < \ldots < i_{d-1} \ .$$

*Pour $n_c$ textes chiffrés $c^k$, $1 \leq k \leq n_c$, de longueurs respectives $LC(k)$, on considère les séquences binaires $(z_t^k)_{i_{d-1} \leq t < LC(k), 1 \leq k \leq n_c}$ définies par*

$$z_t^k = c_t^k \oplus \bigoplus_{j=1}^{d-1} c_{t-i_j}^k \ .$$

*Alors la variable aléatoire*

$$Z = \sum_{k=1}^{n_c} \sum_{t=i_{d-1}}^{LC(k)-1} (-1)^{z_t^k}$$

*suit une loi normale de moyenne*

$$M = \pm(2\varepsilon)^d \sum_{k=1}^{n_c} (LC(k) - i_{d-1})$$

*et de variance*

$$\sigma^2 = (1 - (2\varepsilon)^{2d}) \sum_{k=1}^{n_c} (LC(k) - i_{d-1})$$

*où* $\varepsilon = \max_{s \in \mathcal{S}(Q)} |Pr[c_t = s_t] - \frac{1}{2}|$.

Pour tout $(d-1)$-uplets $(i_1, \ldots, i_{d-1})$ tel que
$0 < i_1 < \ldots < i_{d-1} < D$

1. $N \leftarrow \sum_{k=1}^{n_c} (LC(k) - i_{d-1})$.
2. $T \leftarrow N(2\varepsilon_{\min})^d - 3\sqrt{N(1-(2\varepsilon_{\min})^{2d})}$.
3. $Z \leftarrow 0$.
4. Pour chaque message $(c_t^k)_{0 \leq t < LC(k)}$ où $LC(k) > i_{d-1}$
   – Pour tout $t$ de $i_{d-1}$ à $LC(k) - 1$
   (a) $z \leftarrow c_t^k \oplus \bigoplus_{j=1}^{d-1} c_{t-i_j}^k$.
   (b) $Z \leftarrow Z + (-1)^z$.
5. Si $|Z| \geq T$, factoriser $1 + \sum_{j=1}^{d-1} X^{i_j}$ et mémoriser tous ses facteurs.
6. Mémoriser $Z$.

TABLE 4.1 – Algorithme de reconstruction des registres.

En final, l'algorithme suivant réalise pratiquement la méthode proposée, pour la reconstruction des polynômes de rebouclage.

Les facteurs détectés plusieurs fois sont avec une grande probabilité, les polynômes de rebouclage des registres constituant le système étudié. En fait, la présence de facteurs communs à plusieurs multiples de petits poids permet de jouer sur la probabilité de fausse alarme et ainsi diminuer la taille de chiffré nécessaire. Des expériences ont montré que de façon efficace, la détection de facteurs communs ramène finalement cette probabilité vers la valeur nulle.

### 4.2.4 Analyse de complexité

Je vais maintenant discuter le choix des paramètres d'entrée $d$, $D$ et $\varepsilon_{\min}$. En particulier, nous allons voir que le choix d'explorer les multiples de petits poids permet de gagner en complexité par rapport à la recherche exhaustive sur tous les polynômes de rebouclage.

Rappelons que le but est de retrouver un maximum de multiples des polynômes

$$\prod_{i \in T} P_i, \quad T \subset \{1, \ldots, n\}$$

tels que

$$|P[c_t = \bigoplus_{i \in T} s_t^i] - 1/2| \geq \varepsilon_{\min}.$$

D'après la formule (4.1), ces sous-ensembles $T$ sont caractérisés par

$$\frac{|2p_0 - 1|}{2^{n+1}} |\widehat{\chi}_f(1_T)| \geq \varepsilon_{\min}$$

où la $i$ème composante du $n$-uplet $1_T$ vaut 1 si et seulement si $i \in T$. Il est connu que tous les coefficients de Walsh d'une fonction booléenne $f$ en $n$ variables sont divisibles par 4, sauf si $f$ est de degré $n$. Mais on suppose ici que le degré de $f$ est inférieur à $n$, ce qui est pratiquement toujours vrai. En effet, une fonction de degré $n$ ne peut être équilibrée. En choisissant

$$\varepsilon_{\min} = \frac{|2p_0 - 1|}{2^{n-1}} \tag{4.2}$$

nous sommes assurés de détecter tous les polynômes $\prod_{i \in T} P_i$ tels que

$$\widehat{\chi}_f(1_T) \neq 0.$$

Dans la plupart des cas pratiques, le nombre de variables ne dépasse guère 9.

Je suppose maintenant que la recherche peut être restreinte à tous les produits $\prod_{i \in T} P_i$ de degré au plus $L_{\max}$. Ceci signifie que je suppose que tous les polynômes de rebouclage $P_1, \ldots, P_n$ peuvent être retrouvés à partir de tous les produits $\prod_{i \in T} P_i$ tels que

$$\widehat{\chi}_f(1_T) \neq 0 \text{ et } \sum_{i \in T} L_i \leq L_{\max}.$$

Notons que $L_{\max}$ doit de façon évidente, être supérieur à la longueur maximum de tous les registres présents dans le système. Un polynôme de degré $L_{\max}$ est donc retrouvé par l'algorithme si il divise au moins un polynôme de poids $d$ et de degré au plus $D$. On déduit alors de la formule (2.2) que la valeur minimale possible pour $D$ est approximativement

$$D = (d-1)! \frac{1}{d-1} 2^{\frac{L_{\max}}{d-1}} \ . \tag{4.3}$$

Cela implique également que la reconstruction peut n'utiliser que des messages de longueur au moins égale à $LC$ avec

$$LC \geq (d-1)! \frac{1}{d-1} 2^{\frac{L_{\max}}{d-1}} \ . \tag{4.4}$$

De plus, j'ai pris comme hypothèse que la probabilité de fausse alarme soit inférieure à $2^{-20}$. Cela implique que

$$(\sum_{k=1}^{n_c} LC(k)) - n_c D \geq \left(\frac{T}{5}\right)^2 .$$

En replaçant $T$ par sa valeur, j'obtiens la condition suivante :

$$N_t - n_c D \geq \frac{1}{25}\left((N_t - n_c D)(2\varepsilon_{\min})^d - 3\sqrt{(N_t - n_c D)(1-(2\varepsilon_{\min})^{2d})}\right)^2$$

où $N_t = \sum_{k=1}^{n_c} LC(k)$ représente la longueur totale (tous messages cumulés) de texte chiffré utilisée. On en déduit :

$$N_t - n_c D \geq \frac{\left(5 + 3\sqrt{1-(2\varepsilon_{\min})^{2d}}\right)^2}{(2\varepsilon_{\min})^{2d}} . \tag{4.5}$$

Finalement la longueur totale de texte chiffré doit satisfaire :

$$N_t \geq n_c(d-1)!^{\frac{1}{d-1}} 2^{\frac{L_{\max}}{d-1}} + \frac{\left(5 + 3\sqrt{1-(2\varepsilon_{\min})^{2d}}\right)^2}{(2\varepsilon_{\min})^{2d}} . \tag{4.6}$$

Cette valeur est minimale si $n_c = 1$, c'est-à-dire si tous les bits de chiffré sont consécutifs et composent un seul message. Dans ce cas, la longueur minimale de texte chiffré nécessaire à la reconstruction est

$$N_t = \min_d \left\lceil (d-1)!^{\frac{1}{d-1}} 2^{\frac{L_{\max}}{d-1}} + \frac{\left(5 + 3\sqrt{1-(2\varepsilon_{\min})^{2d}}\right)^2}{(2\varepsilon_{\min})^{2d}} \right\rceil . \tag{4.7}$$

Les abaques 4.2, 4.3 et 4.4 montrent, pour différentes valeurs de $L_{\max}$, comment varient $N_t$ la valeur optimale de $d$ avec $\varepsilon_{\min}$. En situation réelle, les messages chiffrés ont *grosso modo* tous la même longueur $LC$. Le nombre nécessaire $n_c$ de tels messages, pour la reconstruction est donc

$$n_c \geq \frac{\left(5 + 3\sqrt{1-(2\varepsilon_{\min})^{2d}}\right)^2}{(2\varepsilon_{\min})^{2d}(LC - (d-1)^{\frac{1}{d-1}} 2^{\frac{L_{\max}}{d-1}})} .$$

L'algorithme est alors utilisé avec les valeurs de $d$ minimisant cette valeur $n_c$.

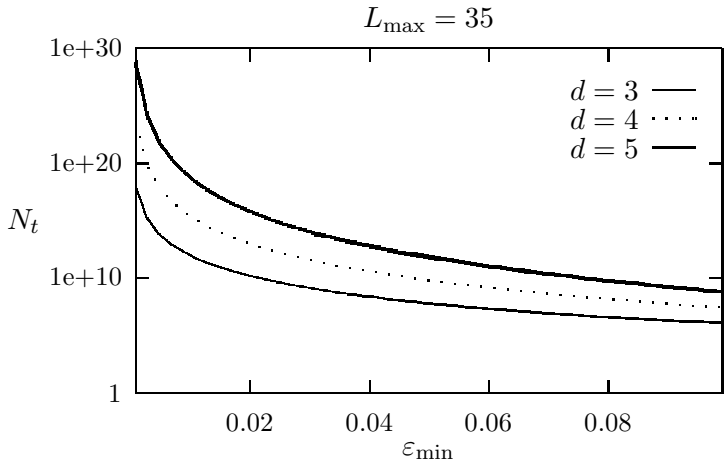


FIGURE 4.2 – Longueur minimum de chiffré nécessaire pour $L_{max} = 35$

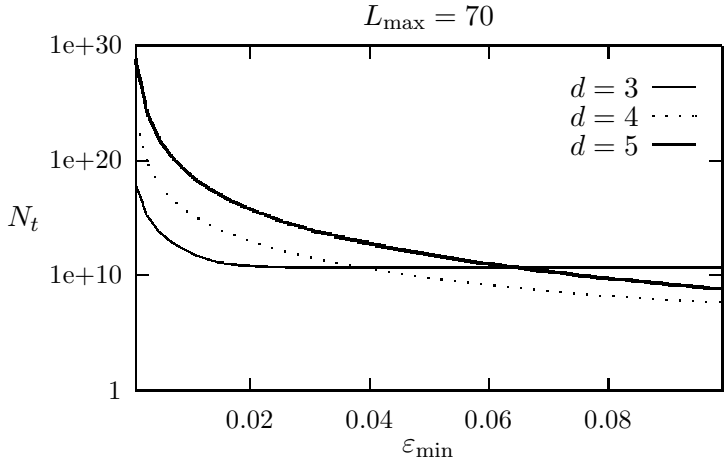


FIGURE 4.3 – Longueur minimum de chiffré nécessaire pour $L_{max} = 70$

**Exemple 8** *Supposons que* $\varepsilon_{\min} = 0.1$ *et* $L_{\max} = 35$*. On utilisera des messages de longueur commune en moyenne égale à* $LC = 10000$ *bits (cela représente environ 1 200 caracteères ASCII).*

*La condition (4.4) impose que* $d \geq 4$*. Le nombre nécessaire de messages chiffrés est* $n_c = 6109$ *pour* $d = 4$ *et* $n_c = 69083$ *pour* $d = 5$*. L'algorithme examinera tous les polynômes de poids 4 et de degré au plus* $D = 5910$*.*

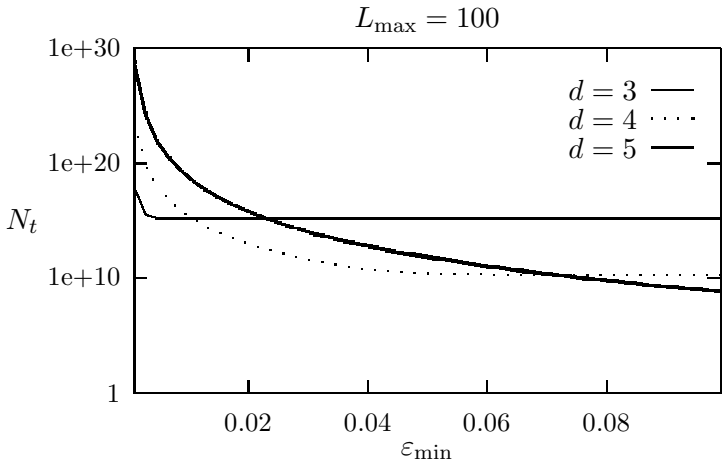


FIGURE 4.4 – Longueur minimum de chiffré nécessaire pour $L_{max} = 100$

Le nombre d'opérations effectuées par l'algorithme (sans compter l'étape de factorisation) est

$$\frac{D^{d-1}}{(d-1)!} d(N_t - n_c D) \ .$$

En utilisant les équations (4.3) et (4.6), on obtient la complexité suivante pour l'algorithme :

$$\frac{d 2^{L_{\max}} \left(5 + 3\sqrt{1 - (2\varepsilon_{\min})^{2d}}\right)^2}{(2\varepsilon_{\min})^{2d}} \ .$$

Une autre méthode (naïve celle-là) pour retrouver les polynômes de recouclage des registres du système considéré, consiste à examiner tous les polynômes de degré au plus $L_{\max}$ et à évaluer l'équation de parité correspondante sur la séquence de texte chiffré. Une analyse de complexité similaire peut alors être faite concernant cette approche.

Il faut choisir alors $D = L_{\max}$ et $d \simeq L_{\max}/2$ puisque le poids moyen d'un polynôme de degré $L_{\max}$ vaut $L_{\max}/2$. Avec ces paramètres, la formule (4.5) fournit la longueur minimale nécessaire à cette seconde approche :

$$N'_t = L_{\max} + \frac{\left(5 + 3\sqrt{1 - (2\varepsilon_{\min})^{L_{\max}}}\right)^2}{(2\varepsilon_{\min})^{L_{\max}}} \ .$$

L'abaque 4.5 montre que ce nombre est beaucoup plus important que le nombre de bits de chiffré nécessaire par l'attaque utilisant seulement les multiples de poids $d$ et de degré au plus $D$ (voir la formule (4.6)). De plus le

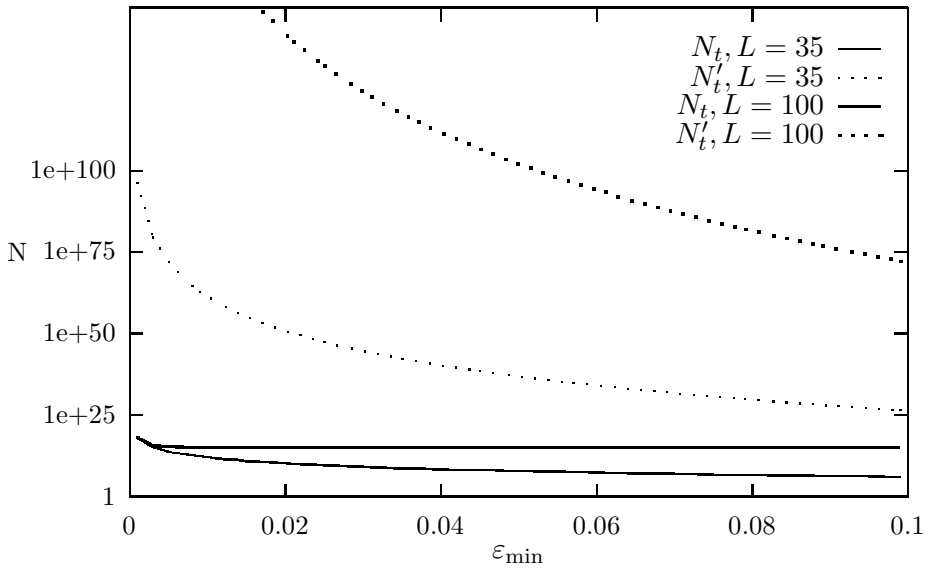


FIGURE 4.5 – Nombre minimum de bits de chiffré nécessaire aux deux approches de reconstruction pour $L_{max} = 35$ et 100

nombre d'opérations effectuées lors de cette seconde approche est environ :

$$\frac{d2^{L_{\max}}\left(5+3\sqrt{1-(2\varepsilon_{\min})^{L_{\max}}}\right)^2}{2(2\varepsilon_{\min})^{L_{\max}}} .$$

Ceci montre que l'algorithme de reconstruction que j'ai mis au point est beaucoup plus efficace que l'énumération de tous les polynômes de degré $L_{\max}$.

### 4.2.5 Simulations numériques

J'ai considéré le générateur à combinaison de registres suivant (cas d'école). Trois registres à décalage sont combinés par la fonction majorité :

$$f(x_1, x_2, x_3) = x_1x_2 \oplus x_1x_3 \oplus x_2x_3 .$$

Les polynômes de rebouclage qui ont été choisis sont respectivement

$$\begin{aligned} P_1(x) &= 1 \oplus x^2 \oplus x^4 \oplus x^5 \oplus x^6 \oplus x^8 \oplus x^9 \oplus x^{10} \oplus x^{11} \oplus x^{13} \oplus \\ &\quad x^{14} \oplus x^{15} \oplus x^{17} \\ P_2(x) &= 1 \oplus x \oplus x^7 \oplus x^8 \oplus x^{10} \oplus x^{12} \oplus x^{15} \oplus x^{16} \oplus x^{17} \oplus x^{20} \oplus x^{21} \oplus \\ &= x^{22} \oplus x^{23} \\ P_3(x) &= 1 \oplus x \oplus x^3 \oplus x^6 \oplus x^7 \oplus x^8 \oplus x^{13} \oplus x^{16} \oplus x^{19} \oplus x^{20} \oplus x^{25} \oplus \\ &\quad x^{26} \oplus x^{27} \oplus x^{28} \oplus x^{29} \oplus x^{31} \oplus x^{33} \end{aligned}$$

La séquence produite par ce générateur est combinée bit à bit par addition modulo 2 à une suite de texte clair telle que $p_0 = 0.70$. J'ai utilisé 6 109 messages de longueur 10 000.

L'algorithme a été appliqué sur ces messages avec les paramètres de degré maximum $D = 5,910$ et de densité $d = 3$ et 4. J'ai pris $\varepsilon_{\min} = 0.1$ ; cette valeur correspond à celle de la formule (4.2) avec $n = 3$. Le tableau 4.2 donne tous les polynômes primitifs détectés par l'algorithme pour $d = 3$ et $d = 4$. Le nombre de multiples obtenus pour ces polynômes est comparé avec les valeurs théoriques données par la formule (2.2). Le

| | $d$ | Nb détecté | $P_1$ | $P_2$ | $P_3$ |
|---|---|---|---|---|---|
| simulations | 3 | 126 | 123 | 3 | 0 |
| théorie | 3 | 139 | 137 | 2 | 0 |
| simulations | 4 | 266,191 | 262,043 | 4,146 | 3 |
| théorie | 4 | 266,589 | 262,483 | 4,102 | 4 |

TABLE 4.2 – Polynômes détectés pour le cas d'école

nombre théorique d'opérations effectuées par l'algorithme est $2^{50}$ pour $d = 3$ et $2^{61}$ pour $d = 4$. Cette simulation a nécessité 4 jours de calcul pour tester tous les bons $d$-multiples et 0.5% de tous les $d$-multiples possibles sur un Pentium II 400 sous Linux. Pour chaque $P_i(x)$ $P_j(x)$ détectés, on calcule les 5-nômes sur $D = 5,910$ bits pour chaque $P_i(x)P_j(x)$. Seul $P_1(x)P_2(x)$ est potentiellement détectable selon la formule (2.2). Il ne l'a pas été. Cela est normal car $\widehat{\chi}_f(011) = 0$. Il n'existe pas de corrélation exploitable pour ce produit de polynômes.

## 4.3 Reconstruction de la fonction de combinaison

Récemment (voir [134], 1999) la reconstruction de la fonction de combinaison d'un schéma de générateur à combinaison de registres a été présentée. Mais en fait, quoique intéressante, l'approche décrite souffre de sérieuses limitations pour en faire une technique exploitable opérationnellement :

- La reconstruction de la fonction de combinaison requiert au préalable de retrouver les initialisations de tous les registres, supposés connus par les auteurs. Autrement dit, il faut d'abord cryptanalyser le système.
- L'attaque utilisée est celle de Siegenthaler [154] nécessitant une complexité exponentielle.
- La reconstruction de la fonction utilise la distribution multinomiale des $2^n$ entrées de la fonction, que l'on doit donc explorer au préalable, requiérant à nouveau une complexité exponentielle. De plus, elle suppose de disposer d'une quantité de textes chiffrés bien plus importante.

Je vais présenter, à présent comment contourner ces limitations et comment pratiquement reconstruire la fonction de combinaison. De l'étape précédente (reconstruction des polynômes), les données suivantes ont été obtenues sur la fonction $f$ :

1. Le nombre de variables de la fonction de combinaison est déduit de cette étape. C'est le nombre de polynômes de rebouclage différents détectés.
2. L'analyse précédente (étape 6 de l'algorithme) fournit en même temps une estimation de certains coefficients de Walsh de la fonction de combinaison. Supposons que certains multiples de densité $d$ de

$$\prod_{i \in T} P_i, \quad T \subset \{1, \ldots, n\}$$

aient été détectés par l'algorithme. Pour les multiples concernés, la valeur moyenne de l'estimateur $Z$ vaut

$$N(2p-1)^d, \text{ où } p = Pr[c_t = s_t]$$

avec $s = g(s^1, \ldots, s^n)$ et $g(x) = 1_T \cdot x$. Les valeurs de $Z$ obtenues pour tous les multiples détectés de $\prod_{i \in T} P_i$ fournissent donc une estimation de la probabilité $p$.

A l'aide de la formule (4.1), il est possible de calculer la valeur du coefficient de Walsh correspondant $\widehat{\chi_f}(1_T)$. Cette valeur est arrondie au multiple de 4 le plus proche, comme tous les coefficients de Walsh sont divisibles par 4, pour une fonction booléenne équilibrée.

Si $\prod_{i\in T} P_i$ est de degré $L$ supérieur à $L_{\max}$, aucun multiple ne sera détecté par l'algorithme. Il suffit alors de choisir une valeur plus grande pour $d$ satisfaisant

$$(d-1)!^{\frac{1}{d-1}} 2^{\frac{L}{d-1}} \leq LC \ .$$

On calcule alors tous les multiples de $\prod_{i\in T} P_i$ de densité $d$ et de degré au plus $LC$, et les valeurs correspondantes de $Z$. Les coefficients de Walsh restants à déterminer sont alors obtenus en réïtérant l'étape de reconstruction pour $d+1$, $d+2$,...

**Exemple 9** *Dans le cas d'école vu précédemment (section 4.2.5, la valeur moyenne de l'estimateur $Z$ (e.g. 12.384 pour $P_1(x)$) obtenue pour chaque multiple de poids 3 de $P_1$ fournit exactement*

$$P[z_t = 0] = 0.5005 \quad \textit{et } P[s_t = y_t] = 0.55.$$

*La formule (4.1) donne alors (si on note $x = (x_3, x_2, x_1)$) :*

$$\widehat{\chi_f}(0,0,1) = 4.00.$$

*De la même manière, on obtient les données suivantes pendant l'étape de reconstruction des registres :*

$$\widehat{\chi_f}(0,1,0) = \widehat{\chi_f}(1,0,0) = 4 \quad \widehat{\chi_f}(0,0,0) = 0 \ .$$

*Pour chaque $P_i$ détecté, on calcule ses multiples de densité 5 et de degré au plus 10,000 pour chaque produit $P_iP_j$. Bien que ces produits soient potentiellement détectables, aucun ne l'a été. On en déduit raisonnablement que*

$$\widehat{\chi_f}(1,1,0) = \widehat{\chi_f}(1,0,1) = \widehat{\chi_f}(0,1,1) = 0 \ .$$

*Des simulations similaires pour $d = 7$ permettent de déterminer le coefficient restant :*

$$\widehat{\chi_f}(1,1,1) = -4 \ .$$

Les coefficients de la forme algébrique normale de la fonction sont obtenus à partir des coefficients de Walsh, en appliquant le résultat que j'ai établi avec la proposition 14.

# Chapitre 5

# Combinatoire et fonctions booléennes

## 5.1 Introduction

Nous avons montré dans la section 2.4 combien les fonctions booléennes sont la clef de voute de la sécurité des systèmes de chiffrement par flot. Il est donc nécessaire de pouvoir les classer, les étudier afin de disposer d'un corpus de fonctions, disposant de telles ou telles propriétés. En particulier, les fonctions $k$-résilientes, c'est-à-dire les fonctions à la fois équilibrées et immunes aux corrélations à l'ordre $k$, les fonctions hautement non linéaires, sont très importantes en cryptologie [58].

Les fonctions booléennes sont également très utilisées dans d'autres domaines (conception et tests de circuits électroniques, théorie de la complexité,...). Là, c'est généralement la propriété d'équilibre qui est principalement recherchée.

L'objet de ce chapitre est double. Dans un premier temps, il s'agit de présenter un début de tentative de classification des fonctions booléennes selon leur poids ou au minimum selon leur famille de poids. Soit une fonction $f : \mathbb{F}_2^n \to \mathbb{F}_2$. Le poids d'une fonction booléenne $f$ est défini par

$$wt(f) = |\{x \in \mathbb{F}_2^n | f(x) = 1\}|$$

Une fonction sera dite *équilibrée* si son poids est égal à $2^{n-1}$ autrement dit si cette fonction vaut autant de fois 0 que 1.

La notion de poids est une notion difficile à caractériser et à étudier. La seule manière de l'évaluer repose dans le cas général, sur des méthodes énumératives. Autrement dit, évaluer le poids d'un objet mathématique,

comme une fonction booléenne, est de complexité exponentielle. En particulier identifier des fonctions équilibrées, et plus généralement pouvoir effectivement construire des fonctions booléennes équilibrées non triviales est difficile.

Considérons d'abord les applications pratiques pour lesquelles le poids est important. Dans la section 2.4, l'importance de l'équilibre d'une fonction a été présentée, pour la cryptologie (en effet le non-équilibre signifie relative prédictabilité, c'est-à-dire faiblesse). En conception de circuits logiques [38], en test de circuits logiques [39, 40], il est particulièrement utile de disposer de fonctions équilibrées. Elles permettent la réalisation pratique de circuits optimaux ou de méthodes de test efficaces. De plus disposer de fonctions d'une classe de poids connue (sur- ou sous-équilibrées) permet de construire des fonctions qui elles sont équilibrées. Dans [38], K. Chakrabarty et J.P. Hayes ont donné plusieurs méthodes de construction de familles de fonctions équilibrées, soit à l'aide d'autres fonctions équilibrées, soit à l'aide de fonctions non-équilibrées.

Dans un contexte d'études fondamentales, la propriété d'équilibre (et plus généralement toute connaissance, même partielle sur le poids) joue un grand rôle. Elle permet de caractériser et d'étudier théoriquement certains objets et propriétés. A titre d'exemple, toujours emprunté à la cryptologie, nous pouvons citer le critère de propagation. Il est précisément défini à l'aide de fonctions équilibrées [26, 27, 141, 169]. Récemment, dans [22, sections 5 et 6], plusieurs résultats ont été obtenus sur ce critère (notamment en liaison avec la non-linéarité). Ainsi, les auteurs ont caractérisé les fonctions optimales pour ce critère, de degré 3, en montrant qu'elles possèdaient de nombreuses fonctions dérivées équilibrées. Cela a permis de nouvelles caractérisations des fonctions *courbes*, entre autres.

Les fonctions courbes ont la propriété extrêmement importante d'être à égale distance (au sens de Hamming) de toute fonction linéaire. Or ces fonctions ne sont jamais équilibrées. Pour contourner ce problème, diverses constructions ont été envisagées. Soit des fonctions partiellement courbes ont été définies [26] ou bien le caractère équilibré a été obtenu seulement avec une haute non linéarité, à partir des fonctions courbes [50]. Il est alors facile à travers ces deux exemples de mesurer combien la connaissance du poids d'une fonction est important, mais pas forcément facile à déterminer, surtout si on veut conjuguer cette connaissance à d'autres propriétés.

L'approche que j'ai choisie est de trouver des propriétés structurelles d'une représentation pratique des fonctions booléennes, vérifiables en une complexité la plus basse possible (idéalement polynomiale), permettant à la fois la construction et la caractérisation de l'équilibre, avec éventuellement

d'autres propriétés. Dans le cas du non-équilibre, je souhaite au moins avoir certaines bornes sur le poids (au minimum pouvoir décider entre sous-équilibre et sur équilibre), toujours à l'aide de propriétés structurelles faciles à obtenir et à vérifier. Cela me permettrait de jeter les bases d'une classification selon le poids, affinable dans le futur. Ces résultats ont été publiés dans [59].

En fait, l'équilibre est très souvent considéré comme une propriété annexe ou secondaire, c'est-à-dire que des propriétés plus fortes (haute non linéarité, immunité aux corrélations, caractère courbe,...) sont d'abord recherchés et l'équilibre vient ensuite comme une propriété nécessaire mais complémentaire (voir par exemple [21, 65]).

Ceci amène la deuxième préoccupation de ce chapitre. L'obtention simultanément de plusieurs fortes propriétés cryptologiques pour une fonction booléenne, passe obligatoirement par un compromis. Cette contrainte, présentée dans la section 2.4.5, provient de l'incompatibilité de certaines propriétés (équilibre et caractère courbe) ou de la difficulté à les concilier (degré et immunité aux corrélations, haute non linéarité et immunité aux corrélations,...).

Pour le concepteur de systèmes de chiffrement, à flot en particulier, deux alternatives sont à considérer :

- Soit rechercher le meilleur compromis possible, ce qui devient très vite impossible dès que le nombre de variables dépasse 13. De plus, l'obtention d'un compromis optimal peut très bien cacher des faiblesses structurelles que des cryptanalyses futures exploiteront avec profit (variables séparées par exemple).
- Soit accepter un compromis légèrement sous-optimal, en "amoindrissant" la propriété la "moins essentielle". Un choix judicieux de conception permettra alors de contrebalancer cette relative et légère faiblesse.

Caroline Fontaine, grâce à une approche exploratoire des translatés du code de Reed-Müller d'ordre 1, a exhibé de très nombreuses fonctions réalisant une excellent compromis [65]. Avec elle nous avons montré qu'elles présentaient d'excellentes propriétés cryptologiques et comment les utiliser en concevant un système de chiffrement par flot générique extrêmement robuste. Ces résultats ont été publiés à Eurocrypt'98 [58].

J'ai choisi de considérer dans les deux parties la *Forme Algébrique Normale (FAN)* pour représenter une fonction booléenne (Algebraic Normal Form (A.N.F.)) et plus précisément dans la première partie ses propriétés combinatoires. Cette approche avait déja été initiée [20] pour caractériser l'immunité aux corrélations à l'aide de tableaux orthogonaux en considérant la table de vérité de la fonction. Récemment [179], la forme algébrique nor-

male a été utilisée pour étudier la non-linéarité au travers du nombre de ses termes. Outre qu'elle est la plus classique et la plus usuelle, dans tous les domaines où interviennent les fonctions booléennes, elle présente de plus l'avantage d'être une représentation compacte.

Après avoir rappelé succinctement quelques concepts et notations concernant les fonctions booléennes, je présenterai rapidement les objets et propriétés combinatoires utilisées. Je présenterai ensuite une nouvelle description, combinatoire, de la forme algébrique normale, à l'aide de *designs* (définition 28). Je montrerai ensuite comment construire plusieurs classes infinies de fonctions ayant une poids déterminé ou appartenant à une famille déterminée de poids (théorème 9). Elles sont construites à partir de structures possédant de fortes propriétés de régularité (designs symétriques par exemple).

Dans le cas de ces fonctions, il devient alors facile de calculer leur poids (ou d'en déterminer la famille) en considérant uniquement leur FAN. Je montrerai que l'abandon de certaines propriétés pour les structures combinatoires utilisées, permet alors d'obtenir des fonctions équilibrées (théorème 10).

Je présenterai ensuite plus particulièrement les fonctions majorité. Un résultat nouveau m'a permis de complètement caractériser la FAN de ces fonctions (proposition 32) permettant ainsi désormais de l'obtenir en temps polynomial, sans la calculer. Je montrerai également que ces fonctions sont mauvaises en cryptographie, quand le nombre de variables est petit (proposition 33).

Enfin je présenterai succinctement les fonctions cryptologiquement fortes construites par Caroline Fontaine et décrirai une manière de les utiliser pour obtenir un système de chiffrement par flot extrêmement sûr. J'expliquerai également comment l'absence de telle ou telle propriété peut mener à une attaque.

Dans ce chapitre, je me limiterai aux fonctions booléennes sans terme constant car $wt(1 \oplus f) = 2^n - wt(f)$. De façon évidente, à une fonction sous-équilibrée $f(x)$ correspondra une fonction sur-équilibrée $1 \oplus f(x)$. Pour les autres propriétés cryptologiques, il y a invariance.

## 5.2 Concepts de base et notation

### 5.2.1 Fonctions booléennes

Comme je vais utiliser la forme algébrique normale d'une fonction booléenne, j'en rappelle la définition, donnée au chapitre 2. Je complèterai ensuite

par quelques notations et définitions spécifiques à cette étude.

Soit $f$ une fonction booléenne, $f : \mathbb{F}_2^n \to \mathbb{F}_2$. Sa forme algébrique normale sera décrite par

$$f(x_1, x_2, \ldots, x_n) = \sum_{\alpha \in \mathbb{F}_2^n} a_\alpha x^\alpha \qquad a_\alpha \in \mathbb{F}_2$$

où $x = (x_1, x_2, \ldots, x_n)$ et $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$ avec $\alpha_i \in \mathbb{F}_2$. Les coefficients $a_\alpha$ de la FAN peuvent être obtenus grâce à la transformée de Möbius [114] (voir [82] pour un exposé exhaustif de l'utilisation de cette transformée dans le cadre des fonctions booléennes) :

$$a_\alpha = g(\alpha) = \sum_{\beta \preceq \alpha} f(\beta_1, \beta_2, \ldots, \beta_n) \tag{5.1}$$

où $\beta \preceq \alpha$ décrit l'ordre partiel sur le treillis des sous-ensembles de $\mathbb{F}_2^n$. Autrement dit, $\beta \preceq \alpha \Leftrightarrow \beta_i \leq \alpha_i \ \forall i$. En utilisant la formule d'inversion de Möbius, on montre que la transformée de Möbius est involutive (pour une démonstration voir [82, p. 43, proposition II.11]) et on a

$$f(x) = \sum_{\beta \preceq x} g(\beta). \tag{5.2}$$

Dans un contexte binaire, nous pouvons noter tout monôme de $f$ par $\alpha$ et la FAN elle-même comme l'ensemble :

$$\mathcal{A}(f) = \{\alpha \in \mathbb{F}_2^n | a_\alpha = 1\}$$

Cet ensemble fournit un outil et une notation plus pratiques pour manipuler cette FAN. Le *degré minimum* de $f$ ou de sa FAN, $d_{min}(f)$ est défini par :

$$d_{min}(f) = \min_{\alpha \in \mathcal{A}(f)} \{wt(\alpha)\}$$

où *wt(.)* désigne le poids de Hamming. Le degré minimum est donc le degré total minimum des différents monômes de la FAN. Le degré de $f$, qui est vu comme un polynôme multivarié, sera noté $deg(f)$.

J'utiliserai également la notion de *support* d'un monôme $\alpha \in \mathbb{F}_2^n$ :

$$supp(\alpha) = \{i \in \mathbb{N}_n^* | \alpha_i = 1\}$$

Ainsi $supp(\alpha)$ est un sous-ensemble de $\mathbb{N}_n^* = \{1, 2, \ldots, n\}$ lorsque l'on considère les coordonnées non nulles de $\alpha$ dans sa décomposition en base 2.

De façon équivalente, $supp(\alpha)$ est identifiée à $\alpha$ lui-même dans le contexte binaire.

J'aurai ensuite besoin de la notion de *couverture* d'un monôme $\alpha$ que je noterai :

$$cov(\alpha) = \{\beta \in \mathcal{A}(f) | supp(\beta) \subseteq supp(\alpha)\}$$

Le cardinal de cet ensemble sera noté $|cov(\alpha)|$.

Quand $\alpha \in \mathcal{A}(f)$ alors $|cov(\alpha)| \equiv 1 \bmod 2$. Il est facile de montrer que cette valeur est égale au coefficient de Möbius $a_\alpha$. La couverture de $\alpha$ sera dite impaire si $|cov(\alpha)| \equiv 1 \bmod 2$ et paire autrement. Enfin, une valeur donnée $a \in \mathbb{F}_2^n$ couvre $\alpha \in \mathcal{A}(f)$ si $supp(\alpha) \subseteq supp(a)$, noté de façon équivalente, si $\alpha \preceq a$. $\alpha <> \beta$ indiquera que $\alpha$ and $\beta$ sont non comparables pour cet ordre partiel (antichaine).

Illustrons tout cela par un exemple.

**Exemple 10** *Soit la fonction booléenne sur $\mathbb{F}_2^3$ donnée par*

$$f(x_3, x_2, x_1) = x_1 + x_1x_2 + x_2x_3 + x_1x_2x_3$$

*Alors nous avons*

$$\mathcal{A}(f) = \{(0,0,1), (0,1,1), (1,1,0), (1,1,1)\}$$

*Soit la valeur $a = (1,1,1)$ alors*

$$supp(a) = supp((1,1,1)) = \{1,2,3\}$$

*et a couvre tous les monômes. On a $|cov(a)| = 1$ d'où la présence du monôme $x_1x_2x_3$ dans la FAN. De plus $d_{min}(f) = 1$.*

Les notations $f(x)$ et $f(A)$ pour un quelconque $A \subset \mathbb{N}_n^*$ et $A = supp(x)$ seront considérées comme équivalentes. Evaluer l'une d'entre elles consiste à compter les monômes $\alpha \preceq x$ (par involutivité de la transformée de Möbius). Enfin, $f(\bar{x}) = f(\bar{A})$ désignera $f(\mathbb{N}_n^* \backslash A) = f(x + \bar{1})$ avec $\bar{1} = (1,1,1,\ldots,1,1)$.

Un résultat connu [154] affirme que toute fonction à $n$ variables de degré $n$ est de poids impair. Cela implique que la FAN d'une fonction booléenne équilibrée est de degré $deg(f) < n$. Je donne maintenant une caractérisation comparable avec le degré minimal de $f$.

**Proposition 27** *Soit $f$ une fonction sur $\mathbb{F}_2^n$. Si $d_{min}(f) > \lceil \frac{n}{2} \rceil$ alors $f$ est sous-équilibrée ($wt(f) < 2^{n-1}$).*

*Preuve.*
Si $d_{min}(f) > \lceil \frac{n}{2} \rceil$, alors $f(x) = 1$ seulement si $wt(x) \geq d_{min}$ par involutivité de la transformée de Möbius (équation 5.2). En effet si $\forall \beta \in \mathbb{F}_2^n$ tel que $wt(\beta) < d_{min}(f)$ alors $a_\alpha = 0$. Selon que $n$ est pair ou impair, en utilisant l'expansion polynomiale de $2^n$, on voit facilement que moins de $2^{n-1}$ telles valeurs donnent $f(x) = 1$. ■

**Exemple 11** *Soit la fonction sur* $\mathbb{F}_2^6$ *:*

$$f(x_1, \ldots, x_6) = x_1x_2x_3x_4 \oplus x_1x_2x_5x_6$$

*Cette fonction a un poids de* $6 < 2^5$.

### 5.2.2 Designs et familles intersectantes

Les *designs* sont des objets combinatoires étudiés et utilisés dans de très nombreux domaines. On trouve des applications en combinatoire (théorie des graphes en particulier [19], géométries finies,....), en théorie des codes [3, 19], en radioastronomie [159], en informatique [44, pp 543–549] (en particulier avec les mémoires de type WOM (*Write-Once Memory*) [43]), en cryptologie [44, pp 549–557], en statistiques,...

La première classe connue de designs est constituée des systèmes triples de Steiner notés $S(t, k, v)$ ou encore $t - (v, k, 1)$. Leur introduction est attribuée à Woolhouse [171] en 1844 et non à Steiner. Ce dernier ne s'y intéressa qu'en 1853, quand il étudia la configuration de 28 tangentes doubles d'une courbe plane [161].

En fait, la célébrité de ces objets [1] vient du fameux problème [105] de T.P. Kirkman, prêtre de l'église anglicane, *le problème des 15 écolières* :

> 15 écolières défilent en rang par 3 chaque jour d'une semaine.
> On veut les disposer journellement de sorte qu'aucune d'entre
> elles ne défile dans la même rangée de 3 plus d'une fois.

En fait, cela revient à chercher un design par bloc, résoluble de paramètres $t = 2$, $v = 15, k = 3$ et $\lambda = 1$. La solution fut publiée la première fois en 1971 [143] [2].

Mais l'une des contributions les plus importantes est certainement celle de Sir R.A. Fisher, l'un des statisticiens les plus éminents du 19ème siècle.

1. L'histoire, qui jamais n'est censée se répéter est identique pour les carrés latins et le problème des 36 officiers

2. A noter qu'une solution antérieure est maintenant connue, due en 1965, à un mathématicien de Mongolie [97]. L'article n'a été publié que beaucoup plus tard, en raison de la révolution culturelle chinoise.

Il a notamment systématisé l'usage des designs dans les plans factoriels d'expériences (voir sur ce sujet la monographie de D.C. Montgomery [130] ; voir aussi [99, 64]), à l'époque pour l'expérimentation en agriculture. Ses études ont d'ailleurs déterminé la plupart des notations en théorie des designs ($v$ le nombre de points vient de variétés et $r$ est le nombre de "repiquage"). On lui doit notamment la fameuse inégalité dite de Fisher concernant la borne inférieure du nombre de blocs d'un design.

**Définition 26** *Un $t-(v,k,\lambda)$ design est une paire $(\mathcal{V},\mathcal{B})$ où $\mathcal{V}$ est un ensemble de $v$ points et $\mathcal{B}$ est une collection de sous-ensembles de $\mathcal{V}$ de $k$ points (blocs) satisfaisant la propriété que tout sous-ensemble de $t$ points de $\mathcal{V}$ est contenu dans exactement $\lambda$ blocs.*

Quand $t = 2$ un tel design est appelé un *design par blocs, incomplet et équilibré* (*Balanced Incomplete Block Design (BIBD)*. J'utiliserai l'acronyme anglo-saxon dans la suite ce dernier étant quasi universellement employé. Dans ce cas, deux autres paramètres sont importants : $r$ le nombre de répétitions, c'est-à-dire le nombre de blocs dans lesquels figure chaque point et $b$ qui est le nombre $|\mathcal{B}|$ de blocs du design.

**Exemple 12** *Soit l'ensemble $\mathcal{V}$ contenant 7 points et*

$$\mathcal{B} = \{\{0,1,3\},\{1,2,4\},\{2,3,5\},\{3,4,6\},\{4,5,0\},\{5,6,1\},\{6,0,2\}\}.$$

*On a alors l'exemple du plus célèbre des designs, le* plan de Fano *ou encore plan projectif d'ordre 2. C'est un $2-(7,3,1)$ BIBD. Les 7 blocs sont représentés par les lignes (droites ou circulaires ; voir figure 5.1)*

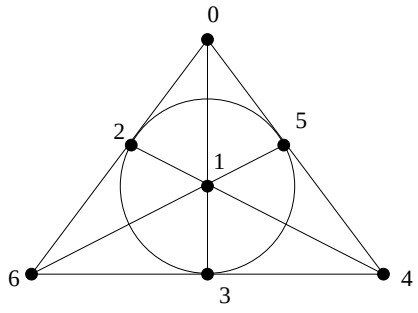


FIGURE 5.1 – Plan de Fano

Ces paramètres étant à valeurs entières, certaines conditions sont nécessaires pour assurer l'existence d'un design donné (*paramètres admissibles*) :

- $r(k-1) = \lambda(v-1)$
- $bk = vr$
- $v \geq b$ (Inégalité de Fisher)

Dans un contexte binaire, je ne considérerai que des BIBD *simples*, c'est-à-dire des designs dans lesquels aucun bloc n'est répété. De plus, un design sera appelé *complet* s'il est simple. Il contient $\binom{v}{k}$ blocs et $\lambda = \frac{\binom{v}{k}k(k-1)}{v(v-1)}$.

Un BIBD est *symétrique* (SBIBD) quand $b = v$ ou de façon équivalente $k = r$. Alors $\lambda$ représente le nombre, constant, de points dans chaque intersection de deux blocs. Ces designs possèdent donc de très fortes propriétés de régularité et de symétrie. Si l'on considère le dual $\mathcal{D}^* = (\mathcal{B}, \mathcal{V})$ d'un SBIBD $\mathcal{D}$, le dual d'un design est également un design si et seulement si il est sumétrique. Les paramètres de $\mathcal{D}$ et $\mathcal{D}^*$ sont les mêmes sans qu'ils soient nécessairement isomorphes. Cette notion de dualité signifie que points et blocs sont interchangeables. Le plan de Fano (voir exemple 5.1) est un design symétrique.

On appelle *arc* un sous-ensemble de $s$ points de $\mathcal{V}$ ne contenant aucun bloc de $\mathcal{B}$ et un $s$-arc sera dit complet si il n'est pas strictement contenu dans un $(s+1)$-arc.

Afin d'unifier la notation entre les designs et les fonctions booléennes, je remplacerai $v$ par $n$ où $n$ décrira le nombre de variables de la fonction. Ainsi je parlerai de $(n, k, \lambda)$ designs lorsqu'ils seront utilisés pour étudier une telle fonction.

Un autre concept combinatoire sera utilisé : celui de famille intersectante.

**Définition 27** *[18] Soit $\mathcal{F}$ une famille de sous-ensembles d'un ensemble de $n$ points. $\mathcal{F}$ est dite intersectante si*

$$\forall A \in \mathcal{F},\ \forall B \in \mathcal{F}, \quad A \cap B \neq \emptyset$$

La propriété la plus intéressante concernant ces objets combinatoires, très importante pour caractériser les fonctions booléennes équilibrées est la suivante :

**Proposition 28** *Soit $\mathcal{F}$ une famille intersectante de sous-ensembles d'un ensemble de $n$ points. Alors*

$$|\mathcal{F}| \leq 2^{n-1}.$$

*Il existe des familles intersectantes atteignant cette borne.*

Les familles intersectantes de taille $2^{n-1}$ sont trop nombreuses pour envisager ne serait ce qu'une classification. Pour plus de détails, voir [18] et pour une présentation exhaustive des designs, voir [8, 18, 44, 69].

## 5.3 Forme algébrique normale et SBIBD

La très forte régularité et les propriétés de symétrie des designs symétriques de type SBIBD m'ont suggéré qu'ils pouvaient être de très bons candidats pour la caractérisation et la construction de fonctions booléennes équilibrées. En effet, l'équilibre peut lui-même être considéré comme une propriété de régularité. Je me suis limité à considérer les SBIBD, ces objets combinatoires étant les plus fortement réguliers mais les résultats qui vont suivre peuvent être généralisés à d'autres $t-(v,k,\lambda)$ designs.

Je vais d'abord introduire et définir précisément comment j'utilise certains objets combinatoires comme les designs pour étudier la forme algébrique normale d'une fonction booléenne.

**Définition 28** *Soit $f:\mathbb{F}_2^n \to \mathbb{F}_2$ une fonction booléenne. Soit*

$$\mathcal{B} = \{supp(\alpha) | \alpha \in \mathcal{A}(f)\}$$

*et $\mathcal{V} = \mathbb{N}_n^*$. Alors $f$ et la structure $(\mathcal{V},\mathcal{B})$ seront dits associés. Quand $(\mathcal{V},\mathcal{B})$ est un $t-(n,k,\lambda)$ design pour des paramètres $n$, $k$ et $\lambda$ donnés, il est dénommé le* design associé *de $f$.*

Le structure $(\mathcal{V},\mathcal{B})$ peut être une des quelconques structures définies en combinatoire (PIBD, plan projectif, designs, tableaux orthogonaux, ...[44]). Ainsi dans le cas d'un $t-(n,k,\lambda)$ design, la forme algébrique normale contient seulement des monômes de degré total $k$, c'est-à-dire, $d_{min} = k = deg(f)$. Chaque bloc est en fait le support d'un monôme de la FAN.

D'où, évaluer $f(x)$ pour un quelconque $x \in \mathbb{F}_2^n$ consiste à considérer combien de blocs sont inclus dans $supp(x)$.

**Exemple 13** *La FAN de la fonction majorité d'ordre 3 est donnée par :*

$$f(x_3,x_2,x_1) = x_1x_2 + x_1x_3 + x_2x_3$$

*Elle est associée avec le $(3,2,1)$-design complet et symétrique avec*

$$\mathcal{V} = \{1,2,3\} \text{ et } \mathcal{B} = \{\{1,2\},\{1,3\},\{2,3\}\}$$

*Alors, $f((1,1,1)) = 1$ comme $supp((1,1,1)) = \{1,2,3\}$ contient les 3 blocs (monômes).*

Une première propriété concernant les SBIBD doit être établie pour démontrer les résultats principaux qui vont suivre.

**Proposition 29** *Dans un* $2-(n,k,\lambda)$ *SBIBD, avec* $n \neq k$*, quel que soit* $(b_i, b_j) \in \mathcal{B} \times \mathcal{B}$ *nous avons*

$$b_i \cup b_j = \mathcal{V} \quad \Leftrightarrow \quad k = n-1$$

*Preuve.*
Par définition $|b_i| = |b_j| = k$ et par propriété de symétrie $|b_i \cap b_j| = \lambda$. Si $b_i \cup b_j = \mathcal{V}$ alors nous avons $\lambda = 2 \cdot k - n$. A nouveau en considérant la propriété de symétrie nous avons également $\lambda = \frac{k \cdot (k-1)}{n-1}$. J'obtiens facilement l'équation suivante d'inconnue $k$ :

$$k^2 - (2n-1) \cdot k + (n^2 - n) = 0$$

Les seules solutions possibles donnent $k = n-1$. La réciproque est évidente. ■

*Remarque :* les SBIBD concernés par la proposition 29 sont les $(n, n-1, n-2)$-designs complets symétriques.

Pour illustrer cette approche combinatoire, il est possible d'obtenir différemment un résultat simple, généralement obtenu par une approche uniquement polynomiale :

**Proposition 30** *Soit* $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ *une fonction booléenne associée au* $(n, n-1, n-2)$*-design symétrique. Alors* $f$ *n'est jamais équilibrée sauf pour* $n \in \{2, 3\}$ *et son poids est donné par :*

$$wt(f) = \begin{cases} n & \text{si } n \text{ pair} \\ n+1 & \text{si } n \text{ impair} \end{cases}$$

*Preuve.*
Par définition, la FAN de $f$ contient seulement des monômes de degré $n-1$. Il s'ensuit que seuls les $x \in \mathbb{F}_2^n$ de poids $n$ et $n-1$ sont susceptibles de donner $f(x) = 1$ (par application de la proposition 29). C'est le cas pour ceux de poids $n-1$. Il y en a $n$ correspondant au nombre de blocs. Il suffit de vérifier pour la seule valeur $x = \bar{1}$ de poids $n$ et nous obtenons le résultat sur le poids. Pour que $f$ soir équilibrée, il faut résoudre $2^{n-1} = n$ ($n$ pair) ou $2^{n-1} = n+1$ ($n$ impair) ce qui correspond à $n \in \{2, 3\}$. ■

Les seules fonctions booléennes équilibrées associées aux SBIBD de la proposition 30 sont donc les fonctions linéaires à deux variables et la seule fonction de degré 2 de l'exemple 13. Pour généraliser la proposition 30, j'utiliserai la proposition suivante :

**Proposition 31** *Soit un $(n, k, \lambda)$ SBIBD. Alors le nombre de sous-ensembles de $\mathbb{N}_n^*$ contenant au moins un bloc du SBIBD est inférieur ou égal à $2^{n-1}$ .*

*Preuve.*
Utilisons le fait que les blocs du SBIBD forment une famille intersectante (puisque $\lambda \neq 0$). Il s'ensuit que les sous-ensembles contenant au moins l'un des blocs constituent également une famille intersectante. Le résultat s'obtient facilement en appliquant la proposition 28. ■

*Remarque :* Il n'est pas possible de prévoir l'égalité à $2^{n-1}$. Par exemple le $(7, 3, 1)$-SBIBD (plan projectif d'ordre 2) atteint cette borne mais son design complémentaire $(7, 4, 2)$ ne l'atteint pas.

Je peux à présent donner le résultat général que j'ai obtenu :

**Théorème 9** *Soit un $(n, k, \lambda)$ SBIBD avec $(n, k) \notin \{(2, 1), (3, 2)\}$ et $f$ sa fonction booléenne associée. Alors $f$ est sous-équilibrée autrement dit*

$$wt(f) < 2^{n-1}$$

*Preuve.*
Considérons seulement le cas $n > k$ sinon pour $k = n$ la fonction associée est de façon évidente sous-équilibrée (son poids vaut de plus 1). Considérons $\mathcal{U} = \{supp(x) | f(x) = 1\}$ et soit $\mathcal{F}$ la famille intersectante consistant en tous les sous-ensembles de $\mathbb{N}_n^*$ contenant au moins un bloc. Il est évident que $\mathcal{U} \subseteq \mathcal{F}$ ($f(x) = 1$ si il existe un nombre impair de blocs contenus dans $supp(x)$).
Si $|\mathcal{F}| < 2^{n-1}$ le résultat est immédiat.
Supposons que $\mathcal{F} = 2^{n-1}$ et considérons l'union $a = b_i \cup b_j$ de 2 blocs quelconques de $\mathcal{B}$ ($|a| = 2k - \lambda$). Alors $a \in \mathcal{U}$ si et seulement si le nombre de blocs inclus dans $a$ est impair. Montrons maintenant que l'union de deux blocs quelconques contient seulement ces deux blocs. Supposons donc qu'il existe un bloc $b_l$ tel que $b_l \subseteq (b_i \cup b_j)$ avec $i, j, k$ tous distincts. Ceci est équivalent à l'un des trois cas suivants, comme $|b_i \cup b_j \cup b_l| = 2k - \lambda = 3k - 3\lambda + |b_i \cap b_j \cap b_l|$ :

- $2k - \lambda = 3k - 2\lambda$ ($|b_l \cap b_j \cap b_i| = \lambda$). Cela conduit à la solution non valide $v = k$.
- $2k - \lambda = 3k - 3\lambda$ (toutes les intersections de deux blocs sont disjointes). Cette équation donne un $(2k - 1, k, \frac{k}{2})$ SBIBD, ou, comme $2k - 1$ est impair et $k$ pair à un $(4q - 1, 2q, q)$ SBIBD pour un $q$ donné (qui existe : il suffit de considérer le théorème de Brück-Ryser Chowla [44] et de

prendre $q = p^2$). Considérons maintenant $\bar{\mathcal{U}} = \{x | f(x) = 0\}$. Ainsi $|\bar{\mathcal{U}}| \geq \underbrace{\sum_{i=0}^{k-1} \binom{n}{i} + \binom{n}{k}}_{>2^{n-1}} - b$ où $b = \frac{\lambda n(n-1)}{k(k-1)}$ est le nombre de blocs. Alors $|\mathcal{U}| = 2^n - |\bar{\mathcal{U}}| < 2^{n-1}$.

- $2k - \lambda = 3k - 2\lambda' - 3\lambda''$ ($|b_l \cap b_j \cap b_i| = \lambda'$ and $\lambda = \lambda' + \lambda''$ with $\lambda' \neq 0$ and $\lambda'' \neq 0$ ; rappelons que $\lambda$ dans un SBIBD représente le nombre de points communs à deux blocs quelconques. On obtient facilement l'inégalité $\lambda < k < 2\lambda$. (en prenant $\lambda' < \lambda$). Comme $\lambda = \frac{k(k+1)}{n-1}$ est équivalent à $k < n < 2k - 1$ c'est-à-dire $k > \frac{n+1}{2}$. Dans ce cas, la fonction est sous-équilibrée comme le degré minimal $d_{min}$ est trop élevé.

Le théorème est prouvé. ■

Le théorème 9 permet donc de construire facilement des fonctions booléennes non triviales sous-équilibrées. Inversement, avec la seule FAN, il est facile de déterminer que ces fonctions sont de cette famille de poids par vérification pour les monômes de la structure de design.

**Exemple 14** *L'ensemble $\mathcal{V}$ contenant 7 points, la fonction boolénne associée comportera sera définie sur $\mathbb{F}_2^7$. Comme*

$$\mathcal{B} = \{\{0,1,3\},\{1,2,4\},\{2,3,5\},\{3,4,6\},\{4,5,0\},\{5,6,1\},\{6,0,2\}\}.$$

*La FAN associée sera donnée par*

$$\begin{aligned} f(x_0,\ldots,x_6) &= x_0x_1x_3 \oplus x_1x_2x_4 \oplus x_2x_3x_5 \oplus x_3x_4x_6 \\ &\quad \oplus x_0x_4, x_5 \oplus x_1x_5x_6 \oplus x_0x_2x_6. \end{aligned}$$

*Le poids de $f$ est de 36. Cette fonction est bien sous-équilibrée.*

*Remarque :* si $b$ est pair le résultat est immédiat comme $f(\bar{1}) = 0$ d'où $\mathcal{U} < 2^{n-1}$. La définition suivante sera utile plus loin :

**Définition 29** *Soit $f$ une fonction booléenne dont la forme algébrique normale satisfait la propriété de famille intersectante (i.e. deux blocs quelconques sont d'intersection non nulle). La famille intersectante associée $\mathcal{F}$ est la famille intersectante dont tout élément contient le support d'au moins un monôme de $\mathcal{A}(f)$.*

**Corollaire 4** *Soit un $(n, k, \lambda)$ SBIBD et soit $f$ sa fonction booléenne associée ($(n, k) \notin \{(2, 1), (3, 2)\}$). Alors :*

$$\sum_{i=0}^{k-\lambda-1} b\binom{n-k}{i} < wt(f) < 2^{n-1}$$

*où $b = \frac{\lambda n(n-1)}{k(k-1)}$ est le nombre de blocs du SBIBD.*

*Preuve.*
La borne supérieure vient du théorème 9. La borne inférieure décrit le nombre de valeurs $x$ couvrant seulement un bloc (et $f(x) = 1$), dont le poids varie de $k$ à $2k - \lambda - 1$ (comme l'intersection de deux blocs quelconques contient $\lambda$ points). ■

Pour obtenir une meilleure borne inférieure, il faudrait dénombrer le nombre de $s$-arcs ($2k - \lambda \leq s \leq m$, où $m$ représente la taille maximale d'arc dans un SBIBD donné). Or énumérer ces arcs demeure encore un problème ouvert. Seuls quelques résultats d'existence sont connus (en particulier pour les géométries partielles ; pour plus de détails voir [9, 91, 164])

Ce corollaire donne une caractérisation intéressante d'une famille de fonctions booléennes sous-équilibrées (et bien sûr de fonctions sur-équilibrées en prenant $1 + f(x)$) de poids borné, seulement à l'aide de la forme algébrique normale de ces fonctions. Le théorème 9 illustre et renforce un principe général et fondamental en cryptologie : **la présence de structure signifie souvent faiblesse**.

Ici les fortes propriétés des SBIBD confèrent à leur fonction booléenne associée un déséquilibre dans la répartition des valeurs qu'elles prennent, ce qui constitue une faiblesse cryptologique. Cependant si nous abandonnons tomber la propriété de symétrie des SBIBD, nous obtenons le résultat très important suivant :

**Théorème 10** *Soit $f : \mathbb{F}_2^n \to \mathbb{F}_2$ avec $n = 2^q - 1$, dont la forme algébrique normale décrit un $(2^q - 1, 2^{q-1}, \binom{2^q-3}{2^{q-1}-2})$ design (design associé). Alors $f$ est équilibrée.*

*Preuve.*
Une telle fonction (le design associé étant complet) contient $\binom{2^q-1}{2^{q-1}}$ monômes (blocs). La famille intersectante associée est de taille $2^{n-1}$ où $n = 2^q - 1$. Pour deux blocs distincts quelconques $b_i$ et $b_j$ de ce design, nous avons $|b_i \cap b_j| \in K = \{1, 2, \ldots, \frac{n-1}{2}\}$ et $|b_i \cup b_j| = |b_i| + |b_j| - |b_i \cap b_j|$. Autrement

dit $|b_i \cup b_j| = 2(\frac{n+1}{2}) - k = n + 1 - k$ où $k \in K$. Ainsi $f$ sera équilibrée si l'union de deux blocs couvre un nombre impair de blocs (sans perte de généralité, il suffit de considérer l'union de seulement deux blocs ; en fait dans la preuve du théorème (32), nous montrerons que considérer seulement ce cas est suffisant) ou de façon équivalente si

$$\forall k \in K \qquad \binom{n+1-k}{\frac{n+1}{2}} \equiv 1 \qquad mod\ 2$$

Comme $supp(\frac{n+1}{2}) \preceq supp(n+1-t)$, j'obtiens le résultat en appliquant le théorème de Lucas [7, 113] ■

## 5.4 Structure de la FAN des fonctions majorité

Les designs du théorème 10 sont les designs complets d'ordre $2^q - 1$. Les fonctions associées sont les fonctions $MAJ_{2^q-1}$. Afin de le prouver, j'établirai d'abord le résultat important suivant concernant la structure de la forme algébrique normale des fonctions $MAJ_n$.

**Définition 30** *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 $\binom{n}{\frac{n}{2}}$ valeurs $x$ de poids $\frac{n}{2}$ sont telles que $f(x) = 1$.*

La fonction majorité est unique pour chaque $n$ impair et il y en a $\binom{n}{\frac{n}{2}}$ différentes pour chaque $n$ pair.

Les fonctions $MAJ_n$ sont connues pour être équilibrées quelle que soit la valeur $n$ de variables [38]. Sans perte de généralité, je me limiterai au cas $n$ impair. Quant $n$ est pair, $\binom{n}{\frac{n}{2}}$ fonctions sont à considérer mais la preuve reste la même, quoique un peu plus technique.

Les fonctions $MAJ_n$ sont des fonctions d'un usage très répandu. Elles ont été et sont encore très utilisées dans un grand nombre de domaines : conception de circuits logiques comme des additionneurs à $n$ bits, en tests de circuits logiques, en cryptographie (les fonctions de hachage MD5, SHA... les utilisent ; on les trouve comme fonctions de combinaison dans certains systèmes de chiffrement par flot comme l'algorithme A5 du G.S.M. ), en

théorie des codes (décodage par majorité où ces fonctions sont implémentées directement en hardware [14, Meggit decoder, pp 1581]). Elles ont été également très utilisées dans la partie communication des programmes de missiles balistiques américains [77].

Leur utilisation et implémentation a été jusqu'à présent limitée à des valeurs de $n$ relativement petites. En effet construire leur forme algébrique normale (celle utilisée en pratique) est de complexité exponentielle.

**Proposition 32** *Soit $f$ une fonction $MAJ_n$. Elle est équilibrée car sa forme algébrique normale satisfait les propriétés suivantes :*

1. $\forall \alpha \in \mathcal{A}(f),\ \forall \beta \in \mathcal{A}(f),\ supp(\alpha) \cap supp(\beta) \neq \emptyset$ (propriété d'intersectance).
2. $\forall \alpha \in \mathcal{A}(f),\ |cov(\alpha)| \equiv 1 \mod 2$ (propriété de couverture impaire).
3. $\forall \alpha \in \mathcal{A}(f), \forall \beta \in \mathcal{A}(f), |cov(\alpha \cup \beta)| \equiv 1 \mod 2$ (propriété de couverture impaire pour l'union).

*Preuve.*
Par définition de $f$ et selon la parité de $n$, nous aurons $\binom{n}{\frac{n+1}{2}}$ ($n$ impair) ou au moins $\frac{1}{2}\binom{n}{\frac{n}{2}}$ ($n$ pair) monômes. Alors en prenant tous les sous-ensembles de $k$ points de $\mathbb{N}_n^*$ tels que $k \geq d_{min}$ ($d_{min} = \frac{n+1}{2}$ ou $\frac{n}{2}$), la famille $\mathcal{F}$ de tous les sous-ensembles contenant au moins le support d'un monôme de $\mathcal{A}(f)$ est intersectante et est de taille $2^{n-1}$ (par application de la propriété d'intersectance de la FAN et par le nombre total de monômes de degré total $d_{min}$). $\forall A \in \mathcal{F}$ deux cas sont à considérer :

- $\exists \alpha \in \mathcal{A}(f)$ tel que $supp(\alpha) = A$. La propriété de couverture impaire (démontrée dans la preuve du théorème 10 assure que $A$ couvre un nombre impair de $\alpha$ et alors $f(x) = 1$ avec $supp(x) = A$.
- $A \notin \mathcal{A}(f)$. $A$ peut être obtenu par l'union des supports de certains $\alpha \in \mathcal{A}(f)$ de poids $\frac{n+1}{2}$ (ou $\frac{n}{2}$ si $n$ pair) (puisque par définition de la fonction, tous les monômes de ce degré sont présents dans la FAN) et la propriété de couverture impaire pour l'union assure alors que $f(x) = 1$ pour $supp(x) = A$.

En fait, pour le dernier cas, en toute rigueur, il faudrait prouver que pour tout $I \subseteq \mathcal{A}(f),\ |I| = i,\ \ |cov(\bigcup_{\alpha \in I})| \equiv 1 \bmod 2$ ($i$-monômes) mais en fait il suffit de le prouver pour $i = 2$. Si $I_{\alpha,\beta} = \{\lambda \in \mathcal{A}(f) | \lambda \leq \alpha \cup \beta\}$, en considérant $I_{\alpha,\beta}, I_{\alpha,\delta}, I_{\beta,\delta}$ and $I_{\alpha,\beta,\delta}$ et en supposant que $I_{\alpha,\beta}, I_{\alpha,\delta}$ et $I_{\beta,\delta}$ sont tous trois de cardinal impair (c'est-à-dire que nous considérons la propriété vraie pour $i = 2$), on montre facilement que $I_{\alpha,\beta,\delta}$ est également de cardinal

impair, par application de la formule d'inversion de Möbius, comme

$$|I_{\alpha,\beta,\delta}| = |I_{\alpha,\beta}| + |I_{\alpha,\delta}| + |I_{\beta,\delta}| - \sum_{i,j,k} |I_{i,j} \cap I_{j,k}| + |I_{\alpha,\beta} \cap I_{\alpha,\delta} \cap I_{\beta,\delta}|$$

La généralisation à tout autre $i$-monôme de $\mathcal{A}(f)$ ($i > 3$) est immediate. Par construction, on voit facilement que ces fonctions sont les fonctions $MAJ_n$ car quel que soit $x$ tel que $wt(x) \geq d_{min}$ $f(x) = 1$. D'après ce qui précède, tous les $x$ de poids supérieur ou égal à $\frac{n+1}{2}$ sont tels que $f(x) = 1$. Comme il y en a $2^{n-1}$, $f$ est bien équilibrée. ■

*Remarque :* Par application du théorème 10 et de la proposition 32, j'ai totalement défini la structure de la forme algébrique normale des fonctions $MAJ_{2^q-1}$. Elle consiste en tous les monônes de degré total $2^{q-1}$. Ce résultat est important pour tous les domaines où cette FAN doit être connue (en particulier en conception de circuits logiques). En effet, il n'est plus nécessaire de calculer leur FAN (complexité exponentielle $\mathcal{O}(2^n)$ ) pour complètement l'obtenir et connaitre leur structure.

Dans le cas des fonctions $MAJ_n$ avec $n \neq 2^q - 1$ les designs associés sont de type PBIBD (*Pairwise Block Incomplete Balanced Designs*) . Pour ce type de designs, les blocs sont d'un nombre fini de tailles différentes. Cette généralisation est en cours d'étude.

Les propriétés 2 et 3 de la proposition 32 assurent que la famille intersectante contenant $\mathcal{A}(f)$ ne réalise pas la *propriété d'union* ($F \cup F' \neq \mathbb{N}_n^*,\ \forall F \in \mathcal{F}, \forall F' \in \mathcal{F}$). Autrement, selon le théorème de Daykin-Lovàsz-Schönheim [69], on aurait eu $|\mathcal{F}| < 2^{n-2}$.

Malheureusement, les fonctions $MAJ_n$ ne sont pas de bonnes fonctions cryptographiques, sauf pour des valeurs asymptotiques de $n$.

**Proposition 33** *Les fonctions booléennes $MAJ_n$ ont un ordre de corrélation de 1 (immunes aux corrélations d'ordre 0) pour toutes les variables et*

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

*Preuve.*
En fait, en utilisant le fait que $MAJ_n(x) + MAJ_n(x + \overline{1}) = 1$ (fonctions auto-duales [38] dans le jargon de la conception de circuit ; en cryptologie et en théorie des codes, on utilisera plutôt le terme de fonctions possédant la structure linéaire $\overline{1}$ ) il est facile de prouver l'ordre de corrélation au moyen

de la transformée de Walsh mais cette technique ne donne pas directement la valeur de corrélation. De plus, le calcul de cette transformée est exponentiel en temps et mémoire. Je vais donc utiliser une autre approche, pour prouver ce résultat. Encore une fois, je me limiterai, sans perte de généralité au cas $n$ impair. Par définition, l'ordre de corrélation vaut 1 si $P[MAJ_n(x) = x_i] \neq \frac{1}{2}$. Calculons $|I| = |\{x \in \mathbb{F}_2^n | MAJ_n(x) = x_i\}|$ pour un $i$ quelconque. Deux cas différents sont à considérer :

- $x_i = 0$ et $MAJ_n(x) = 0$. Une fois $x_i$ fixé à 0, on doit choisir au plus $\frac{n-1}{2}$ "1" parmi $n-1$ positions. Alors $x$ sera de poids $< \frac{n+1}{2}$ et $MAJ_n(x) = 0$. Il y a $\sum_{j=0}^{\frac{n-1}{2}} \binom{n-1}{j}$ telles valeurs.
- $x_i = 1$ et $MAJ_n(x) = 1$. De la même manière, nous avons $\sum_{j=\frac{n-1}{2}}^{n-1} \binom{n-1}{j}$ telles valeurs.

Finalement $|I| = \sum_{j=0}^{\frac{n-1}{2}} \binom{n-1}{j} + \sum_{j=\frac{n-1}{2}}^{n-1} \binom{n-1}{j}$ c'est-à-dire $|I| = 2^{n-1} + \binom{n-1}{\frac{n-1}{2}}$. Pour parler en terme de probabilité, on normalise par $2^n$ et nous obtenons le résultat. ■

*Remarque :*

1. Il est aisé de calculer de la même manière que $P[MAJ_n(x) = \sum_i x_i] = \frac{1}{2}$ quand le nombre de $x_i$ est pair (la fonction $MAJ_n$ est corrélée à toutes les fonctions linéaires des variables d'entrée, de poids pair).
2. Quand $n \to \infty$ et avec l'approximation de Stirling de $n!$, on voit facilement que le second terme (déviation de $\frac{1}{2}$) tend vers 0. Cela signifie qu'asymptotiquement, $MAJ_n(x)$ est une bonne fonction cryptographique.

## 5.5 Fonctions booléennes et cryptanalyse

La première idée pour trouver des fonctions booléennes équilibrées et hautement non linéaires est de les tirer au hasard avec l'espoir que les propriétés requises sont réalisées. En fait la proportion de telles fonctions semble tellement faible que cette approche est illusoire.

L'approche choisie par Caroline Fontaine a été de restreindre sa recherche à un corpus de fonctions particulières : celles présentes dans les translatés du code de Reed-Müller d'ordre 1 (fonctions booléennes de degré au plus 1 dont l'ensemble sera noté $\mathcal{A}_n$), générés par des fonctions particulières appelées *idempotents*.

**Définition 31** *Pour une fonction $f$ de $\mathbb{F}_2^n$, le* translaté $\mathcal{C}_f$ *de $\mathcal{A}_n$ généré*

*par $f$ est l'ensemble $\{f + g, g \in \mathcal{A}_n\}$. $f$ est un* idempotent *si et seulement si $f^2 = f$.*

Le lecteur intéressé par une description détaillée de la méthode pourra consulter [65].

Cette recherche a donné plusieurs milliers de fonctions jusqu'à 9 variables, équilibrées et présentant une très haute non linéarité et de degré au plus 7. Mais ces fonctions n'étaient en général pas immunes aux corrélations. Ce problème a été contourné en utilisant la construction de P. Camion, C. Carlet, P. Charpin et N. Sendrier [20]. Avec une fonction $t$-résiliente à $n$ variables, soit la fonction $g$ à $n + 1$ variables définie par

$$g(x_1, x_2, \ldots, x_{n+1}) = f(x_1, x_2, \ldots, x_n) \oplus x_{n+1}$$

Alors $g$ est $t + 1$-résiliente.

En appliquant cette construction deux fois aux fonctions trouvées lors de la recherche sur les idempotents, des fonctions 2-résilientes ont ainsi été obtenues, à 9 variables. Le compromis obtenu permet de disposer de fonctions 2-résilientes, de haute non linéarité et de degré optimal pour l'ordre de résilience obtenu.

En fait d'autres constructions ont été depuis proposées [117, 118, 162]. En général l'ordre de résilience est légèrement plus grand mais au prix d'un degré plus faible. Alors qu'elles sont les meilleures fonctions, si cette question a un sens. Pour cela je vais expliquer comment l'absence de telle ou telle propriété permet telle ou telle attaque.

Je vais pour cela utiliser le schéma générique du générateur à combinaison de registres présenté dans la figure 5.2 (pour plus de détail voir section 2.3.3).

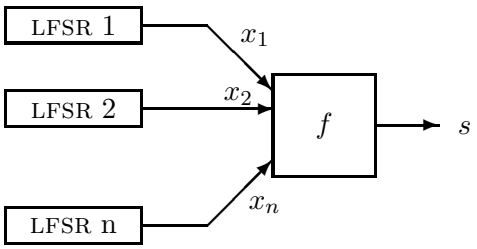


FIGURE 5.2 – Système à combinaison de registres

### Absence d'équilibre

Dans ce cas la fonction de combinaison $f$ produira plus souvent une valeur, 0 par exemple, qu'une autre. La suite $s$ est donc prédictible avec un certain biais. Lorsque combinée au clair, il est alors possible de retrouver ce dernier (en tout ou partie) avec la connaissance de ce biais et la redondance du langage. A noter que ce biais ne doit pas nécessairement être élevé pour permettre l'attaque. Précisons ici que l'attaque ne permet que de retrouver le clair et non pas (en général) la clef, c'est-à-dire l'initialisation des registres.

### Mauvaise non linéarité

Le but de ce critère est de briser, d'une manière globale, la linéarité du système. Plus la non linéarité de la fonction $f$ sera élevée plus le degré de la meilleure fonction $g$ pouvant approximer $f$ sera élevé.

Quand la non-linéarité est insuffisante certaines attaques sont alors possibles [175, 176].

### Degré de la fonction trop faible

Ce critère est en fait trop souvent négligé. La raison est qu'aucune attaque publiée n'exploite véritablement et efficacement le degré de la fonction booléenne (sauf dans le cas trivial où le degré de $f$ vaut 1 [3]).

On sait bien sûr que le degré de $f$ détermine directement la complexité linéaire de la suite $s$ (voir section 2.3.3). Il doit donc être relativement élevé. Il est clair que l'importance d'un degré élevé a été sous-estimée.

Cette affirmation résulte de constatations faite récemment lors de l'analyse statistique d'un nouveau type de système de chiffrement ultra-rapide, que j'ai développé avec Caroline Fontaine et Djessy Vianne. Avec des fonctions de degré 7, celles produites par C. Fontaine, les résultats des tests sont légérement meilleurs globalement qu'avec des fonctions de degré 6 produites par différents autres auteurs [117, 118, 162], les autres critères étant optimaux. L'étude en cours avec Caroline Fontaine tente d'expliquer cette constatation et surtout de déterminer quelle attaque est éventuellement possible quand le degré n'est pas suffisant.

3. De tels systèmes ont pourtant été proposés et bien sûr cassés (chiffrement de Hill par exemple) ! !

### Ordre d'immunité aux corrélations peu élevé

Tout d'abord il convient de signaler qu'un ordre de résilience plus élevé fait courir le risque d'avoir ponctuellement des valeurs de corrélations plus élevées en vertu de l'équation de Parseval (voir section 2.4). Cela rend certaines attaques plus faciles. C'est précisément cet écueil que rencontrent les constructions récentes. Or les fonctions obtenues par Caroline Fontaine présentent un spectre de Walsh extrêmement bien réparti, interdisant les attaques par corrélations réalistes.

Si l'ordre d'immunité aux corrélations est faible, il est aussi le plus facile à corriger par un choix judicieux de conception. Cela autorise de considérer des fonctions sous-optimales pour ce critère, sous réserve que le spectre soit aussi uniformément réparti que possible. Rappelons que dans le schéma de la figure 5.2, si l'ordre d'immunité aux corrélations est $k$, il faudra alors attaquer simultanément $k+1$ registres lors d'une attaque par corrélation (rapide ou non). On peut donc jouer sur la taille des registres en entrée pour corriger un éventuel ordre d'immunité aux corrélations insuffisant.

Pour preuve et pour conclure le système à combinaison de registres suivant mis au point avec Caroline Fontaine utilise une fonction optimale d'ordre d'immunité aux corrélations de 2.

– La fonction de combinaison est à 9 variables, de degré 6, CI(2) et optimalement non linéaire (non linéarité 224). Sa FAN est la suivante :

$$
\begin{aligned}
&x_8 \oplus x_9 \oplus x_5x_3x_2 \oplus x_7x_4x_2 \oplus x_7x_5x_4 \oplus x_1x_5x_3 \oplus x_4x_3x_2 \\
&\oplus x_7x_6x_2 \oplus x_1x_3x_2 \oplus x_7x_6x_3 \oplus x_5x_4x_2 \oplus x_7x_5x_3 \oplus x_1x_5x_2 \\
&\oplus x_6x_4x_3 \oplus x_6x_5x_2 \oplus x_1x_4x_2 \oplus x_7x_3x_2 \oplus x_1x_7x_3 \oplus x_1x_6x_4 \\
&\oplus x_7x_5x_4x_2 \oplus x_7x_6x_4x_2 \oplus x_7x_6x_3x_2 \oplus x_6x_5x_3x_2 \oplus x_7x_5x_3x_2 \\
&\oplus x_6x_4x_3x_2 \oplus x_5x_4x_3x_2 \oplus x_7x_6x_5x_4x_2 \oplus x_7x_6x_5x_4x_3x_2 \oplus \\
&x_7x_6x_5x_3x_2 \oplus x_7x_6x_4x_3x_2 \oplus x_7x_5x_4x_3x_2 \oplus x_6x_5x_4x_3x_2 \oplus x_1x_7 \\
&\oplus x_1x_5 \oplus x_1x_4 \oplus x_1x_3 \oplus x_1x_2 \oplus x_5x_2 \oplus x_3x_2 \oplus x_7x_6x_5x_4x_3 \\
&\oplus x_7x_5x_4x_3 \oplus x_6x_5x_4x_3 \oplus x_1 \oplus x_7 \oplus x_6 \oplus x_5 \oplus x_4 \oplus x_3 \\
&\oplus x_2 \oplus x_7x_6 \oplus x_7x_5 \oplus x_6x_5 \oplus x_7x_4 \oplus x_6x_4 \oplus x_5x_4 \oplus x_6x_3 \\
&\oplus x_5x_3 \oplus x_4x_3 \oplus x_1x_6x_5x_4x_2 \oplus x_1x_7x_6x_5x_2 \oplus x_1x_7x_6x_5x_4 \\
&\oplus x_1x_7x_6x_5x_3 \oplus x_1x_7x_6x_4x_3 \oplus x_1x_6x_5x_4x_3 \oplus x_1x_7x_6x_5 \oplus x_1x_6x_4x_2 \\
&\oplus x_1x_7x_4x_2 \oplus x_1x_6x_5x_4 \oplus x_1x_7x_5x_4 \oplus x_1x_7x_6x_2 \oplus x_1x_7x_6x_4 \\
&\oplus x_1x_7x_6x_3 \oplus x_1x_5x_4x_2 \oplus x_1x_6x_3x_2 \oplus x_1x_6x_4x_3 \oplus x_1x_6x_5x_2 \\
&\oplus x_1x_7x_5x_2 \oplus x_1x_7x_3x_2 \oplus x_1x_5x_4x_3 \oplus x_1x_7x_4x_3 \oplus x_1x_7x_4x_3x_2 \\
&\oplus x_1x_7x_5x_4x_2 \oplus x_1x_7x_6x_3x_2 \oplus x_1x_5x_3x_2
\end{aligned}
$$

La fonction étant 2-résiliente il faut donc considérer au minimum trois registres simultanément pour mener une attaque par corrélation. Cela va déterminer le choix des registres.

- Neuf registres ont été choisis, de polynômes rétroactifs primitifs et longueurs deux à deux premières entre elles. Chaque registre de polynômes $P_i(x)$ produit une suite qui entrera dans la fonction $f$ par la variable $x_i$.
  - $P_1(x) = x^{17} \oplus x^{15} \oplus x^{14} \oplus x^{13} \oplus x^{11} \oplus x^{10} \oplus x^9 \oplus x^8 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^2 \oplus 1$
  - $P_2(x) = x^{19} \oplus x^{18} \oplus x^{16} \oplus x^{15} \oplus x^{11} \oplus x^{10} \oplus x^5 \oplus x^3 \oplus x^2 \oplus x \oplus 1$
  - $P_3(x) = x^{23} \oplus x^{22} \oplus x^{21} \oplus x^{20} \oplus x^{17} \oplus x^{16} \oplus x^{15} \oplus x^{12} \oplus x^{10} \oplus x^8 \oplus x^7 \oplus x \oplus 1$
  - $P_4(x) = x^{29} \oplus x^{28} \oplus x^{27} \oplus x^{26} \oplus x^{24} \oplus x^{23} \oplus x^{22} \oplus x^{20} \oplus x^{19} \oplus x^{18} \oplus x^{10} \oplus x^9 \oplus x^6 \oplus x \oplus 1$
  - $P_5(x) = x^{33} \oplus x^{31} \oplus x^{29} \oplus x^{28} \oplus x^{27} \oplus x^{26} \oplus x^{25} \oplus x^{20} \oplus x^{19} \oplus x^{16} \oplus x^{13} \oplus x^8 \oplus x^7 \oplus x^6 \oplus x^3 \oplus x \oplus 1$
  - $P_6(x) = x^{37} \oplus x^{34} \oplus x^{33} \oplus x^{30} \oplus x^{29} \oplus x^{28} \oplus x^{27} \oplus x^{24} \oplus x^{23} \oplus x^{21} \oplus x^{20} \oplus x^{12} \oplus x^7 \oplus x^6 \oplus x^5 \oplus x^3 \oplus 1$
  - $P_7(x) = x^{40} \oplus x^{35} \oplus x^{29} \oplus x^{28} \oplus x^{27} \oplus x^{26} \oplus x^{25} \oplus x^{24} \oplus x^{22} \oplus x^{19} \oplus x^{17} \oplus x^{15} \oplus x^{12} \oplus x^{11} \oplus x^8 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$
  - $P_8(x) = x^{41} \oplus x^{38} \oplus x^{37} \oplus x^{35} \oplus x^{34} \oplus x^{33} \oplus x^{31} \oplus x^{30} \oplus x^{28} \oplus x^{26} \oplus x^{22} \oplus x^{21} \oplus x^{18} \oplus x^{17} \oplus x^{16} \oplus x^{15} \oplus x^{14} \oplus x^{12} \oplus x^{10} \oplus x^9 \oplus x^8 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$
  - $P_9(x) = x^{43} \oplus x^{40} \oplus x^{39} \oplus x^{38} \oplus x^{37} \oplus x^{36} \oplus x^{35} \oplus x^{32} \oplus x^{31} \oplus x^{30} \oplus x^{28} \oplus x^{27} \oplus x^{26} \oplus x^{25} \oplus x^{21} \oplus x^{19} \oplus x^{15} \oplus x^{14} \oplus x^{10} \oplus x^8 \oplus x^7 \oplus x^6 \oplus x^5 \oplus x^2 \oplus 1$

  Ainsi la clef, c'est-à-dire l'initialisation des 9 registres est de 282 bits. Une attaque par corrélation doit au minimum considérer 3 registres simultanément, nécessitant le test de $2^{101}$ initialisations (ou d'un registre de longueur 101 pour une attaque par corrélation rapide). En effet l'attaquant, du fait du spectre de $f$ devra obligatoirement considérer les registres $R_8$ et $R_9$ ainsi que le plus petit des autres registres (soit $R_1$). Mais le spectre étant quasi uniformément réparti, les corrélations présentes sont inexploitables. Enfin notons que ce système résiste à l'attaque par décimation.

### Nombre de variables trop élevé

Ce critère n'est jamais pris en compte et la littérature ne l'a jusque là jamais considéré. Pourtant il découle de l'équation 13 dite de Parseval :

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

et de l'ordre de non corrélation. Il est évident que plus l'ordre de non corrélation est élevé plus l'énergie totale de la fonction (le membre de droite de l'équation de Parseval) est concentrée sur les valeurs non nulles du spectre de Walsh de la fonction considérée.

En vertu des autres compromis, si le nombre de variables de la fonction est trop important cela peut grandement faciliter une attaque par corrélation rapide. Chaque valeur non nulle du spectre fournit, pour chaque bit produit par la fonction, une équation de parité. Si le nombre de ces valeurs non nulles est important, le nombre de bits de séquence nécessaires sera d'autant moins important.

Voyons sur l'exemple du schéma précédent, développé avec C. Fontaine, ce que cet argument devient. La fonction est à neuf variables, CI(2) et de degré 6. Son spectre de Walsh possède 127 valeurs non nulles ce qui donne pour chaque bit de séquence chiffrante :

- 35 équations vraies à 56,25 %.
- 36 équations vraies à 54,7 %.
- 28 équations vraies à 53,12 %.
- 28 équations vraies à 51,5 %.

Ce nombre est relativement important. Toutefois, le nombre d'inconnues intervenant dans ces équations est de 282.

Dans le cas d'un registre filtré de longueur 80 par exemple, en revanche, les équations ne font plus intervenir que 80 inconnues. La fonction n'est plus adaptée pour assurer un sécurité suffisante vis-à-vis des nouvelles attaques par corrélations rapides comme celle, par exemple, développée par A. Canteaut et M. Trabbia [25]. Les techniques de décodage des équations de parité sont d'autant plus optimales qu'elles disposent de nombreuses de ces équations.

Pour illustrer encore plus l'effet affaiblissant d'un trop grand nombre de variables, pour une fonction présentant par ailleurs un excellent compromis sur les autres propriétés, considérons la fonction construite par P. Sarkar et S. Maitra [117] et utilisé dans l'algorithme Lili-128 de J. Dj. Golić, E. Dawson, W. Millan et L. Simpson [157, 158]. Cet algorithme a été proposé en réponse à l'appel européen NESSIE [131] pour des primitives et solutions cryptologiques fortes.

Dans cet algorithme, une fonction à 10 variables, CI(3) de degré 5 est utilisée dans un registre filtré de longueur 89, contrôlé irrégulièrement par un autre registre de longueur 39, soit au total 128 bits de clef. Le spectre de cette fonction, mal réparti, possède 304 entrées non nulles, soit 304 équations de parité par bit de séquence produit par cette fonction. Chaque équation fait intervenir seulement 89 inconnues. Dans le détail, nous avons donc :

– 240 équations vraies à 53,125 %.
– 64 équations vraies à 51,56 %.

Une attaque par corrélation rapide sera donc plus facile.

**Mais alors comment faire... ?**

D'après ce qui précède, il est clair que la conception de machine à chiffrer ne peut se résumer à mettre bout à bout des primitives cryptographiques fortes ou optimales. Dans le cas des fonctions booléennes, dans la mesure où seul un compromis est envisageable, il n'est pas possible de les utiliser telles quelles. L'exemple de l'algorithme LILI-128 est flagrant.

La seule possibilité est de corriger les faiblesses que laissent les compromis par des choix de conception judicieux. Là seul l'art de l'ingénieur cryptologue peut parvenir à obtenir une sécurité optimale, que la théorie ne sait pas encore décrire. C'est dans cet esprit que j'ai mis au point l'algorithme COS [45], avec les fonctions de Caroline Fontaine.

# Conclusion

Cette étude a prouvé qu'il était possible de reconstruire un algorithme de chiffrement par flot du type *système à combinaison de registres à décalage*, à partir du seul chiffré intercepté. La technique que j'ai développée est opérationnelle et permet une reconstruction effective. De nombreuses optimisations sont possibles, notamment avec la parallélisation des programmes.

J'ai pu reconstruire les registres en utilisant une approche comparable à celle des attaques par corrélation rapide, à la seule différence que le travail est fait une fois pour toute et que l'information concerne le polynôme de rétroaction et non pas l'initialisation du registre. Cela permet d'utiliser des messages chiffrés provenant de clefs différentes, ce qui accroit ainsi la portée opérationnelle de cette technique.

En utilisant les connaissances sur les fonctions booléennes, et notamment celles sur leur spectre de Walsh, cette reconstruction s'étend à la fonction de combinaison, permettant de retrouver la totalité de l'algorithme.

En final, le fait de pouvoir reconstruire un algorithme par flot, prouve que la sécurité de ces systèmes ne réside pas dans le secret de l'algorithme mais bien dans la clef secrète. Cela confirme l'importance des lois de Kerckhoffs. Cette technique est applicable à tous les systèmes qui peuvent, par transformation de schéma, se ramener à un système à combinaison de registres. J'ai montré quelques exemples d'une telle équivalence.

En étudiant les registres à décalage contrôlés régulièrement, équivalents au fait de décimer la suite qu'ils produisent, j'ai mis au point une nouvelle attaque de tous les systèmes de chiffrement par flot équivalents aux systèmes à combinaison de registres : l'attaque par décimation. Elle permet de considérablement réduire la complexité de l'attaque de Siegenthaler et peut concurrencer les meilleures attaques par corrélation rapides connues pour de longs registres.

Elle exploite la propriété selon laquelle la sous-suite obtenue par une décimation régulière adéquate de la suite de sortie d'un registre, est générée par un registre plus court. J'ai alors pu déterminer un critère de résistance

à cette attaque. Si le registre est de longueur première, l'attaque se révèle inopérante. Une généralisation de cette attaque quand ce critère de résistance est vérifié est actuellement en cours d'étude et devrait permettre l'attaque par décimation dans tous les cas.

Enfin, j'ai donné une nouvelle caractérisation en termes de designs combinatoires de fonctions booléennes équilibrées. Cela m'a permis de construire une famille infinie de fonctions de classe de poids connue et surtout de caractériser complètement la forme algébrique normale des fonctions majorité. Ces fonctions sont très utilisées dans de nombreux domaines et jusqu'à présent le calcul de cette forme requièrait une complexité exponentielle.

## Deuxième partie

# Reconstruction des codes convolutifs

# Introduction

Les codes convolutifs ont été introduits en 1955 par Elias [52] comme une alternative aux codes en blocs. L'information est alors traitée non plus par morceaux (les blocs) mais par flux. Depuis, ils ont été et sont largement utilisés par les militaires et les civils dans différents systèmes de communications. Rapides, faciles à implémenter (surtout en hardware), leur principal avantage réside dans leur décodage souple, plus efficacement réalisable que pour les codes en blocs. De plus, les problèmes de synchronisation sont pour les codes convolutifs beaucoup plus simples.

Wozencraft [172] proposa une méthode efficace de décodage : le décodage séquentiel (*sequential decoding*). Massey [120] proposa ensuite une méthode moins performante mais plus simple à implémenter, le décodage par seuil (*threshold decoding*). Ces deux techniques de décodage favorisèrent alors très vite les applications dans le domaine des transmissions digitales et radios.

En 1967, Viterbi [168] proposa alors un algorithme de décodage par maximum de vraisemblance, relativement simple à programmer pour des codes de faible mémoire, appelé maintenant *décodage de Viterbi*. Cet algorithme de décodage, ainsi que des améliorations sensibles du décodage séquentiel, permirent l'utilisation intensive des codes convolutifs dans les communications spatiales et par satellites dès le début des années 70. Une liste complète des applications utilisant les codes convolutifs pourra être trouvée dans l'ouvrage de Lin et Costello [112, chap. 17].

La reconstruction des codes est un problème qui n'a fait l'objet que de rares publications. Les rares exceptions concernent les codes en blocs [137, 165]. Toute interception (légale ou illégale) produit un train binaire inexploitable en lui seul. Sans la connaissance du code utilisé pour le générer, l'information d'origine reste difficile (codage systématique) ou impossible d'accès. L'attaquant doit donc reconstituer le code à partir de la seule communication interceptée.

Dans ce contexte, le cas précis des codes convolutifs est essentiel : ce type de codage est extrêmement utilisé, notamment dans les communica-

tions gouvernementales, mais aussi dans la téléphonie mobile, domaines où le problème de l'interception est crucial et sensible.

De plus, ces codes sont principalement utilisés pour protéger un message déja chiffré. Le problème de la reconstruction de ces codes est donc fondamental. La seule étude connue donne l'esquisse d'une solution dans le cas (théorique) d'un canal sans bruit, pour des codeurs de taux $\frac{1}{n}$ [145].

Le tableau 5.1 présente quelques exemples célèbres d'applications du codage convolutif.

| Dates | Codeurs | Systèmes |
|---|---|---|
| 1968 - 1973 | (2,1,31) | Missions Pionneer 9,10,11 et 12 |
| 1969 | (2,1,6) | US Defense Satellite Communication Agency |
| 1977 | (2,1,6)<br>(3,1,6) | Mission Voyager |
| 70s | (2,1,12) | US Army PSK Modem |
| 1973 | (2,1,5) | US Naval Electronics System Commands |
| 1977 | (2,1,6) | US Defense Satellite Network DSCS |
| 80s | (2,1,6) | Mission Voyager (N.A.S.A.) |
| 80s - 90s | (2,1,4)<br>(3,1,4)<br>(6,1,4) | G.S.M. |

TABLE 5.1 – Exemples d'utilisation de codage convolutif

Je présente dans cette partie la résolution complète du problème de la reconstruction des codes convolutifs, quel que soit leur taux, pour un canal bruité ou non. Les expériences ont montré qu'un taux de bruit relativement important pouvait être envisagé.

Je rappelle dans le chapitre 6 les principes fondamentaux de la théorie des codes, pour les codes en blocs. Le chapitre 7 présentera en détail les codes convolutifs, en particulier en les comparant avec les codes en blocs. Le chapitre 8 exposera la technique de reconstruction que j'ai développée, d'abord dans le cas d'un canal sans bruit, puis d'un canal bruité. De nombreux résultats numériques seront présentés. Le chapitre 9 présentera les codes convolutifs poinçonnés et je montrerai comment étendre la technique de reconstruction du chapitre 8 à ces codes particuliers.

Les principaux résultats ont été présentés lors des conférences *Cryptography and Coding'97* [57] et *International Symposium On Information Theory and Applications 2000* [61] et sont publiés dans les actes de ces conférences.

# Chapitre 6

# Introduction à la théorie des codes

Claude E. Shannon montra en 1948 [152] que le problème de la transmission de l'information entre deux points, la source et la destination, au travers d'un canal, peut toujours se subdiviser en deux sous-problèmes :

- le codage de source (*source coding*) qui concerne la représentation optimale de l'information sous forme de bits.
- transmettre au travers d'un canal, des bits aléatoires et indépendants ou codage de canal (*channel coding*).

Un système générique de communication peut alors être schématisé par la figure 6.1. Ce principe de séparation de Shannon permet de considérer

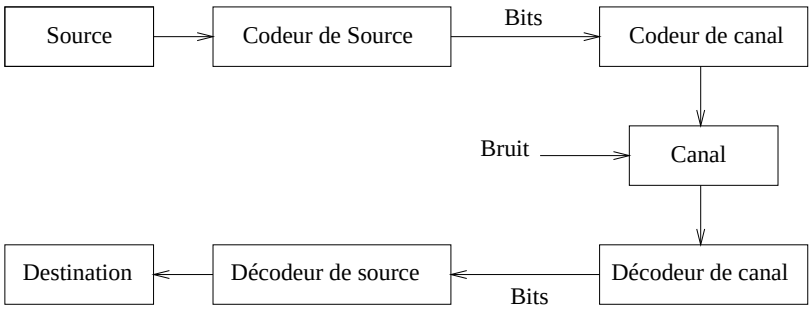


FIGURE 6.1 – Système générique de communication

séparément et indépendamment la partie canal de la partie source, un même canal pouvant être utilisé pour différentes sources.

Le célèbre théorème de codage de source de Shannon permet de caractériser tout canal par un unique paramètre, la capacité de canal $C_t$ (en bits par seconde). Toute transmission fiable de $r_t$ bits aléatoires par seconde (le taux de transmission) est possible dans le canal si et seulement si $R_t < C_t$.

En particulier, Shannon montra que le rapport signal à bruit $\frac{E_b}{N_0}$ n'avait pas d'importance significative tant que cette inégalité était respectée [1]. Seule la façon de coder l'information est importante : non pas bit à bit (informatif) mais de sorte que chacun de ces bits d'information transmis ait une influence sur un grand nombre des bits transmis, autrement dit par bout de séquence ou blocs. Cette idée radicalement novatrice a donné naissance à la *théorie des codes*.

## 6.1 Les codes en blocs

La théorie des codes est généralement présentée selon une classification qui se préoccupe plutôt des propriétés foncières des codes (la linéarité par exemple). Le lecteur intéressé trouvera dans les ouvrages de reférence [114, 166] une telle approche. Afin d'être plus près des principes fondateurs de Shannon et surtout pour renforcer le contraste avec les codes convolutifs, j'ai plutôt choisi de présenter les bases de cette théorie en considérant la forme de ces codes, et tout particuliérement les codes en blocs. Le corps de base sera $\mathbb{F}_2$ mais la théorie se généralise pour tout autre corps $\mathbb{F}_q$.

Un message est divisé en blocs de $k$ bits chacun : $\mathbf{u} = u_{k-1}, \ldots, u_1, u_0$. Un $(n, k)$ code binaire en blocs $\mathcal{C}$ est alors un ensemble de $m = 2^k$ $n$-uplets binaires $\mathbf{v} = v_{n-1}, \ldots, v_1, v_0$ appelés mots de codes. Le paramètre $n$ est appelé la longueur du code et la quantité

$$r = \frac{k}{n} \tag{6.1}$$

est dénommée taux du code et s'exprime en bits par symbole codé.

**Exemple 15** *L'ensemble* $\mathcal{C} = \{000, 011, 101, 110\}$ *est un* $(3, 2)$ *code de taux* $r = \frac{2}{3}$ *et les deux tableaux suivants constituent des codages possibles :*

L'opération de codage est une bijection de $K$, ensembles des blocs de message possibles, vers l'ensemble $\mathcal{C}$. De cette manière, il est donc possible de définir $n!$ codes différents à partir d'un même ensemble $\mathcal{C}$. Notons que le

1. En fait pour être plus précis, si le rapport $\frac{E_b}{N_0}$ est réduit de sorte que $R_t > C_t$ la transmission par codage n'est plus possible. Pour un canal de type AWGN (*Additive White Gaussian Noise*), par exemple, il faut que $\frac{E_b}{N_0} \geq -1.68\ dB$.

| $u_1u_0$ | $v_2v_1v_0$ |
|---|---|
| 00 | 000 |
| 01 | 011 |
| 10 | 101 |
| 11 | 110 |

| $u_1u_0$ | $v_2v_1v_0$ |
|---|---|
| 00 | 101 |
| 01 | 011 |
| 10 | 110 |
| 11 | 000 |

taux du code est exactement la fraction des bits du mots de code nécessaires pour représenter l'information, tandis que la fraction restante $1 - r = \frac{n-k}{n}$ représente la redondance utilisée pour détecter et/ou corriger les erreurs.

A l'autre bout du canal comment s'opère le décodage ? Considérons le cas générique du canal binaire symétrique (BSC), illustré par la figure 6.2, où un bit est modifié par le bruit en son complémentaire à 1 avec une probabilité $p$. Supposons que le bloc de message $\mathbf{u}$ soit transmis dans un BSC. Le $n$-uplet

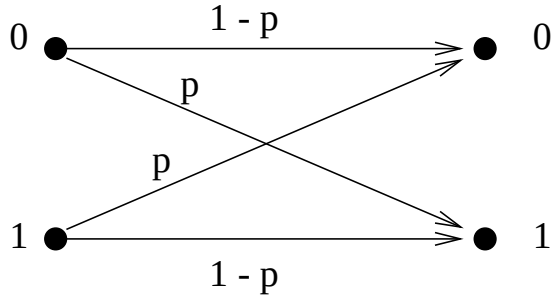


FIGURE 6.2 – Canal binaire symétrique

reçu qui a pu être modifié par le bruit sur certains bits, est décodé comme le $k$-uplet $\hat{\mathbf{u}}$. En l'absence de bruit, il est évident que $\mathbf{u} = \hat{\mathbf{u}}$ mais le bruit peut provoquer des erreurs au décodage. Comme le codage est bijectif, on peut, de façon équivalente, considérer uniquement les mots de code $\hat{\mathbf{v}}$ en sortie du canal. Si $\mathbf{v}$ a été transmis, une erreur au décodage surviendra si $\hat{\mathbf{v}} \neq \mathbf{v}$.

Soit $P_e$ la probabilité d'erreur sur un bloc, c'est-à-dire $P_e = P[\hat{\mathbf{v}} \neq \mathbf{v}]$. Alors nous avons

$$P_e = \sum_{\mathbf{r}} P[\hat{\mathbf{v}} \neq \mathbf{v}|\mathbf{r}].P[\mathbf{r}]$$

où $\mathbf{r}$ est le $n$-uplet sorti du canal avant décodage et $P[\mathbf{r}]$ est la probabilité de recevoir $\mathbf{r}$, indépendante de la règle adoptée pour le décodage.

Le but étant de minimiser $P_e$, cette règle doit être fixée de sorte que

$P[\hat{\mathbf{v}} \neq \mathbf{v}|\mathbf{r}]$ soit minimale quel que soit $\mathbf{r}$, ou encore à maximiser $P[\hat{\mathbf{v}}|\mathbf{r}] = P[\hat{\mathbf{v}} = \mathbf{v}|\mathbf{r}]$. En d'autres termes cela revient à choisir $\hat{\mathbf{v}}$ de sorte que $P[\mathbf{v}|\mathbf{r}]$ soit maximale. Cela équivaut à prendre $\hat{\mathbf{v}}$ comme le mot de code le plus probable sachant que $\mathbf{r}$ a été reçu. Utilisant le théorème de Bayes nous avons

$$P[\mathbf{v}|\mathbf{r}] = \frac{P[\mathbf{r}|\mathbf{v}].P[\mathbf{v}]}{P[\mathbf{r}]}$$

Comme les mots de code sont équiprobables[2], maximiser $P[\mathbf{v}|\mathbf{r}]$ revient à maximiser $P[\mathbf{r}|\mathbf{v}]$. Le décodeur choisissant $\hat{\mathbf{v}}$ pour maximiser $P[\mathbf{r}|\mathbf{v}]$ est appelé décodeur par *maximum de vraisemblance*.

En fait, on utilise plutôt la probabilité d'erreur sur les bits que celle sur les mots du code. En effet, un décodage incorrect sur un mot, conserve toutefois plusieurs bits corrects dans ce mot. Cela permet une meilleure mesure de la qualité du décodage et surtout d'utiliser l'outil, certainement le plus puissant de la théorie des codes, qu'est la distance de Hamming.

**Définition 32** *La distance de Hamming entre deux mots* $(x_{n-1}, \ldots, x_0)$ *et* $(y_{n-1}, \ldots, y_0)$ *de* $\mathbb{F}_2^n$ *est le nombre de positions en lesquelles ils diffèrent :*

$$d_H(x, y) = |\{i,\ 0 \leq i < n, x_i \neq y_i\}|$$

*Le poids de Hamming d'un mot* $x = (x_{n-1}, \ldots, x_1, x_0)$*, noté* $w_H(x)$*, est le nombre de ses positions non nulles.*

On montre facilement que cette distance est une métrique. Dans un canal binaire symétrique (voir figure 6.2), un bit est transmis avec erreur avec une probabilité $p$. Dans le cas du décodage par maximum de vraisemblance, $\hat{\mathbf{v}}$ est choisi pour maximiser $P[\mathbf{r}|\mathbf{v}]$. Or

$$P[\mathbf{r}|\mathbf{v}] = p^{d_H(r,v)}(1-p)^{n-d_H(r,v)} = (1-p)^n \left(\frac{p}{1-p}\right)^{d_H(r,v)}$$

Comme $0 < p < \frac{1}{2}$, on a $0 < \frac{p}{1-p} < 1$ et maximiser $P[\mathbf{r}|\mathbf{v}]$ revient à minimiser $d_H(r, v)$. Ceci montre pourquoi le décodage par maximum de vraisemblance est en fait un décodage par distance de Hamming minimale et pourquoi cette distance est un outil fondamental de la théorie des codes. On choisira $\hat{\mathbf{u}}$ correspondant au mot de code $\hat{\mathbf{v}}$, le plus proche au sens de cette distance, de la séquence reçue $\mathbf{r}$. On a alors la définition suivante, importante :

2. Si cela n'est pas le cas le décodage devrait fonctionner, peut-être de façon sous-optimale.

**Définition 33** *On appelle distance minimale $d_{min}$ d'un code $\mathcal{C}$ la valeur minimale de $d_H(\mathbf{v}, \mathbf{v}')$, pour tous les mots de code $\mathbf{v}$ et $\mathbf{v}'$ de $\mathcal{C}$, avec $\mathbf{v} \neq \mathbf{v}'$.*

A titre d'exemple, le code de l'exemple 15 a une distance minimale de 2.

Avec les notations précédentes, soit $\mathbf{e} = (e_{n-1}, \ldots, e_1, e_0)$ le motif d'erreur pour un mot de code $\mathbf{v}$. On a alors de façon évidente

$$\mathbf{r} = \mathbf{v} \oplus \mathbf{e}$$

et le nombre d'erreurs est donné par

$$w_H(\mathbf{e}) = d_H(\mathbf{r}, \mathbf{v})$$

Ceci permet de donner un des théorèmes fondateurs de la théorie des codes (le lecteur intéressé trouvera une démonstration dans [114, Chap. 1, page 10]) :

**Théorème 11** *Soit $\mathcal{C}$ un $(n, k, d)$ code. Alors $\mathcal{C}$ est $(d-1)$-détecteur d'erreurs et $t = \lfloor \frac{d-1}{2} \rfloor$-correcteur d'erreurs.*

La distance minimale d'un code est donc un paramètre fondamental car c'est elle qui précise la capacité de détection et de correction de code. Pour un code de taille et de longueur fixées, elle ne peut toutefois pas prendre n'importe qu'elle valeur : elle est limitée par la *borne de Singleton*.

**Proposition 34** *(Borne de Singleton) Soit $\mathcal{C}$ un $(n, k, d)$-code binaire. Alors*

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

*Un code atteignant cette borne est appelé code MDS (de l'anglais "Maximum Separable Distance").*

## 6.2 Les codes linéaires

Les codes linéaires sont les codes les plus utilisés et les plus étudiés. La structure linéaire imposée aux codes en blocs les rend plus faciles à décrire, à analyser, à coder et à décoder.

**Définition 34** *Un code $\mathcal{C}$ de longueur $n$ sur $\mathbb{F}_2$ est linéaire s'il constitue un sous-espace vectoriel de $\mathbb{F}_2^n$. On parle alors de sa dimension, $k$, en tant qu'espace vectoriel et on note $[n, k, d]$ ses paramètres.*

Par définition, il est possible de représenter un code linéaire de dimension $k$ par $k$ vecteurs, présentés comme les lignes d'une $k \times n$ matrice sur $\mathbb{F}_2$ : *la matrice génératrice* du code. Une matrice est dite *systématique* si elle s'écrit $(I_k|A)$ où $A$ est une matrice à $(n-k)$ colonnes et $k$ lignes.

Dans le cas des codes linéaires la distance minimale se caractérise très simplement :

**Proposition 35** *La distance minimale d d'un code linéaire est le plus petit poids de Hamming de ses mots de codes non nuls.*

Cela se démontre très simplement en utilisant la structure de groupe additif du code. L'étude de la distance minimale de ces codes revient à étudier la distribution des poids dans le code. Une description détaillée pourra être trouvée dans [114, Chap.1 page9 et Chap.5].

Le codage s'écrit alors comme une simple multiplication matricielle

$$\mathbf{v} = \mathbf{u} \cdot G$$

où

$$G = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix}$$

De la même manière, on construit la *matrice de parité* d'un $[n,k,d]$-code linéaire. C'est une matrice $H$ à $(n-k)$-lignes et $n$ colonnes vérifiant

$$\mathcal{C} = \ker(H) = \{x, x^t H = 0\}$$

Dans le cas du codage systématique, la proposition suivante explicite le lien entre matrices génératrice et de parité.

**Proposition 36** *Soit $\mathcal{C}$ un $[n,k,d]$-code binaire et $G = (I_k|A)$ sa matrice génératrice. Alors $H = ({}^tA|I_{n-k})$ est une matrice de parité de $\mathcal{C}$.*

L'algèbre linéaire classique permet d'associer à tout code linéaire son code dual noté $\mathcal{C}^\perp$ :

**Définition 35** *(Dual d'un code linéaire) Le dual $\mathcal{C}^\perp$ d'un $[n,k]$-code linéaire binaire $\mathcal{C}$ est le $[n,n-k]$-code linéaire, orthogonal de $\mathcal{C}$ dans $\mathbb{F}_2^n$ pour le produit scalaire usuel, $< x, y >= \sum_{i=0}^{n-1} x_i y_i$.*

Il est donc facile à voir que la matrice de parité d'un code $\mathcal{C}$ est la matrice génératrice de son dual.

La notion de matrice de parité prend tout son intérêt pour le décodage de ces codes qui devient alors très facile. Elle permet, grâce à l'application syndrôme, de définir une partition des mots de l'espace.

**Proposition 37** *(Syndrôme) Soit $\mathcal{C}$ un $[n, k, d]$-code binaire et $H$ une matrice de parité de $\mathcal{C}$. Le* syndrôme *d'un vecteur $x$ de $\mathbb{F}_2^n$ est le vecteur $s$ de $\mathbb{F}_2^{n-k}$ défini par*

$$s = x^t H$$

L'application qui, à tout mot de l'espace, associe son syndrôme est donc $\mathbb{F}_2$-linéaire. Elle induit une relation d'équivalence sur tout l'espace définie par

$$\begin{aligned} x\mathcal{R}y &\iff x^t H = y^t H \\ &\iff (x \oplus y) \in \mathcal{C} \end{aligned}$$

La classe d'équivalence d'un mot $x$ de $\mathbb{F}_2^n$ est appelé un *translaté* du code $\mathcal{C}$ de générateur $x$ et est noté $(x+\mathcal{C})$. Elle correspond à tous les mots de l'espace de même syndrôme $x^t H$. Le décodage par maximum de vraisemblance pour un mot ayant au plus $\lfloor \frac{d-1}{2} \rfloor$ coordonnées erronées, revient à rechercher le mot de plus petit poids dans le translaté de ce mot.

# Chapitre 7

# Les codes convolutifs

## 7.1 Introduction

Introduits par P. Elias en 1955 [52], réintroduits sous le terme de "codes récurrents" par D.W. Hagelbarger [85], les codes convolutifs sont souvent envisagés et présentés comme des codes linéaires "non blocs" sur des corps finis (on se limitera ici au cas binaire, c'est-à-dire au corps d'ordre 2). Par "non blocs", il est sous-entendu que l'information est traitée non pas par morceaux mais par flux. Mais en fait, parmi les nombreuses représentations de ces codes, on s'aperçoit que codes en blocs et codes convolutifs sont à la fois similaires et en même temps très différents.

Un $(n, k)$-code en blocs (voir chapitre précédent) peut être vu comme une bijection $\Phi$ de $\mathbb{F}_2^k$ vers $\mathbb{F}_2^n$. A une séquence de blocs messages $u(0), u(1), \ldots$ correspondra une séquence de mots de code $v(0), v(1), \ldots$ où $\Phi(u(i)) = v(i)$. Un $(n, k)$-code convolutif quant à lui peut être également décrit comme une application linéaire de l'ensemble des séquences de mots de dimension $k$ (appartenant à $\mathbb{F}_k$) vers l'ensemble des séquences de mots de dimension $n$ (appartenant à $\mathbb{F}_n$) : $\Phi : u(0), u(1), \ldots \mapsto v(0), v(1), \ldots$.

La principale différence est, que pour les codes convolutifs, le ième mot $v(i)$ est une fonction non seulement de $u(i)$ mais également d'un certain nombre de mots précédents $u(i-1), \ldots, u(i-M)$ où $M$ sera appelé mémoire du codeur. Cet aspect des choses incite quelquefois à penser que les codes en blocs sont un cas dégénéré des codes convolutifs sans mémoire, vision techniquement correcte mais trop réductrice et trompeuse.

Les codes convolutifs peuvent aussi être vus comme des codes en blocs sur des corps infinis particuliers : les corps des séries de Laurent binaires $\mathbb{F}_2((D))$ et plus particulièrement sur le sous-corps des fonctions rationnelles

binaires $\mathbb{F}_2(D)$. C'est cette vision qui permet de décrire la sortie du codeur comme le résultat de convolutions, d'où le nom donné à ces codes.

L'algorithmique et la théorie des codes convolutifs est extrêmement riche et la littérature la concernant est abondante. Le but de ce chapitre est de présenter ces codes d'une manière simple et concise. Après quelques exemples introductifs, je présenterai les codes convolutifs plus formellement et notamment à partir des codes en blocs (vision enpruntée à R.J. McEliece [115, Chap. 12]) et donnerai leurs principales propriétés. La notion de matrice canonique sera ensuite présentée. Elle permettra de comprendre pourquoi, lors de la reconstruction d'un codeur, plusieurs solutions sont obtenues. Le problème du décodage de ces codes sera finalement présenté. Une présentation exhaustive des codes convolutifs pourra être trouvée dans [98, 115].

## 7.2 Exemples introductifs

Considérons un exemple de code convolutif binaire de taux $\frac{1}{2}$ schématisé par la figure 7.1. Le message $\mathbf{u}$ est une séquence de bits $\mathbf{u} = u_0, u_1, \ldots$, qui

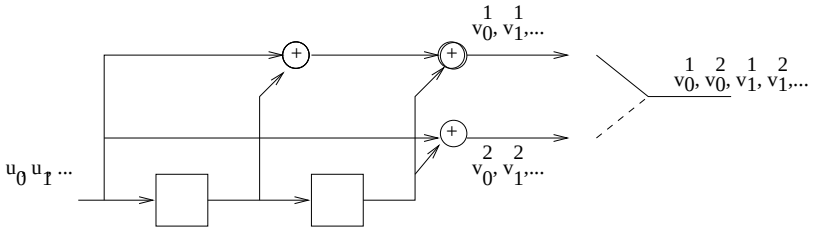


FIGURE 7.1 – Codeur convolutif de taux $\frac{1}{2}$

entrent et circulent par décalages successifs dans un registre, ici de longueur (ou mémoire) $M = 2$. Ce codeur possède deux fonctions linéaires produisant chacune une séquence de sortie. Chaque fonction, au temps $t$, calcule et produit un bit en fonction du contenu du registre, la séquence $\mathbf{u}$ est décalée, et ainsi de suite...

Les deux séquences de sortie $v_0^{(1)}, v_1^{(1)}, \ldots$, et $v_0^{(2)}, v_1^{(2)}, \ldots$, sont entrelacées (augmentation de la résistance au bruit) et la suite finale $\mathbf{v} = v_0^{(1)}, v_0^{(2)}, v_1^{(1)}, v_1^{(2)}, \ldots$, est transmise. Pour un bit de message, deux bits sont finalement produits. Le taux du code est donc bien de $\frac{1}{2}$.

En pratique (pour correspondre à la définition formelle par des séries

causales de Laurent (voir la section 7.3)) le contenu initial du registre est à zéro ($t = 0$). Cela permet de décrire les séquences de sortie comme une convolution de la suite message $\mathbf{u}$ avec les deux séquences **1110000**... et **1010000**..., correspondant aux deux fonctions linéaires $1 \oplus x \oplus x^2$ et $1 \oplus x^2$. Plus précisément, il est possible de décrire l'opération de codage par de simples multiplications polynomiales si l'on utilise cette représentation [1] pour le message $\mathbf{u}$. Ainsi :

$$\begin{aligned} v^{(1)}(x) &= u(x)(1 \oplus x \oplus x^2) \\ v^{(2)}(x) &= u(x)(1 \oplus x^2) \end{aligned} \tag{7.1}$$

J'utiliserai cette représentation pour décrire la technique de reconstruction de ces codes après l'avoir formellement définie dans la section suivante.

Dans le cas général d'un code de taux $\frac{k}{n}$ ($k \leq n$) binaire, le message

$$\mathbf{u} = u_0, u_1, \ldots = u_0^{(1)}, u_0^{(2)}, \ldots, u_0^{(k)}, u_1^{(1)}, \ldots, u_1^{(k)}, \ldots$$

sera codé en la séquence

$$\mathbf{v} = v_0, v_1, \ldots = v_0^{(1)}, v_0^{(2)}, \ldots, v_0^{(n)}, v_1^{(1)}, \ldots, v_1^{(n)}, \ldots$$

où $v_t = f(u_t, u_{t-1}, \ldots, u_{t-M})$ ($M$ est la mémoire du codeur). La fonction $f$ est une fonction linéaire de $\mathbb{F}_2^{(M+1)k}$ vers $\mathbb{F}_2^n$ et est couramment représentée sous forme matricielle :

$$v_t = u_t G_0 \oplus u_{t-1} G_1 \oplus \ldots \oplus u_{t-M} G_M \tag{7.2}$$

où les $G_i$ ($0 \leq i \leq M$) sont des $k \times n$ matrices binaires. On a donc

$$v_0 v_1 \ldots = (u_0 u_1 \ldots) G \quad (v_i \in \mathbb{F}_2^n, u_i \in \mathbb{F}_2^k)$$

ou encore

$$\mathbf{v} = \mathbf{u} \cdot G$$

avec

$$G = \begin{pmatrix} G_0 & G_1 & \cdots & G_M & & \\ & G_0 & G_1 & \cdots & G_M & \\ & & \ddots & \ddots & & \ddots \end{pmatrix}$$

$G$ est la matrice génératrice du code et les $G_i$, les sous-matrices génératrices.

1. Les polynômes étant un sous-ensemble du corps des séries de Laurent $\mathbb{F}_2((D))$

**Exemple 16** *Pour le codeur représenté par le schéma 7.1, la matrice $G$ et les matrices $G_i$ correspondantes sont les suivantes :*

$$G_0 = (1\ 1)\ G_1 = (1\ 0)\ G_2 = (1\ 1) \qquad G = \begin{pmatrix} 11 & 10 & 11 & & \\ & 11 & 10 & 11 & \\ & & \ddots & \ddots & \ddots \end{pmatrix}$$

**Exemple 17** *Soit un codeur convolutif de taux $\frac{2}{3}$ représenté par le schéma 7.2. La matrice $G$ et les matrices $G_i$ correspondantes sont les suivantes :*

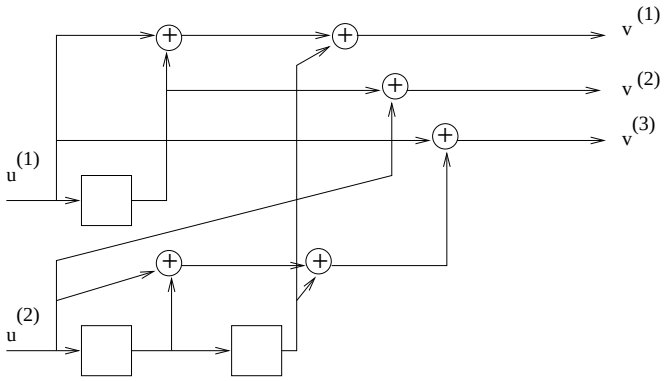


FIGURE 7.2 – Codeur convolutif de taux $\frac{2}{3}$

$$G_0 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} G_1 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} G_2 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$$G = \begin{pmatrix} 101 & 110 & 000 & & \\ 011 & 001 & 101 & & \\ & 101 & 110 & 000 & \\ & 011 & 001 & 101 & \\ & & \ddots & \ddots & \ddots \end{pmatrix}$$

## 7.3 Représentation formelle

### 7.3.1 Des codes en blocs aux codes convolutifs

En les présentant un peu différemment, les codes convolutifs peuvent en être vus comme une sorte de généralisation des codes en blocs. Soit $\mathcal{C}$ un

$[n, k]$ code en blocs binaire, de matrice génératrice $G$. Le codage est donc décrit par

$$v_t = u_t G \quad t = 0, 1, \ldots \tag{7.3}$$

où $u_t \in \mathbb{F}_2^k$ et $v_t \in \mathbb{F}_2^n$ décrivent respectivement les blocs de message et de code au temps $t$. Généralisons le code $\mathcal{C}$ en le redéfinissant comme l'ensemble de toutes les séquences possibles de mots de code $(v_0, v_1, \ldots)$ produites par l'équation de codage (7.3). Cette équation peut être exprimée plus concisément en utilisant les fonctions génératrices [2]. En utilisant l'indéterminée $x$, en multipliant les deux membres de (7.3) par $x^t$ et en sommant sur tous les $t \geq 0$ on obtient :

$$\bigoplus_{t \geq 0} v_t x^t = \left( \bigoplus_{t \geq 0} u_t x^t \right) G \tag{7.4}$$

Si on note $U(x) = \bigoplus_{t \geq 0} u_t x^t$ et $V(x) = \bigoplus_{t \geq 0} v_t x^t$ alors (7.4) devient

$$V(x) = U(x) G \tag{7.5}$$

L'équation (7.3) décrit l'action du codeur sur un seul bloc tandis que (7.5) décrit son action sur une famille infinie de séquences.

Dans (7.3), $v_t$ et $u_t$ sont des éléments des espaces vectoriels $\mathbb{F}_2^n$ et $\mathbb{F}_2^k$ alors que $U(x)$ et $V(x)$ sont des objets plus complexes (de dimension respectivement $n$ et $k$) : les séries formelles de Laurent de la forme $A(x) = \sum_{i \geq 0} a_i x^i$ avec $a_i \in \mathbb{F}_2$ encore appelées séries causales.

L'ensemble de ces séries formelles constituent un anneau commutatif noté $\mathbb{F}_2[[X]]$ qui peut être étendu naturellement en un corps noté $\mathbb{F}_2((X))$ constitué des séries formelles de Laurent suivantes :

$$A(x) = \bigoplus_{i \geq i_0} a_i x^i$$

où $i_0$ est un entier relatif quelconque. Les polynômes sont alors des séries formelles de Laurent de la forme $A(x) = a_0 \oplus \ldots \oplus a_n x^n$ et forment un ensemble noté $\mathbb{F}_2[X]$.

Enfin nous noterons $\mathbb{F}_2(X)$ le sous-corps de $\mathbb{F}_2((X))$, l'ensemble de toutes les séries de Laurent rationnelles (série de Laurent, unique expansion d'une

2. On préférera la notation en $x$ de l'indéterminée plutôt que celle, $D$, généralement utilisée pour exprimer l'opérateur "délai". Ce changement de notation permet de mieux comprendre la notation polynomiale utilisée dans le chapitre 8, traitant de la reconstruction de ces codes.

fraction rationnelle $\frac{P(X)}{Q(X)}$ où $P(X)$ et $Q(X) \neq 0$ sont des polynômes). C'est le plus petit sous-corps de $\mathbb{F}_2((X))$ contenant $X$ et $\mathbb{F}_2$. Le lecteur intéressé trouvera un exposé complet des propriétés algébriques de $\mathbb{F}_q((X))$ pour un corps fini quelconque $\mathbb{F}_q$ dans [2].

On peut alors définir un code en bloc comme l'ensemble des fonctions génératrices $V(X)$ de (7.5). Le passage de $\mathbb{F}_2[[X]]^k$ à $\mathbb{F}_2((X))^k$ pour décrire les séquences $U(X)$ (signifiant simplement le fait que le codeur peut être considéré à un instant $t$ quelconque et pas simplement 0), permet avec (7.5) de dire que $\mathcal{C}$ est le $\mathbb{F}_2((X))$-espace vectoriel de dimension $k$ engendré par les lignes de la matrice $G$, sous-espace de $\mathbb{F}_2((X))^n$. Ainsi un $[n,k]$-code linéaire en blocs peut recevoir la définition suivante, peu habituelle :

**Définition 36** *Un $[n,k]$-code linéaire en blocs sur $\mathbb{F}_2$ est un sous-espace vectoriel de $\mathbb{F}_2((X))^n$ de dimension $k$ dont une base est formée de vecteurs de $\mathbb{F}_2^n$.*

Soit à présent un $(n,k)$-code convolutif. C'est une "application linéaire" qui à une séquence de mots de $k$ bits $u_0, u_1, \ldots$ fait correspondre une séquence de mots de $n$ bits $v_0, v_1, \ldots$. Cela correspond d'une certaine façon à un code en bloc. La différence est la présence d'une mémoire interne au codeur de taille $m$, encore appelé vecteur d'état, $s_t$, à chaque instant $t$. Le $t$-ième mot de code $v_t$ sera une fonction linéaire non seulement de $u_t$ mais aussi de $s_t$. Il est alors possible de décrire formellement les choses ainsi [3] :

$$\begin{aligned} s_0 &= 0 \\ s_{t+1} &= s_t\mathcal{A} \oplus u_t\mathcal{B} \;\; t \geq 0 \qquad (7.6) \\ v_t &= s_t\mathcal{M} \oplus u_t\mathcal{N} \qquad (7.7) \end{aligned}$$

où les quatre matrices $\mathcal{A}, \mathcal{B}, \mathcal{M}$ et $\mathcal{N}$ sont binaires et respectivement de taille $M \times M, k \times n, M \times n$ et $k \times n$. L'entier $M$ est appelé le *degré* du codeur. Alors qu'un $[n,k]$-code en bloc est caractérisé par une unique matrice génératrice $G$ de taille $k \times n$, un $(n,k)$-code convolutif sera complètement caractérisé par les quatre matrices binaires $(\mathcal{A}, \mathcal{B}, \mathcal{M}, \mathcal{N})$ précédentes.

Si on supprime la mémoire du codeur, c'est-à-dire si $M = 0$, alors les matrices $\mathcal{A}, \mathcal{B}$ et $\mathcal{M}$ disparaissent et les équations de codage (7.6) et (7.7) se réduisent à l'équation de codage (7.3). Cela permet (même si cette vision est vraiment réductrice) de voir un code en bloc comme un code convolutif sans mémoire, ou encore de degré nul.

3. Ces équations sont appelées *équations de description de l'espace état* et ont été introduites pour la première fois par J.L. Massey en 1967 [121].

Le code associé à ces quatre matrices est donc l'ensemble de toutes les séquences de sortie $v_0, v_1, \ldots$ possibles, produites par les équations (7.6) et (7.7). On retrouvera la forme précédemment utilisée pour les codes en blocs, avec les séries de Laurent, en définissant le code convolutif, de manière équivalente, comme l'ensemble des séries causales de Laurent de forme :

$$V(X) = u_0 \oplus u_1 X \oplus u_2 X^2 \oplus \ldots$$

L'utilisation de quatre matrices, si elle permet de mieux comprendre le mécanisme du codage convolutif, n'est pas pratique. Il serait mieux de disposer, comme pour les codes en blocs, d'une seule matrice génératrice. Pour cela, utilisons une nouvelle fois, les fonctions génératrices. Multiplions les deux membres des équations (7.6) et (7.7) par $X^t$ et sommons pour tous les $t$. Rappelons que nous avons $u_t, v_t$ et $s_t$ nuls pour $t < 0$. Nous obtenons :

$$\begin{aligned} S(X)X^{-1} &= S(X)\mathcal{A} \oplus U(X)\mathcal{B} && (7.8)\\ V(X) &= S(X)\mathcal{M} \oplus U(X)\mathcal{N} && (7.9) \end{aligned}$$

Les séries $U(X)$ et $V(X)$ sont définies comme précédemment. $S(X)$ est définie par

$$S(X) = \bigoplus_{t \geq 0} s_t X^t.$$

Les équations (7.8) et (7.9) donnent une expression explicite de $S(X)$ et de $V(X)$ en fonction de $U(X)$ :

$$\begin{aligned} S(X) &= U(X)E(X) && (7.10)\\ V(X) &= U(X)G(X) && (7.11) \end{aligned}$$

où $E(X)$ et $G(X)$ sont des matrices, respectivement de taille $k \times M$ et $k \times n$, données par :

$$\begin{aligned} E(X) &= \mathcal{B}(X^{-1} I_M \oplus \mathcal{A})^{-1} && (7.12)\\ G(X) &= \mathcal{N} \oplus E(X)\mathcal{M} = \mathcal{N} \oplus \mathcal{B}(X^{-1} I_M \oplus \mathcal{A})^{-1}\mathcal{M} && (7.13) \end{aligned}$$

où $I_M$ désigne la matrice identité d'ordre $M$.

Comme pour les codes en blocs, si l'espace message, c'est-à-dire l'ensemble de toutes les séries possibles d'entrée $U(X)$, est étendu de $\mathbb{F}_2[[X]]^k$ à $\mathbb{F}_2((X))^k$, il s'ensuit que le code convolutif associé aux matrices $(\mathcal{A}, \mathcal{B}, \mathcal{M}, \mathcal{N})$ est un $\mathbb{F}_2((X))$-espace vectoriel engendré par les lignes de $G(X)$ (notons qu'il s'agit là d'une matrice polynomiale, d'où la notation ; cet aspect sera discuté plus longuement dans la section 7.4). $G(X)$ est donc bien la matrice génératrice de ce code convolutif.

La matrice $G(X)$, avec (7.13), est à entrées rationnelles en $X$. Comme $\mathbb{F}_2(X)$ est un sous-corps de $\mathbb{F}_2((X))$, avec (7.11), tout $(n,k)$-code convolutif est un sous-espace vectoriel de dimension $k$ de $\mathbb{F}_2((X))^n$, dont une base est constituée de vecteurs de $\mathbb{F}_2(X)^n$.

Réciproquement si $G(X)$ est une $k \times n$ matrice quelconque sur $\mathbb{F}_2(X)$, elle est, à des $\mathbb{F}_2(X)$-combinaisons linéaires de ses lignes près, équivalente à une matrice causale $G'(X)$ (*i.e.* sur $\mathbb{F}_2[[X]]$).

Par résolution de systèmes d'équations linéaires, il est évident que $G'(X)$ peut-être obtenue à partir de quatre matrices $(\mathcal{A}, \mathcal{B}, \mathcal{M}, \mathcal{N})$, d'où le $\mathbb{F}_2((X))$-espace vectoriel généré par les lignes de $G(X)$ est le code convolutif engendré par ces quatre matrices. Et l'ensemble des $(n,k)$-codes convolutifs est identique à l'ensemble des sous-espaces de dimension $k$ de $\mathbb{F}_2((X))^n$ dont les bases sont des parties de $\mathbb{F}_2(X)^n$. D'où la définition :

**Définition 37** *Un $(n,k)$-code convolutif est un sous-espace vectoriel de dimension $k$ de $\mathbb{F}_2((X))^n$ dont les bases sont des parties de $\mathbb{F}_2(X)^n$.*

Les vecteurs de base étant pris dans $\mathbb{F}_2(X)^n$, on peut en fait considérer le sous-code rationnel, c'est-à-dire des entrées matricielles prises dans $\mathbb{F}_2(X)$. C'est ce sous-code qui est généralement considéré dans les applications (voir [115, p. 1073]. D'où la définition plus générale suivante :

**Définition 38** *Un $(n,k)$-code convolutif est un sous-espace vectoriel de dimension $k$ de $\mathbb{F}_2(X)^n$.*

Comme pour les codes en blocs, le paramètre le plus important d'un code convolutif est celui qui mesure la capacité de correction, ici appelé la *distance libre*. C'est l'analogue de la distance minimale et il joue le même rôle pour déterminer les performances du code convolutif considéré lors du décodage. Plus la distance libre est élevée, meilleur est le code.

**Définition 39** *La distance libre d'un $(n,k)$-code convolutif est le poids minimum de tout mot de code non nul $V(X) = (v_0, v_1, \ldots, v_n)$. C'est encore la distance de Hamming minimum entre deux mots de code distincts :*

$$d_{libre} = \min_{V(X) \neq V'(X)} \{d_H(V(X), V'(X))\}.$$

Comme pour la distance minimale, la distance libre ne peut être déterminée *a priori*. Elle nécessite une étude spécifique pour chaque code. Nous verrons dans la section 7.5 le rôle de ce paramètre dans le décodage.

### 7.3.2 Exemple détaillé

Afin de mieux comprendre la présentation formelle du chapitre précédent, nous allons détailler l'exemple 16 de la section 7.2. Les paramètres sont les suivants : $n = 2, k = 1$ et $M = 2$. Les entrées seront notées par $u_t$, les sorties par $(v_t^1, v_t^2)$ et les états du registre par $(s_t^1, s_t^2)$. Les quatre matrices $(\mathcal{A}, \mathcal{B}, \mathcal{M}, \mathcal{N})$ sont alors données par :

$$\mathcal{A} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad \mathcal{M} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad \mathcal{B} = (1\ 0) \quad \mathcal{N} = (1\ 1). \tag{7.14}$$

Les équations (7.13) et (7.13) donnent alors :

$$\begin{aligned} E(X) &= \mathcal{B}(X^{-1} I_M \oplus \mathcal{A})^{-1} = (1\ 0) \begin{pmatrix} X^{-1} & 1 \\ 0 & X^{-1} \end{pmatrix} = (X\ X^2) \\ G(X) &= \mathcal{N} \oplus E(X)\mathcal{M} = (1\ 1) \oplus (X\ X^2) \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}. \end{aligned}$$

En final, la matrice génératrice de ce code est

$$G(X) = (1 \oplus X \oplus X^2 \quad 1 \oplus X^2). \tag{7.15}$$

Les polynômes définis en (7.2) sont bien retrouvés.

Comme $G(X)$ est *une* matrice génératrice de ce code, toute autre matrice équivalente (obtenu par $\mathbb{F}_2(X)$-combinaisons linéaires des lignes de $G(X)$) est aussi une matrice génératrice de ce code. Si, par exemple, on divise les entrées de $G(X)$ par $1 \oplus X \oplus X^2$ on obtient :

$$G'(X) = \left(1 \quad \frac{1 \oplus X^2}{1 \oplus X \oplus X^2}\right). \tag{7.16}$$

Cette matrice, équivalente à $G(X)$ est la matrice systématique du code. Les quatre matrices $(\mathcal{A}, \mathcal{B}, \mathcal{M}, \mathcal{N})$ correspondantes sont données par

$$\mathcal{A} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \quad \mathcal{M} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad \mathcal{B} = (1\ 0) \quad \mathcal{N} = (1\ 1). \tag{7.17}$$

Si nous multiplions la matrice $G(X)$ par $(1 \oplus X)$ nous obtenons la matrice, équivalente :

$$G''(X) = (1 \oplus X^3 \quad 1 \oplus X \oplus X^2 \oplus X^3) \tag{7.18}$$

Pour cette dernière, nous obtenons une matrice génératrice dite *catastrophique* de ce code.

**Définition 40** *Une matrice génératrice d'un code convolutif est dite* catastrophique *s'il existe une séquence d'entrée $U(X)$ avec un nombre fini de bits nuls ($w_H(U(X)) = \infty$) produisant des mots de code $V(X)$ avec un nombre fini de bits non nuls ($w_H(V(X)) < \infty$).*

Quand une telle matrice est utilisée, cela signifie qu'un petit nombre d'erreurs sur le canal produira un grand nombre d'erreur au décodage, d'où le terme catastrophique. Cette propriété dépendant de la matrice et non du code, tout code convolutif possède à la fois des matrices catastrophique et non catastrophique. Cette propriété a été largement étudiée dans [133] pour les codes convolutifs poinçonnés.

On voit que le choix, pour un code donné, de la matrice génératrice, est fondamental. Il conditionne les performances au décodage. Pour la matrice $G''(X)$ on voit que $M = 3$ ce qui est confirmé par ses quatre matrices d'état :

$$\mathcal{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad \mathcal{M} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \quad \mathcal{B} = (1\,0\,0) \quad \mathcal{N} = (1\,1). \quad (7.19)$$

Les trois matrices $G(X), G'(X)$ et $G''(X)$ sont génératrices du même code convolutif. La distance libre $d_{\text{libre}}$ de ce code vaut 5. En effet, la matrice de parité $H(X)$ correspondant à la matrice $G(X)$ est

$$H(X) = (1 \oplus X^2 \quad 1 \oplus X \oplus X^2)$$

Il est alors évident que les mots de codes

$$V(X) = (v_t^1, v_t^2)$$

(tels que $V(X)^t H(X) = 0$) sont de la forme $(X^L(1 \oplus X \oplus X^2)\ X^L(1 \oplus X^2))$. En appliquant la définition 39, la distance libre est bien de 5.

## 7.4 Matrice canonique d'un code convolutif

Nous venons de voir que le choix de la matrice génératrice est fondamental. Il est donc important de disposer de critères permettant de définir et de sélectionner la meilleure matrice possible. L'objet de cette section est de présenter succinctement une telle matrice, appelée *matrice canonique*. Cela permettra de choisir la meilleure des solutions fournies par l'algorithme de reconstruction.

Soit un $(n, k)$-code convolutif $\mathcal{C}$ de matrice génératrice $G(X)$. Si les entrées de $G(X)$ sont des polynômes, la matrice sera dite *matrice génératrice*

*polynomiale* (MGP). On montre facilement que tout code convolutif $\mathcal{C}$ possède une telle matrice.

Prenons une MGP du code $\mathcal{C}$ notée sous la forme $G(X) = (g_{ij}(X))$. C'est une matrice $k \times n$. On notera la ième ligne de $G(X)$ (le vecteur $(g_{i1}(X), \ldots, g_{i,n}(X))$) par $\mathbf{g}_i(X)$ et on définit le degré de $\mathbf{g}_i(X)$ comme le degré maximum de ses composants.

**Définition 41** *Avec les notations précédentes, on appelle* degré interne, *noté intdeg($G(X)$), le degré maximum des mineurs d'ordre $k$ de $G(X)$. On appelle* degré externe, *noté extdeg($G(X)$), la somme des degrés des $\mathbf{g}_i(X)$ pour $1 \leq i \leq k$.*

**Définition 42** *Une matrice polynomiale $k \times n$ $G(X)$ est dite* basique *si parmi toutes les matrices polynomiales de la forme $T(X)G(X)$, où $T(X)$ est une matrice d'ordre $k$ inversible sur $\mathbb{F}_2(X)$, elle est de plus petit degré interne possible.*

*Elle sera dite* réduite *si, parmi toutes les matrices de la forme $T(X)G(X)$, où $T(X)$ est une matrice inversible d'ordre $k$, elle possède le plus petit degré externe possible.*

De façon équivalente, une matrice polynomiale est réduite si on ne peut diminuer son degré externe par une suite d'opérations élémentaires sur ses lignes. Les deux théorèmes fondamentaux suivants permettent alors de définir des critères précis pour caractériser ces matrices. Les démonstations, complexes et longues, ne seront pas données ici. Le lecteur les trouvera dans [115, p. 1124].

**Théorème 12** *Une $k \times n$ matrice polynomiale $G(X)$ est basique si et seulement si une des six conditions suivantes est satisfaite :*

1. *Les facteurs invariants*[4] *de $G(X)$ sont tous égaux à 1.*
2. *Le pgcd des mineurs d'ordre $k$ de $G(X)$ vaut 1.*
3. *$G(\alpha)$ est de rang $k$ pour tout $\alpha \in \mathbb{F}_2$.*
4. *$G(X)$ possède un $\mathbb{F}_2[X]$-inverse à droite, c'est-à-dire qu'il existe un $n \times k$ matrice polynomiale $J(X)$ telle que $G(X)J(X) = I_k$.*
5. *Si $V(X) = U(X)G(X)$ et si $V(X) \in \mathbb{F}_2[X]^n$, alors $U(X) \in \mathbb{F}_2[X]^k$. (Sortie polynomiale implique entrée polynomiale).*

4. On appelle facteurs invariants d'ordre $i$ de la matrice $G$, la valeur $\gamma_i = \frac{\Delta_i}{\Delta_{i-1}}$ où $\Delta_i$ est le pgcd de tous les mineurs d'ordre $i$ de $G$ avec $\Delta_0 = 1$

6. *$G(X)$ est une sous-matrice d'une matrice inversible, c'est-à-dire qu'il existe une $(n-k)\times n$ matrice $L(X)$ telle que le déterminant de la $n\times n$ matrice $\begin{pmatrix} G(X) \\ L(X) \end{pmatrix}$ est non nul.*

**Théorème 13** *Une $k\times n$ matrice polynomiale $G(X)$ est réduite si et seulement si une des trois conditions suivantes est satisfaite :*

1. *Si on définit $\bar{G}$ (indicateur matriciel du terme de plus haut degré pour chaque ligne) par*
$$\bar{G}_{ij} = coeff_{X^{e_i}}(g_{ij}(X))$$
*où $e_i$ est le degré de la ième ligne de $G(X)$, alors $\bar{G}$ est de rang $k$.*
2. *$extdeg(G(X)) = intdeg(G(X))$.*
3. *(*Propriété de degré prévisible*)*
*Pour tout vecteur $U(X) = (u_1(X), \ldots, u_k(X)) \in \mathbb{F}_2[X]^k$ :*
$$deg(U(X)G(X)) = \max_{1\leq i\leq k}(deg(u_i(X)) + deg(\mathbf{g}_i(X)))$$

**Exemple 18** *Voici six matrices génératrices d'un même $(4,2)$-code convolutif binaire. Seules les matrices de $G_2$ à $G_7$ sont des MGP.*

$$\begin{aligned}
G_1 &= \begin{pmatrix} \frac{1}{1\oplus X\oplus X^2} & 1 & \frac{1\oplus X^2}{1\oplus X\oplus X^2} & \frac{1\oplus X}{1\oplus X\oplus X^2} \\ 1 & \frac{1\oplus X\oplus X^2}{X} & X & \frac{1}{X} \end{pmatrix} \\
G_2 &= \begin{pmatrix} 1 & 1\oplus X\oplus X^2 & 1\oplus X^2 & 1\oplus X \\ X & 1\oplus X\oplus X^2 & X^2 & 1 \end{pmatrix} \\
G_3 &= \begin{pmatrix} 1 & 1\oplus X\oplus X^2 & 1\oplus X^2 & 1\oplus X \\ 0 & 1\oplus X & X & 1 \end{pmatrix} \\
G_4 &= \begin{pmatrix} 1 & X & 1\oplus X & 0 \\ 0 & 1\oplus X & X & 1 \end{pmatrix} \\
G_5 &= \begin{pmatrix} 1\oplus X & 0 & 1 & X \\ X & 1\oplus X\oplus X^2 & X^2 & 1 \end{pmatrix} \\
G_6 &= \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1\oplus X & X^2 & 1 \end{pmatrix} \\
G_7 &= \begin{pmatrix} 1\oplus X & 0 & 1 & X \\ 1 & X & 1\oplus X & 0 \end{pmatrix} \\
G_8 &= \begin{pmatrix} 1 & 0 & \frac{1}{1\oplus X} & \frac{X}{1\oplus X} \\ 0 & 1 & \frac{X}{1\oplus X} & \frac{1}{1\oplus X} \end{pmatrix}
\end{aligned}$$

*Les critères des théorèmes 12 et 13 pour ces huit matrices sont résumés dans le tableau 7.1*

| Matrice | Basique (Th. 12) | Réduite (Th. 13) | intdeg | extdeg |
|---|---|---|---|---|
| $G_1$ | - | - | - | - |
| $G_2$ | non | non | 3 | 4 |
| $G_3$ | oui | non | 1 | 3 |
| $G_4$ | oui | non | 1 | 2 |
| $G_5$ | non | oui | 3 | 3 |
| $G_6$ | oui | oui | 1 | 1 |
| $G_7$ | non | oui | 2 | 2 |
| $G_8$ | - | - | - | - |

TABLE 7.1 – Propriétés de matrices génératrices d'un $(4, 2)$-code convolutif

De la définition 42, il s'ensuit que parmi toutes les MPG d'un même code convolutif, celles possédant le plus petit degré interne sont précisément les matrices basiques. L'ensemble le plus intéressant toutefois est celui des matrices dont le degré externe est le plus petit possible : ce sont les matrices *canoniques*.

**Définition 43** *Parmi toutes les MPG d'un code convolutif $\mathcal{C}$, celles ayant le plus petit degré externe possible sont appelées matrices canoniques du code. Ce degré minimal externe est appelé* degré *du code $\mathcal{C}$ et noté deg($\mathcal{C}$).*

Les matrices canoniques ont des propriétés intéressantes essentiellement déterminées par le théorème suivant (pour une démonstration voir [115, p. 1080]).

**Théorème 14** *Une matrice génératrice polynomiale d'un code convolutif est canonique si et seulement si elle est à la fois réduite et basique.*

Les deux corollaires suivants précisent certaines de ces propiétés.

**Corollaire 5** *Le degré minimal interne de toute MGP d'un code convolutif donné $\mathcal{C}$ est égal au degré de $\mathcal{C}$.*

**Corollaire 6** *Si $G(X)$ est une matrice génératrice basique quelconque de $\mathcal{C}$, alors intdeg$(G(X))$ = deg$(\mathcal{C})$.*

En fait un code convolutif possède plusieurs matrices canoniques différentes mais le théorème suivant montre qu'elles partagent beaucoup de caractéristiques identiques, ce qui précisément en fait leur intérêt.

**Théorème 15** *Si $e_i \leq e_2 \leq \ldots \leq e_k$ désignent les degrés des lignes d'une matrice génératrice canonique $G(X)$ d'un code $\mathcal{C}$ et si $f_1 \leq f_2 \leq \ldots \leq f_k$ désignent les degrés des lignes de toute autre matrice génératrice polynomiale $G'(X)$ de $\mathcal{C}$, alors $e_i \leq f_i \; \forall i \in \{1, 2, \ldots, k\}$.*

**Corollaire 7** *L'ensemble des degrés des lignes est le même pour toutes les matrices génératrices canoniques d'un même code $\mathcal{C}$.*

Les degrés de lignes de matrice $e_1 \leq e_2 \leq \ldots \leq e_k$ sont appelés les *indices de Forney*. La somme $e_1 + e_2 + \ldots e_k$ des indices de Forney représente le plus petit degré externe possible de toute MGP de $\mathcal{C}$ et est égale au degré du code [5].

La valeur maximale des indices de Forney pour un code, c'est-à-dire $\max_{1 \leq i \leq k}\{e_i\}$ est appelée la *mémoire* du code. Ainsi, pour résumer, un $(n, k)$-code convolutif de degré $\delta$ et de distance libre $d_{\text{libre}}$ sera appelé un $(n, k, \delta, d_{\text{libre}})$-code convolutif. Un code convolutif sera dit *optimal* s'il possède la plus grande distance libre parmi tous les codes de paramètres $n, k$ et $\delta$. C'est l'analogue pour les convolutifs des codes en blocs MDS. Pour plus de détail sur la théorie de Forney, voir [66, 67].

**Exemple 19** *Dans l'exemple 18 d'un $(4, 2)$-code convolutif, seule la matrice $G_6$ est canonique. Le degré du code est donc 1 et les indices de Forney sont $(0, 1)$. Il s'agit donc d'un $(4, 2, 1)$-code convolutif. Le calcul de la matrice de parité montre que $d_{libre} = 4$. C'est un code optimal.*

## 7.5 Décodage des codes convolutifs

L'essor des codes convolutifs a eu lieu dès que leur décodage est devenu une chose effective. Historiquement le décodage séquentiel fut le premier décodage des codes convolutifs publié (Wozencraft en 1961 [172]). L'algorithme de Viterbi en 1967 le supplanta, de par son efficacité et son élégance. Il a permis à ces codes de recevoir immédiatement de très nombreuses applications, en particulier militaires [112].

Depuis, de nouveaux algorithmes de décodage sont apparus (voir [98] pour une présentation exhaustive) mais malgré ses limitations, le décodage

5. Cette somme est souvent appelée également *longueur de contrainte* .

de Viterbi reste encore l'un des plus répandus[6]. La majeure partie des applications connues utilisent des codeurs convolutifs de faible degré ($\leq 20$) et ce décodage reste encore attractif malgré sa complexité exponentielle. Je présenterai dans cette section le principe de l'algorithme de Viterbi.

Reprenons l'exemple 7.1 de la section 7.2. Lors du codage, deux données seulement déterminent la sortie du codeur, à chaque temps $t$ : l'état $s_t = (s_t^1, s_t^2)$ du registre et le bit de message entrant dans ce registre $u_t$.

Une manière équivalente de représenter le code et l'opération de codage, représentation plus compacte, est d'utiliser la structure de treillis représentée dans la figure 7.3. Chaque case contient une des quatre valeurs possibles

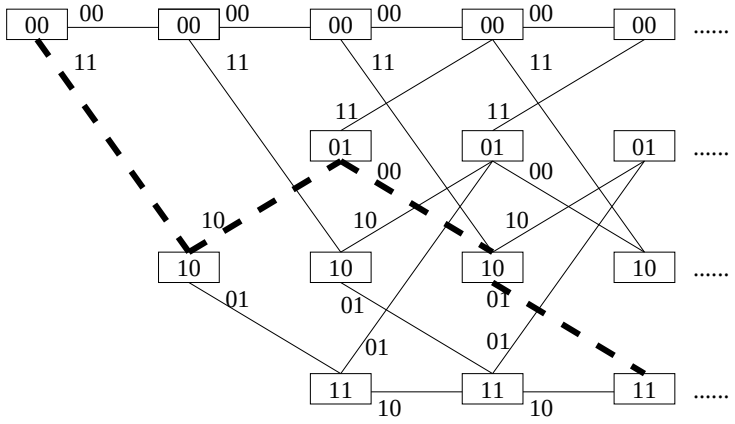


FIGURE 7.3 – Treillis du code convolutif de taux $\frac{1}{2}$

d'état $s_t = (s_t^1, s_t^2)$ et les branches supérieure et inférieure issues de chaque case, correspondent à l'entrée d'un bit de message respectivement 0 et 1. Sur chaque branche figure la valeur des deux bits de sortie du codeur au temps $t$.

Ainsi la séquence message 1011... correspond au chemin dans le treillis 11 10 00 01 ... matérialisé en pointillés gras sur le schéma. Cette représentation est plus pratique et permet facilement de décrire le décodage par l'algorithme de Viterbi. C'est pour cette raison que les codes convolutifs sont souvent appelés *codes en treillis*. Le treillis n'est en fait qu'un diagramme d'état étendu dans le temps.

Notons $\mathbf{u} = u_0 u_1 \dots u_n$ et supposons qu'un canal binaire symétrique soit

6. Toutefois le décodage séquentiel revient actuellement en force, ce dernier permettant un décodage effectif pour de grandes longueurs de contrainte.

utilisé, avec un probabilité d'erreur $0 < p < \frac{1}{2}$. L'état initial du registre est 00 et la séquence **u** suivie de deux zéros de fin de séquence (pour ramener le registre à son état initial) est codée par passage dans le registre. La séquence codée produite est le mot de code **v**. La séquence effectivement reçue sera notée **r**.

Décrivons à l'aide de cette représentation en treillis, le décodage de Viterbi qui consiste en un décodage par maximum de vraisemblance. Supposons que la séquence $\mathbf{r} = 11\ 00\ 11\ 00\ 10$ soit reçue. Le treillis correspondant est représenté par le schéma 7.4. Le décodage va consister à trouver, parmi

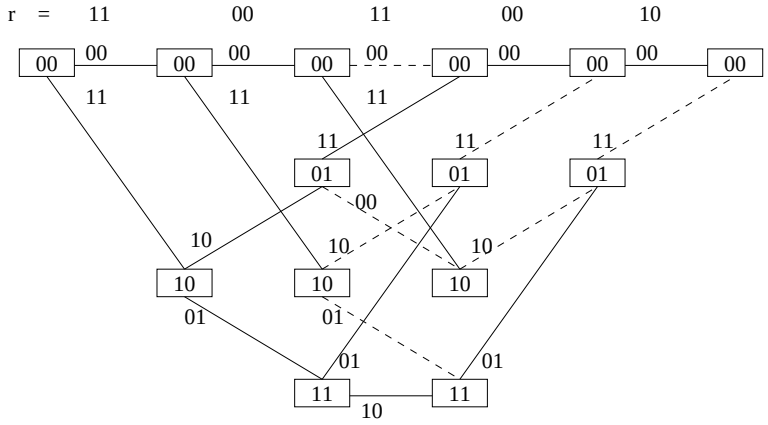


FIGURE 7.4 – Décodage par Viterbi pour la séquence $\mathbf{r} = 11\,00\,11\,00\,10$

les différents chemins du treillis de codage, le chemin le plus probable pour aller d'un état donné au temps $t_0$ (ici 00) à un autre état donné au temps $t$ (ici 00).

La distance de Hamming va permettre de choisir la séquence la plus probable. On considère la séquence $X_t = (x_{t_0}, \ldots, x_t)$ avec $x_i \in \mathbb{F}_2^n$ correspondant à un chemin quelconque, s'arrêtant au temps $t$ dans le treillis. Notons $r_t = (r_{t_0}, \ldots, r_t)$ avec $r_i \in \mathbb{F}_2^n$. On peut calculer facilement $d_H(X_t, r_t)$ à partir de $d_H(X_{t-1}, r_{t-1})$ (où $X_{t-1}$ et $r_{t-1}$ sont les séquences tronquées au temps $t-1$) :

$$d_H(X_t, r_t) = d_H(X_{t-1}, r_{t-1}) + d_H(x_t, r_t)$$

L'algorithme de Viterbi s'effectue alors ainsi : on suppose que pour chaque état du codeur au temps $t$, le meilleur chemin est connu ainsi que la distance de Hamming entre la séquence codée correspondant à ce chemin et la séquence **r** reçue.

1. On calcule pour chaque état au temps $t$, tous ses successeurs (au nombre de $2^k$ dans le cas général, ici 2).
2. Chaque état à l'instant $t+1$ peut-être atteint de $2^k$ manières différentes, on conserve le chemin minimisant la distance de Hamming à la séquence $\mathbf{r}$ reçue.

La complexité de cet algorithme [98] est exponentielle en $k\delta$ ce qui en limite l'usage à de petites longueurs et à des taux de codage $\frac{k}{n} = \frac{1}{n}$.

Pour l'exemple de séquence reçue, $\mathbf{r} = 11\,00\,11\,00\,10$, la meilleure séquence codée trouvée est $\hat{\mathbf{v}} = 11\,10\,11\,00\,00$. Si cette séquence est la séquence véritablement émise, alors deux erreurs ont été corrigées.

Combien d'erreurs peut-on corriger ? On voit facilement dans le schéma 7.4 que la plus petite distance de Hamming entre deux mots de code est 5 ($d_H(00\,00\,00\,00\,00, 11\,10\,11\,00\,00)$. Cela correspond à la distance libre calculée dans la section 7.3.2. Il est donc possible de corriger deux erreurs avec ce code.

# Chapitre 8

# Reconstruction des codes convolutifs

Le problème de la reconstruction des codes convolutifs a été initié par B. Rice [145] en 1995. La solution fournie ne concernait que les codes de taux $\frac{1}{n}$ pour une communication non bruitée. Depuis aucune autre étude avant cette thèse n'a été publiée sur le sujet.

Pourtant l'importance du sujet ne fait aucun doute : les codes convolutifs sont extrêmement utilisés dans un grand nombre d'applications sensibles militaires, en télécommunications spatiales, téléphonie mobile,... Ce sont, de plus, ces codes qui sont généralement utilisés pour coder une information déja chiffrée (par un système à flot par exemple).

Le problème peut se résumer ainsi. Disposant des seules interceptions (message codé), comment peut-on accéder au message clair de départ ? Il s'agit alors de reconstruire, c'est-à-dire de retrouver, l'algorithme de codage utilisé, pour pouvoir, comme le destinataire légitime, décoder. Là encore, comme pour les systèmes de chiffrement à flots, l'avantage est que cette phase de reconstruction est faite une fois pour toute. Les codeurs sont changés très rarement (problèmes de coûts en particulier et de mise en œuvre) et la technique présentée nécessitera toujours beaucoup moins de temps que la durée de vie de ces systèmes [1].

Ce chapitre présente une technique opérationnelle permettant une telle reconstruction que j'ai mise au point et publiée dans [57], quel que soit le type de codeur convolutif, pour une communication bruitée. Après avoir redéfini le problème précisément et fixé les notations utilisées, je présenterai

1. Il existe cependant des cas où les polynômes sont générés aléatoirement pour chaque transmission.

ma technique pour une communication non bruitée. Le résultat essentiel (propositions 38 et 44) est la détermination de la longueur minimale de codé intercepté nécessaire pour être sûr de pouvoir reconstruire le codeur.

Je discuterai ensuite les résultats que j'ai obtenus, en particulier pourquoi plusieurs solutions sont fournies par l'algorithme de reconstruction (propositions 40 et 41). La notion de matrice canonique, présentée dans la section 7.4, prendra ici toute son importance. La proposition 42 permettra de déterminer quelle est la meilleure solution parmi celles fournies par l'algorithme 8.1.

La reconstruction dans le cas d'une communication bruitée sera ensuite présentée. Essentiellement deux approches sont possibles : par les motifs récurrents (théorème 17) ou par essai systématique de tous les motifs d'erreurs de poids maximal donné (proposition 45).

Les résultats de simulation seront finalement donnés. Ils montrent que la reconstruction est véritablement effective et opérationnelle pour des niveaux de bruit relativement élevés. De plus la complexité de la reconstruction se révèle polynomiale en le degré du codeur.

La technique sera présentée pour le cas de code binaire. Elle est immédiatement généralisable pour tout autre corps fini $\mathbb{F}_q$ avec $q = p^r$ et $p$ premier.

Ces résultats ont été publiés dans [57].

## 8.1 Définition du problème et notations

Afin de faciliter la présentation et la compréhension de cette technique, je vais modifier les notations formelles utilisées dans le chapitre 7, section 7.3. En particulier, pour obtenir une technique facile à mettre en œuvre, je vais utiliser non plus des séries formelles de Laurent mais des polynômes et décrire le codage comme une convolution, c'est-à-dire comme finalement le produit de polynômes. Ces notations seront données pour des codeurs de taux $\frac{1}{n}$ pour la simplicité de l'exposé et seront généralisées plus tard.

### 8.1.1 Le codage convolutif

**Définition 44** *Un $(n, k = 1, m)$-code convolutif binaire, de générateurs $f_1(x), f_2(x), \ldots, f_n(x)$ où $f_i(x) = \sum_{j=0}^{m} f_{i,j} x^j$ est le code consistant en tous les mots de code $c(x) = (c_1(x), c_2(x), \ldots, c_n(x))$ où $c_i(x) = m(x) f_i(x)$ et où $m(x) = \sum_{i=0}^{t} m_i x^i$ représente le message d'entrée.*

Le codeur accepte les symboles $m_i$ de la séquence d'entrée à un certain taux ($k$ dans le cas général) et produit les symboles de codé à un taux $n$ fois plus grand. Le taux est alors en $\frac{1}{n}$. On appellera *contrainte* la valeur $K$ égale

à $m+1$. Ce paramètre, fréquent dans un contexte appliqué (implémentation des codes convolutifs), décrit le nombre de cellules du registre utilisé. En fait, par rapport à la figure 7.1 du chapitre 7, section 7.2, on considère les intercellules et non les cellules mémoire de la vision précédente. Ces deux visions sont équivalentes et ne modifient pas les données polynomiales et matricielles définies dans le chapitre 7, section 7.4, les seules déterminant véritablement le codeur considéré. Nous utiliserons dans la suite le paramètre de contrainte $K$ .

Les $c_i(x)$ et $m(x)$ sont des polynômes dont les coefficients, usuellement dans $\mathbb{F}_2$, représentent les symboles respectivement de codé et de clair. Rappelons que les codes convolutifs sont des codes linéaires. En effet, si $c(x)$ et $c'(x)$ sont deux mots de code alors :

$$\begin{aligned} c(x) + c'(x) &= (c_1(x), \ldots, c_n(x)) + (c'_1(x), \ldots, c'_n(x)) \\ &= ((m(x) + m'(x))f_1(x), \ldots, (m(x) + m'(x))f_n(x)) \end{aligned}$$

Les opérations de multiplication polynomiale décrivent aisément le mécanisme de registre à décalage, utilisé pour l'implémentation des polynômes $f_i(x)$ dans un circuit.

**Théorème 16** *[92] Un registre à décalage de polynôme générateur $g(x)$ pour une séquence d'entrée $a_0, a_1, \ldots$, produit la séquence de sortie $c_0, c_1, \ldots$ où*

$$c(x) = a(x)g(x)$$

*avec $c(x) = c_0 + c_1x + \ldots$ et $a(x) = a_0 + a_1x + \ldots$*

**Exemple 20** *Soit le $(2, 1, 3)$-code convolutif de la figure 8.1 de polynômes*

$$f_1(x) = 1 + x + x^3$$
$$f_2(x) = 1 + x^2 + x^3$$

*Le message $m(x) = 1 + x^2$ (10100...) est codé en :*

$$\begin{aligned} c(x) &= ((1 + x^2)f_1(x), (1 + x^2)f_2(x)) \\ &= (1 + x + x^2 + x^5, 1 + x^3 + x^4 + x^5) \\ &= (11100100\ldots, 10011100\ldots) \end{aligned}$$

En pratique $c(x)$ est transmis en un flux unique de symboles, au lieu de $n$ flux parallèles, par un entrelacement des $n$ flux, et ce pour permettre une meilleure résistance au bruit :

$$c_{1,1}, c_{2,1}, c_{1,2}, c_{2,2}, c_{1,3}, c_{2,3}, c_{1,4}, c_{2,4} \ldots$$

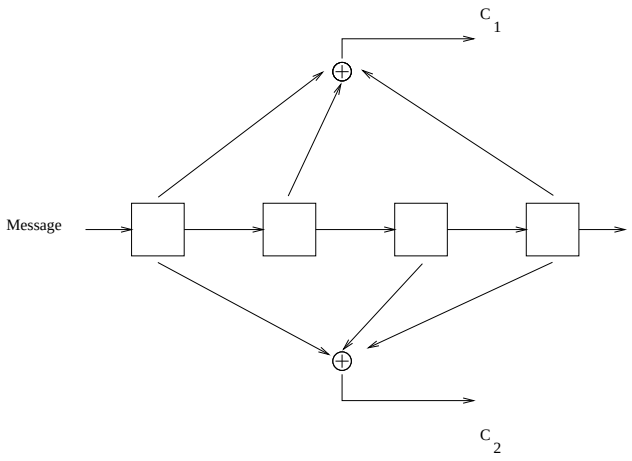


FIGURE 8.1 – $(2, 1, 3)$-code convolutif

Sur l'exemple précédent on transmet donc :

$$11, 10, 10, 01, 01, 11, 00, 00, \ldots$$

### 8.1.2 Le cadre d'étude

Les conditions de cette étude, afin de se placer en situation réelle et d'obtenir une technique opérationnelle, ont été fixées comme suit :

- On dispose de séquences codées interceptées courtes (quelques kbits) ou à défaut le moins possible.
- Le paramètre de bruit $p$ ne dépasse pas 5 %. Toutefois, la technique doit pouvoir traiter, en acceptant un temps de calcul un peu plus long, un bruit avoisinant les 10 %. Le modèle adopté est celui du canal binaire symétrique de paramètre d'erreur $p$.
- Le début et/ou la fin de la transmission peuvent manquer. L'intercepteur ne doit pas à avoir à se soucier de la synchronisation avec la communication interceptée.
- Aucune information ou caractéristique (langue, nature,...) concernant le message n'est utilisée. En particulier, le cas des codeurs systématiques (une des pistes de codé est constituée du message en entrée) n'est pas différencié du cas général.

Les notations suivantes seront utilisées dans ce qui suit :

- $f_i(x) = \sum_{k=0}^{N} f_{i,k}x^k$ pour $i = 1, \ldots, n$ (polynômes générateurs). Dans le cas de codeurs de taux $\frac{k}{n}$, les polynômes seront notés $f_{i,j}(x)$ pour $1 \leq i \leq k$ et $1 \leq j \leq n$.
- $a(x) = \sum_{k=0}^{t+N-1} a_k x^k$, message de symboles $a_k \in \mathbb{F}_2$ de longueur $N+t$, $t$ représentant la longueur utile du message $a(x)$.
- $c_i(x) = \sum_{k=0}^{t-1} c_{i,k}x^k$ piste $i$ de code pour $i = 1, \ldots, n$.
- $N$ est le degré du codeur.
- $K$ la contrainte (ou encore longueur de contrainte du code) avec $K = N + 1$.

## 8.2 Reconstruction pour un canal sans bruit

Je vais d'abord présenter la cas générique et le plus simple, celui esquissé théoriquement par B. Rice [145]. Ce chercheur n'a jamais implémenté sa technique. Je donnerai donc les résultats que j'ai obtenus pour ce cas.

### 8.2.1 Aspects théoriques

Dans sa communication [145], B. Rice présente la méthode théorique suivante pour un taux $\frac{1}{2}$ et une communication non bruitée.

On peut écrire les relations suivantes, liant le message $a(x)$ aux deux pistes de codé :

$$\begin{aligned} a(x)f_1(x) &= u_1(x) + x^N c_1(x) + x^{N+t} v_1(x) \\ a(x)f_2(x) &= u_2(x) + x^N c_2(x) + x^{N+t} v_2(x) \end{aligned}$$

Les polynômes $u_i(x)$ et $v_i(x)$ sont de degré au plus $N-1$. Ils correspondent respectivement au remplissage et à la "vidange" des registres. Les polynômes $c_i(x)$ sont de degré $t$ et correspondent aux deux pistes de code. Il est alors possible d'écrire :

$$\begin{aligned} a(x)f_1(x)f_2(x) &= u_1(x)f_2(x) + x^N c_1(x)f_2(x) + x^{N+t} v_1(x)f_2(x) \\ a(x)f_2(x)f_1(x) &= u_2(x)f_1(x) + x^N c_2(x)f_1(x) + x^{N+t} v_2(x).f_1(x) \end{aligned}$$

ou encore :

$$\begin{aligned} u_1(x)f_2(x) + u_2(x)f_1(x) + x^N(c_1(x)f_2(x) + c_2(x)f_1(x)) \\ + x^{N+t}(v_1(x)f_2(x) + v_2(x)f_1(x)) = 0 \end{aligned}$$

Donc $u_1(x)f_2(x) \equiv u_2(x)f_1(x) \bmod (x^N)$ et alors

$$\begin{aligned} \frac{1}{x^N}(u_1(x)f_2(x) + u_2(x)f_1(x)) + c_1(x)f_2(x) + c_2(x)f_1(x) \\ + x^t(v_1(x)f_2(x) + v_2(x)f_1(x)) = 0 \end{aligned}$$

que l'on peut encore écrire

$$P(x) + c_1(x)f_2(x) + c_2(x)f_1(x) + Q(x) = 0$$

Comme $deg(P(x)) \leq N-1$, les coefficients des puissances de $x$ comprises entre $N$ et $N+t$, dans le polynôme $c_1(x)f_2(x) + c_2(x)f_1(x)$ sont tous nuls. Ceci peut-être exprimé par le système d'équations linéaires homogènes

$$C\underline{f} = \underline{0} \tag{8.1}$$

où $\underline{0}$ est le $(t-N)\times 1$-vecteur nul,$\underline{f}$ le $(2N+2)\times 1$ vecteur colonne dont le transposé est le vecteur

$$(f_{1,0}, f_{1,1}, f_{1,2}, \ldots, f_{1,N}, f_{2,0}, f_{2,1}, f_{2,2}, \ldots, f_{2,N})$$

et $C$ la $(t-N, 2K)$-matrice donnée par :

$$C = \begin{pmatrix} c_{2,N} & c_{2,N-1} & \cdots & c_{2,0} & c_{1,N} & c_{1,N-1} & \cdots & c_{1,0} \\ c_{2,N+1} & c_{2,N} & \cdots & c_{2,1} & c_{1,N+1} & c_{1,N} & \cdots & c_{1,1} \\ c_{2,N+2} & c_{2,N+1} & \cdots & c_{2,2} & c_{1,N+2} & c_{1,N+1} & \cdots & c_{1,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{2,t-1} & c_{2,t-2} & \cdots & c_{2,t-N-1} & c_{1,t-1} & c_{1,t-2} & \cdots & c_{1,t-N-1} \end{pmatrix} \tag{8.2}$$

Les $c_{1,i}$ et $c_{2,i}$ étant connus (codé intercepté), les polynômes sont obtenus par simple résolution du système homogène (calcul du noyau d'une application linéaire). Le rang de la matrice dépend de façon évidente de $a(x)$ (cas trivial si $a(x) = 0$ alors le rang de $C$ est nul).

Pour obtenir une solution non triviale unique ($f_i(x) \neq 0$), le rang doit être au plus égal à $2K-1$ (égal à $2K$ le rang serait maximal et le noyau réduit au vecteur nul). Ceci est possible si

$$t \geq 3N + 1 = 3(K-1) + 1 = 3K - 2$$

Il est donc nécessaire de disposer d'au minimum $L_{\min}$ bits consécutifs de codé avec $L_{\min} = 2(3K-2)$, pour une transmission non bruitée de taux $\frac{1}{2}$.

Pour les valeurs de $N$ utilisées en pratique, les conditions de départ fixées (à la section 8.1.2) sont très largement suffisantes.

Pour le cas du taux $\frac{1}{n}$, la méthode est identique. En considérant les $n-1$ paires de polynômes $(f_1(x), f_i(x))$ (par exemple), on se ramène au cas précédent. Il suffit de vérifier la consistance des solutions sur le polynôme $f_1(x)$ puis on considère un autre polynôme $f_2(x)$ et ainsi de suite. On doit alors disposer de $L_{\min} = n(3K-2)$ bits. En final nous pouvons résumer le résultat ainsi :

**Proposition 38** *Soit un $(n, 1, N)$-code convolutif. La reconstruction du code, pour une communication non bruitée, nécessite au minimum $L_{\min} = n(3N + 1)$ bits consécutifs de séquence codée.*

**Exemple 21** *Pour le G.S.M. où $n = 3$ et $K = 5$, on a seulement besoin de $L_{min} = 39$ bits.*

### 8.2.2 Implémentation

Les valeurs de $N$, et surtout de $n$, ayant des ordres de grandeurs relativement peu importants en pratique, ont été supposés inconnues. Un programme en langage C a été réalisé pour le cas général ($k = 1$ et $n > 2$). L'algorithme est présenté en figure 8.1. En terme de complexité, la triangu-

```
1 lecture du fichier code
2 pour n de 2 à nmax
3       pour K de 3 à Kmax
4             séparer les pistes de code
5             pour p de 2 à n
6                   construire système S_2K pour les pistes 1 et p
7                   trianguler le système S_2K
8                   si rang(S_2K) = 2K aller en 3
9                   sinon calculer la solution p_1,p de S_2K
10            finpour
11            si coincidence des solutions p_1,i sur le polynôme f_1
12               impression des résultats
13               continuer
14            sinon aller en 3
15      finpour
16 finpour
```

TABLE 8.1 – Programme 1 de reconstruction

lation du système $S_{2K}$ (ligne 7) est la phase la plus coûteuse (élimination de Gauss en $\mathcal{O}(K^3)$). La complexité totale de l'algorithme, $K$ et $n$ étant de faibles valeurs (c'est-à-dire bornées), est donc de $\mathcal{O}(nK(n-1)K^3) = \mathcal{O}(K^4)$.

Il est à noter que la reconstruction d'un codeur convolutif est donc d'une complexité beaucoup moins grande que son décodage qui est de complexité exponentielle (voir [98, chap. 4]). La reconstruction sera donc toujours possible pour les codeurs utilisés en pratique.

Une version optimisée du programme a été établie offrant les performances suivantes sur une station D.E.C. Alpha 233 Mhz :

- nmax = 7 et Kmax = 20.
- Temps de calcul $< 1$ s.

### 8.2.3 Analyse des solutions

Plusieurs simulations ont été menées. A chaque fois, les bonnes solutions (*i.e.* les données $n$, $k$ et $f_i$) sont retrouvées mais d'autres solutions sont également proposées. Leur étude a permis de mettre en évidence une structure algébrique assez forte, de définir plus précisément un codeur convolutif et de dégager quelques propriétés intéressantes.

On définit $\mathcal{C}_n = (\mathbb{F}_2[X])^n$ comme l'ensemble des codeurs convolutifs de taux $\frac{1}{n}$. Un codeur convolutif est donc un $n$-uplet de polynômes. Dans la suite, il ne sera considéré que le cas $n = 2$ mais la généralisation des résultats est immédiate. Notons $A = \mathbb{F}_2[X]$.

**Proposition 39** $(\mathcal{C}_2, +, .)$ *est un $A$-module de dimension 2 où les opérations $+$ et $.$ sont définies par*

$$\forall c = (f_1, f_2) \in \mathcal{C}_2, \quad \forall c' = (g_1, g_2) \in \mathcal{C}_2, \quad \forall h \in A$$

$$c + c' = (f_1 + g_1, f_2 + g_2)$$

*et*

$$hc = (hf_1, hf_2)$$

*Preuve.*
démonstration triviale. ■

Avec ces définitions et la description polynomiale du mécanisme de codage convolutif, on a le résultat suivant :

**Proposition 40** *L'ensemble des codeurs convolutifs générant un même codé est un $A$-sous-module $\mathcal{S}$ de dimension 1 de $\mathcal{C}_2$*

*Preuve.*
On veut montrer que si $(f_1(x), f_2(x))$ est une solution du système linéaire

homogène (8.1) alors $\forall h(x) \in A$, $(h(x)f_1(x), h(x)f_2(x))$ est aussi solution du système linéaire homogène étendu, de taille $(2K + 2deg(h(x)))$.

Si $(f_1(x), f_2(x)) \in \mathcal{S}_{2K}$, c'est-à-dire est solution non triviale du système (8.1) à $2K$ inconnues et à $t$ équations, il est aussi solution du système à $(2K + 2)$ inconnues et à $(t - 2)$ équations par plongement de $(\mathbb{F}_2)^{2K}$ dans $(\mathbb{F}_2)^{2K+2}$ car si on a (équation $i$) :

$$(c_{2,N+i}, \ldots, c_{2,i}, c_{1,N+i}, \ldots, c_{1,i}).^t(f_{1,0}, \ldots, f_{1,N}, f_{2,0}, \ldots, f_{2,N}) = 0$$

alors on a

$$(c_{2,N+i+1}, \ldots, c_{2,i}, c_{1,N+i+1}, \ldots, c_{1,i}).^t(f_{1,0}, \ldots, f_{1,N}, 0, f_{2,0}, \ldots, f_{2,N}, 0) = 0$$

qui est l'équation $(i + 1)$ ;

Soit $e \in \mathbb{N}^*$. Plus généralement, $(f_1(x), f_2(x))$ est solution du système à $(2K + 2e)$ inconnues et $(t - 2e)$ équations par plongement de $(\mathbb{F}_2)^{2K}$ dans $(\mathbb{F}_2)^{2K+2e}$. Il suffit donc de montrer que $(xf_1(x), xf_2(x))$ est solution dans $(\mathbb{F}_2)^{2K+2}$, la structure du noyau assurant que

$$(f_1(x), f_2(x)) + (xf_1(x), xf_2(x)) \in \mathcal{S}_{2K+2}$$

En itérant, on peut généraliser à n'importe quel polynôme $h(x)$. Dire que $(xf_1(x), xf_2(x))$ est solution revient à écrire :

$$(c_{2,N+i+1}, \ldots, c_{2,i}, c_{1,N+i+1}, \ldots, c_{1,i}).^t(0, f_{1,0}, \ldots, f_{1,N}, 0, f_{2,0}, \ldots, f_{2,N}) = 0$$

c'est-à-dire

$$\sum_{j=0}^{N} (c_{2,N+1+i-j} f_{1,j} + c_{1,N+1+i-j} f_{2,j}) = 0$$

Cette dernière équation est vérifiée car $(f_1(x), f_2(x))$ est solution dans $(\mathbb{F}_2)^{2K}$. On écrit alors

$$\mathcal{S} = \bigcup_{i=0}^{\infty} \mathcal{S}_{2K+2i}$$

De façon évidente, on voit que $(f_1(x), f_2(x))$ est un élément générateur de $\mathcal{S}$ qui est par conséquent de dimension 1. Une caractérisation plus précise du générateur de $\mathcal{S}$ sera donnée plus loin. ■

L'algorithme détecte donc un saut de rang à partir d'un certain $K$ tel que

$$K = K_{min}$$

(où $\text{rang}(S_{2K}) = 2K-1$)), puis pour les valeurs suivantes de $K$ (on a alors $\text{rang}(S_{2K-(i+1)}) = 2K-(i+1)$ pour $i \geq 1$), les multiples

$$(h(x)f_1(x), h(x)f_2(x))$$

sont construits pour tous les $h(x) \in A$ tels que $\deg(h(x)) = i$.

**Proposition 41** *Soit deux codeurs $(f_1(x), f_2(x))$ et $(g_1(x), g_2(x))$ appartenant au même sous-module $\mathcal{S}$. Il existe alors deux suites-messages $a(x)$ et $a'(x)$ telles que si*

$$\begin{array}{ll}(a(x)f_1(x) = c_1(x)) & (a'(x)g_1(x) = c'_1(x)) \\ (a(x)f_2(x) = c_2(x)) & (a'(x)g_2(x) = c'_2(x))\end{array}$$

*les suites $c_i(x)$ et $c'_i(x)$ ne diffèrent que d'un nombre fini de bits $\mathfrak{b}$ tel que*

$$\mathfrak{b} < 2|deg(f_i(x)) - deg(g_i(x))|.$$

**Définition 45** *Deux codeurs convolutifs sont équivalents si et seulement si ils verifient la propriété 41.*

*Preuve.*
Soit $(f_1(x), f_2(x))$ et $(g_1(x), g_2(x))$, deux codeurs de $\mathcal{S}$ avec $\deg(g_i(x)) > \deg(f_i(x))$. Alors $\exists h(x) \in A$, tel que

$$f_i(x)h(x) = g_i(x).$$

On notera $d = \deg(h(x)) = \deg(g_i(x) - \deg(f_i(x))$. Soient maintenant $a(x)$ et $a'(x)$ deux polynômes "messages" vérifiant

$$\begin{array}{ll}(a(x)f_1(x) = c_1(x)) & (a'(x)g_1(x) = c'_1(x)) \\ (a(x)f_2(x) = c_2(x)) & (a'(x)g_2(x) = c'_2(x))\end{array}$$

Les deux codeurs, appartenant à $\mathcal{S}$, décrivent le même codé (par reconstruction). Prenons $a'(x) = \lfloor \frac{a(x)}{h(x)} \rfloor$ c'est-à-dire

$$a(x) = a'(x)h(x) + r(x) \qquad \text{avec } \deg(r(x)) < \deg(h(x))$$

ou encore

$$a'(x) = \frac{a(x) + r(x)}{h(x)}.$$

Montrons que pour $a(x)$ et $a'(x)$, on obtient le même codé à un nombre, que l'on va borner, de bits près. On sait que :

$$a(x)f_1(x) = u(x) + x^N c_1(x) + x^{N+t}v(x)$$

alors

$$a'(x)g_1(x) = u'(x) + x^{N+d}c'_1(x) + x^{N-d+t}v'(x)$$

avec $\deg(u'(x)) < \deg(f_1(x)) + \deg(h(x))$. Or

$$a'(x)g_1(x) = \frac{a(x)+r(x)}{h(x)}h(x)f_1(x) = a(x)f_1(x) + r(x)f_1(x)$$

On peut alors écrire :

$$u(x) + x^N c_1(x) + x^{N+t}v(x) + r(x)f_1(x) + u'(x) + x^{N+d}c'_1(x) \\ +x^{N-d+t}v'(x) = 0$$

c'est-à-dire :

$$(u(x) + u'(x) + r(x)f_1(x) + \sum_{i=N}^{N+d-1} c_{1,i}x^i) + \\ x^{N+d}(\overline{c_1(x)} + c'_1(x)) + (x^{N+t}v(x) + \\ \sum_{i=N-d+1+t}^{N+t} c_{1,i}x^i + x^{N-d+t}v'(x)) = 0 \qquad (8.3)$$

Or par la reconstruction des deux codeurs décrite dans la partie 8.2.1, $(f_1(x), f_2(x))$ et $(g_1(x), g_2(x))$, entre les indices $(N+d)$ et $(N-d+t)$, les suites $\overline{c_1(x)}$ et $c'_1(x)$ sont identiques. (A l'"extérieur", les longueurs différentes des polynômes $f_i(x)$ et $h(x)f_i(x) = g_i(x)$ produisent des suites $u(x), u'(x), v(x)$ et $v'(x)$ de longueur différentes.Voir la figure 8.2)

Par identification, et en notant que
$\deg(r(x)f_1(x)) < \deg(f_1(x)) + \deg(r(x)) < N + \deg(h(x))$ l'équation (8.3) donne :

$$u'(x) = u(x) + r(x).f_1(x) + \textstyle\sum_{i=N}^{N+d-1} c_{1,i}x^i$$
$$v'(x) = x^d v(x) + \textstyle\sum_{i=1}^{d} c_{1,i+N-d+t}x^i$$

Les suites $c_1(x)$ et $c'_1(x)$ diffèrent de façon claire de $\mathfrak{b} = 2(\deg(h(x)) - 1)$ bits (parties grisées de la figure 8.2) d'où :

$$\mathfrak{b} < 2|\deg(f_i(x) - \deg(g_i(x))|$$

On démontre de même pour $f_2(x), g_2(x), c_2(x)$ et $c'_2(x)$. ■

**Remarques** :

1. La proposition 41 admet une réciproque (omise ici).

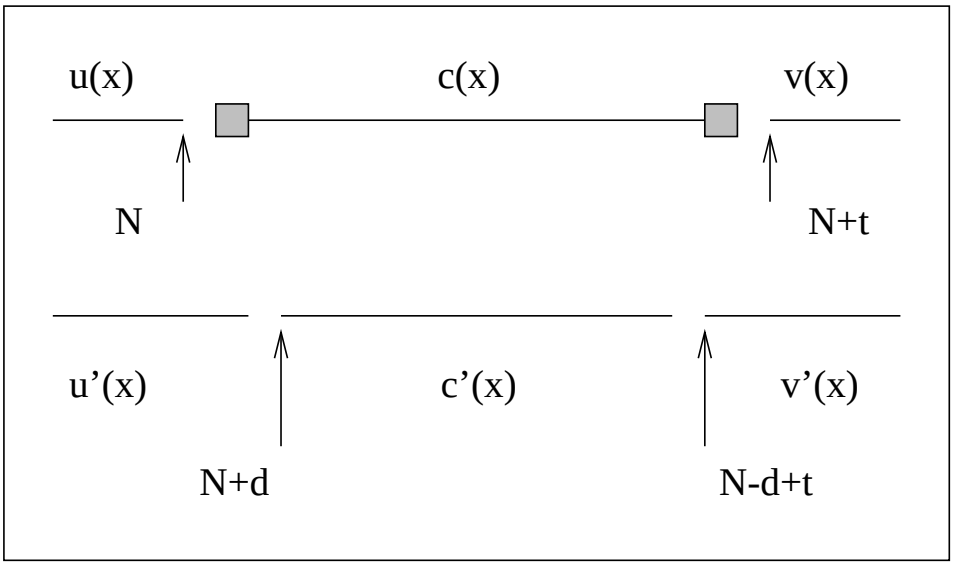


FIGURE 8.2 – Action des deux codeurs

2. La démonstration est constructive : les $\mathfrak{b}$ bits de différence sont calculables, dès lors que l'on connait $h(x)$ et l'un des deux ensembles de données $((f_i(x)),(c_i(x)))$ ou $((g_i(x)),(c'_i(x)))$. Il n'y donc pas de perte d'information. Si on veut transmettre $a(x)$, on peut donc n'envoyer que $c'_i(x)$ c'est-à-dire $c_i(x)$ tronquée de $\mathfrak{b}$ bits $(\overline{c_i(x)})$. On observera que les suites $a(x)$ et $a'(x)$ ont une relation inverse de celle liant les polynômes $f_i(x)$ et $g_i(x)$.
3. Ces résultats se généralisent aisément à $n$ polynômes.

Je vais maintenant donner une caractérisation du générateur de $\mathcal{S}$. Je ne détaille pas la preuve qui s'obtient aisément à partir des propositions 40 et 41.

**Proposition 42** *Le générateur de $\mathcal{S}$ est l'unique couple (respectivement n-uplet) de $\mathcal{S}$ de polynômes $(f_1(x), f_2(x))$ (resp. $(f_1(x),\ldots,f_n(x))$ tels que $f_1(x)$ et $f_2(x)$ soient premiers entre eux (resp. premiers entre eux dans leur ensemble).*

Il est facile de voir que le générateur $g = (f_1(x), f_2(x))$ de $\mathcal{S}$ est le plus

petit élément de $\mathcal{S}$ pour l'ordre suivant (partiel dans $\mathcal{C}_2$ et total dans $\mathcal{S}$) :

$$\begin{array}{ll} & (f_1(x), f_2(x)) \leq (g_1(x), g_2(x)) \\ \Leftrightarrow & \deg(f_i(x)) \leq \deg(g_i(x)) \qquad \text{pour} \quad i = 1, 2 \end{array}$$

On peut donc établir le treillis des sous-modules de $\mathcal{C}_2$ pour l'inclusion. Les sous-modules minimaux de ce treillis sont les couples (respectivement les $n$-uplets) définis par la proposition 42.

Pour les codeurs convolutifs de taux $\frac{1}{n}$ la proposition suivante est alors évidente :

**Proposition 43** *Pour un $(n, 1)$-code convolutif, le générateur $g$ défini par*

$$g = (f_1(x), f_2(x))$$

*de $\mathcal{S}$ forme une matrice canonique.*

*Preuve.*
Cette proposition est évidente pour les codeurs de taux $\frac{1}{n}$. Elle découle naturellement des définitions 41 et 42 et des théorèmes 12 et 13. ■

### 8.2.4 Solution du cas général

Le cas des codeurs de taux $\frac{k}{n}$ va être résolu grâce à la technique présentée dans la section 8.2.1 pour les $(n, 1)$-codes convolutifs. La généralisation passe par une approche récursive.

Tout d'abord, un certain nombre d'observations peuvent être faites, qui vont simplifier le problème. Plus précisément, nous pouvons nous limiter au cas des $(k, k-1)$-codes. En effet, tout $(n, k)$-code peut-être reconstruit en considérant un "sous-codeur" de taux $\frac{k}{k+1}$ (car $k < n$). Il ne reste alors qu'à réitérer la technique pour les $\binom{n}{k+1}$ sous-codeurs possibles. Un simple test de coïncidence sur les polynômes communs permet de tester la consistance des solutions et finalement reconstruire la totalité du codeur.

Considérons donc un $(n, n-1)$-codeur convolutif. Il est défini par les $n$ équations :

$$\sum_{i=1}^{k} a_i(x) f_{i,j}(x) = u_{j,1}(x) + x^N c_j(x) + x^{N+t} u_{j,2}(x) \tag{8.4}$$

avec $i = 1, \ldots, n-1$ et $j = 1, \ldots, n$. Comme précédemment, ne pouvant utiliser le message $a(x)$ (ici matérialisé par $n-1$ sous-séquences $a_i(x)$), nous allons les éliminer successivement. Comme présenté à la section 8.2.1, nous

choisissons (arbitrairement) d'éliminer, par exemple, $a_1(x)$. On obtient alors $(n-1)$ nouvelles équations impliquant seulement $(n-2)$ sous-séquences de message $a_i(x)$ avec $i \neq 1$. Autrement dit, nous avons à traiter le cas d'un codeur de taux $\frac{n-2}{n-1}$. En répétant successivement pour les autres $a_i(x)$, nous obtenons finalement le cas d'un code de taux $\frac{1}{2}$, cas résolu précédemment.

Afin d'illustrer cette approche récursive, considérons un $(3,2)$-codeur convolutif. Il est défini par les trois équations suivantes :

$$\begin{aligned} a_1(x)f_{1,1}(x) + a_2(x)f_{1,2}(x) &= u_{1,1}(x) + x^N c_1(x) + x^{N+t}u_{1,2}(x) \quad (8.5)\\ a_1(x)f_{2,1}(x) + a_2(x)f_{2,2}(x) &= u_{2,1}(x) + x^N c_2(x) + x^{N+t}u_{2,2}(x) \quad (8.6)\\ a_1(x)f_{3,1}(x) + a_2(x)f_{3,2}(x) &= u_{3,1}(x) + x^N c_3(x) + x^{N+t}u_{3,2}(x) \quad (8.7) \end{aligned}$$

Nous choisissons d'éliminer $a_1(x)$. En calculant $f_{2,1}(x).(8.5) + f_{1,1}(x).(8.6)$ et $f_{3,1}(x).(8.5) + f_{1,1}(x).(8.7)$, nous obtenons les deux équations :

$$\begin{aligned} &a_2(x)(\underbrace{f_{1,2}(x)f_{2,1}(x) + f_{2,2}(x)f_{1,1}(x)}_{g_{1,2}(x)}) = \quad (8.8)\\ &\quad f_{2,1}(x)u_{1,1}(x) + f_{1,1}(x)u_{2,1}(x) + x^N(c_1(x)f_{2,1}(x) + c_2(x)f_{1,1}(x))\\ &\quad +x^{N+t}(f_{2,1}(x)u_{1,2}(x) + f_{1,1}u_{2,2}(x)) \end{aligned}$$

$$\begin{aligned} &a_2(x)(\underbrace{f_{1,2}(x)f_{3,1}(x) + f_{3,2}(x)f_{1,1}(x)}_{g_{1,3}(x)}) = \quad (8.9)\\ &\quad f_{3,1}(x)u_{1,1}(x) + f_{1,1}(x)u_{3,1}(x) + x^N(c_1(x)f_{3,1}(x) + c_3(x)f_{1,1}(x))\\ &\quad +x^{N+t}(f_{3,1}(x)u_{1,2}(x) + f_{1,1}u_{3,2}(x)) \end{aligned}$$

Finalement, le calcul de $g_{1,3}(x).(8.8) + g_{1,2}(x).(8.9)$ permet d'éliminer $a_2(x)$. On obtient alors une équation ne faisant plus intervenir de polynômes message :

$$\begin{aligned} 0 = \quad & g_{1,3}(x)f_{1,2}(x)u_{1,1}(x) + g_{1,3}(x)f_{1,1}(x)u_{2,1}(x)\\ + \quad & g_{1,2}(x)f_{3,1}(x)u_{1,1}(x) + f_{1,1}(x)g_{1,2}(x)u_{3,1}(x)\\ + \quad & x^N(c_1(x)f_{2,1}(x)g_{1,3}(x) + c_2(x)f_{1,1}(x)g_{1,3}(x))\\ + \quad & x^N(c_1(x)f_{3,1}(x)g_{1,2}(x) + c_3(x)f_{1,1}(x)g_{1,2}(x))\\ + \quad & x^{N+t}(f_{2,1}(x)g_{1,3}(x)u_{1,2}(x) + f_{1,1}(x)g_{1,3}(x)u_{2,2}(x))\\ + \quad & x^{N+t}(f_{3,1}(x)g_{1,2}(x)u_{1,2}(x) + f_{1,1}(x)g_{1,2}(x)u_{3,2}(x)) \end{aligned}$$

Comme précédemment (cas du taux $\frac{1}{2}$), par identification, les coefficients des puissances de $x$ comprises entre $N$ et $N+t$ dans le polynôme

$$\begin{aligned} c_1(x)(f_{2,1}(x)g_{1,3}(x) + f_{3,1}(x)g_{1,2}(x)) + c_2(x)f_{1,1}(x)g_{1,3}(x)\\ + c_3(x)f_{1,1}(x)g_{1,2}(x) \end{aligned}$$

doivent être nuls. Il faut résoudre un système linéaire homogène à $9N + 3$ inconnues. Dans le cas général, nous avons donc

**Proposition 44** *La reconstruction d'un $(n, n-1, N)$-code convolutif peut être faite par la résolution d'un système linéaire homogène à $n((2^{n-1} - 1)N + 1)$ inconnues. La longueur de suite de codé intercepté nécessaire est $L_{\min} = (n + 1)((2^{n-1} - 1)N + 1) - 2$.*

*Preuve.*
Le nombre d'inconnues s'obtient par récurrence sur $n$. Considérons l'équation $[j]$ $(j = 1, \ldots, n)$.

$$\sum_{i=1}^{k} a_i(x) f_{i,j}(x) = u_{j,1}(x) + x^N c_j(x) + x^{N+t} u_{j,2}(x)$$

où $[k]$ désigne l'équation $k$. On obtient (par élimination d'une première sous-séquence) alors $n - 1$ équations impliquant $n - 2$ sous-séquences $a_i(x)$.

Dans le membre de gauche des équations $[j]$, au départ, les polynômes coefficients des $a_i(x)$ sont de degré $d_G = N$, alors que dans le membre de droite, les coefficients polynomiaux des pistes de code $c_j(x)$ ont un degré $d_D$ nul (bits de codé). Un premier cumul, visant à éliminer $a_1(x)$ (par exemple), s'opère sur les $(n - 1)$ couples $(1, k)$ $(k = 2, \ldots, n)$ :

$$[1].f_{k,1} + [k].f_{1,1}$$

Dans les $n - 1$ équations résultantes, $d_G = 2N$ tandis que $d_D = N$. D'une itération à l'autre (au total, il y en a $k = n-1$, le nombre de sous-séquences de message $a_i(x)$ à éliminer), il est facile de voir que l'on a ($d_D^i$ et $d_G^i$ désignent respectivement les degrés lors de l'élimination de $a_i(x)$) :

$$d_G^i = 2.d_G^{i-1} \qquad d_D^i = d_D^{i-1} + d_G^i$$

Avec les conditions initiales données par les équations de codage de départ, $d_G^1 = N$ et $d_G^1 = 1$ on a finalement après l'élimination des $k = n - 1$ sous-séquences $a_i(x)$ les valeurs suivantes :

$$\begin{aligned} d_G^{n-1} &= 0 \quad \text{polynôme identiquement nul} \\ d_D^{n-1} &= \sum_{i=1}^{n-2} 2^i N = (2^{n-1} - 1)N \end{aligned}$$

Il y a donc $(2^{n-1} - 1)N + 1$ coefficients à retrouver par piste de codé. Comme il y a $n$ pistes de code $c_j(x)$, on obtient le nombre total d'inconnues.

Pour la longueur minimale nécessaire de codé, $t$ doit satisfaire, pour une piste de codé donnée (en considérant la matrice associée au système homogène à résoudre) :

$$t \geq n((2^{n-1}-1)N+1) + (2^{n-1}-1)N + 1 - 2$$

D'où

$$L_{\min} = (n+1)((2^{n-1}-1)N+1) - 2$$

■

La solution du système homogène fournit $n$ polynômes de degré total $(2^{n-1}-1)N$. Pour retrouver les polynômes $f_{i,j}$ de départ, de simples opérations de factorisations ($f_{1,1}$ dans l'exemple précédent pour un codeur de taux $\frac{2}{3}$ est facteur commun de la solution) et de résolution de systèmes linéaires obtenus par identification permettent de complètement retrouver ces polynômes.

Notons qu'en fin d'algorithme, lorsque le saut de rang est détecté, les valeurs $N$ et $n$ sont connues, ce qui rend facile cette recherche des $f_{i,j}$.

L'algorithme utilisé est pratiquement identique à l'algorithme 8.1. On suppose dans la construction du système linéaire que $k = n-1$ et non plus $k = 1$. Il est donc également de complexité $\mathcal{O}(K^4)$.

Enfin, contrairement au cas simple des codeurs de taux $\frac{1}{n}$, la solution n'est pas, de façon évidente, une matrice canonique. S'il est aisé de voir que la solution fournie est réduite au sens de la définition 42, il n'est en revanche pas du tout évident qu'elle soit basique. Les quelques simulations faites semblent l'indiquer mais cela n'est pas un argument suffisant pour le cas général.

Je me hasarderai à énoncer toutefois la conjecture suivante, que je ne suis pas parvenu à démontrer (en fait la propriété "être basique" (proposition 42 et théorème 12) seule est difficile à démontrer) :

**Conjecture 1** *L'algorithme de reconstruction fournit comme solution une matrice canonique.*

## 8.3 Reconstruction pour un canal bruité

Dans un contexte réel, le bruit est généralement de type *burst*, c'est-à-dire que la perturbation des bits se produit par rafales. Ce type de bruit peut-être gênant lors du décodage, si la rafale est trop longue. Le décodage choisit en effet dans le treillis un chemin erroné, d'autant plus difficile à

corriger que la probabilité de bruit sera importante et que la rafale sera longue (voir [98] pour les problèmes de décodage).

Pour la reconstruction, en revanche, ce modèle est très intéressant car il offre l'avantage de conserver des plages propres, c'est-à-dire non bruitées, assez longues, sur lesquelles effectuer la reconstruction. Mais ce modèle est peu aisé à modéliser mathématiquement. J'ai donc choisi un modèle de bruit beaucoup plus défavorable pour les estimations mais plus facile à appréhender : le modèle du canal binaire symétrique [2] (BSC) de paramètre $p$.

Dans ce modèle, le bruit agit de la manière suivante. Pour chaque bit de code, il y a modification (complémentation dans le cas binaire) avec une probabilité $p$, c'est-à-dire ($b_t$ désigne le bit de bruit au temps $t$) :

$$P[b_t = 1] = p \qquad \text{et} \quad P[b_t = 0] = 1 - p = q$$

Si on note $\hat{c}_t$, le bit de codé intercepté au temps $t$, c'est-à-dire $\hat{c}_t = c_t \oplus b_t$, on a :

$$P[\hat{c}_t = c_t] = 1 - p = q \qquad \text{et} \quad P[\hat{c}_t = c_t \oplus 1] = p$$

Le paramètre $p$ est en général inférieur à 0.05. Au delà, le décodage par Viterbi produit trop d'erreurs de décodage pour être efficace [112, pp. 322-329], [128, pp. 310-312].

Le principe est de pouvoir utiliser une plage propre de bits (c'est-à-dire non affectée par le bruit) pour appliquer la méthode précédente, dans le cas non bruité. Il faut donc définir les conditions assurant, avec une probabilité bornée, l'existence d'une telle plage. La reconstruction se fait alors en deux phases :

1. phase statistique : la solution est détectée.
2. phase algébrico-statistique : la solution est validée.

Deux approches ont été considérées. L'une est statistique et utilise la notion d'évènements récurrents. Elle permet de prédire la présence d'une plage

2. Une généralisation de ce modèle a également été considérée : celle du canal AWGN (*Additive White Gaussian Noise*). La variable de bruit ne suit plus une loi discrète à deux états comme dans le cas du BSC, mais une loi normale de moyenne nulle et de variance $\sigma^2$. Les symboles d'entrée peuvent prendre $M$ valeurs discrètes ($M \geq 2$). La fonction de densité de la probabilité conditionnelle de la sortie $y$, étant donné une entrée $x_i$ est alors :

$$P[y|x = x_i] = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-(y-x_i)^2/2\sigma^2}.$$

La technique de reconstruction présentée se généralise sans problème (pour plus de détails sur ce modèle, voir [128, pp. 17-18]).

propre en fonction de $p$ et de la longueur $\mathcal{S}$ de codé intercepté. L'autre envisage tous les motifs d'erreurs possibles d'un poids maximum donné, dépendant de $p$.

### 8.3.1 Les motifs récurrents

En terme de théorie des probabilités, il s'agit d'étudier l'évènement A :

"*avoir une série ininterrompue de $L'_{min}$ succès (success run)*"

si on considère comme succès, l'évènement élémentaire $b_t = 0$, pour l'expérience "bit de bruit". Il suffit alors d'étudier la séquence de bits de bruit $b_t$ et de déterminer les conditions d'existence de séries de 0 d'une longueur donnée.

Une approche efficace et systématique consiste alors à considérer une série de longueur $L'_{\min}$ comme un motif récurrent, c'est-à-dire que A est réalisé à la n-ième expérience, si une nouvelle série de bits réalisant A, de longueur $L'_{min}$ est ajoutée à la séquence générale.

**Définition 46** *[54] Une séquence de $\mathcal{S}$ bits contient autant de séries de zéros de longueur s, qu'il y a de blocs non interrompus, indépendants (non chevauchants) contenant chacun exactement s zéros.*

**Exemple 22** *La séquence* 000010000001 *contient donc 3 séries de zéros de longueur 3 ( pour $n = 3, 8, 11$) et 5 séries de zéros de longueur 2 ($n = 2, 4, 7, 9, 11$).*

L'utilisation d'événements récurrents permet de beaucoup simplifier les calculs mais elle fournit une importante surestimation (assez pessimiste) des conditions réelles. Dans l'exemple précédent, on voit que l'on pourrait considérer non pas 3 séries de zéros mais 6, si l'on considère les chevauchements de blocs, qui alors permettent de considérer, également, une série de longueur 6. Cette surestimation n'est toutefois pas gênante expérimentalement. Elle donne une borne supérieure très large, du nombre de bits à intercepter.

Avec la vision d'événements récurrents, on peut utiliser alors le résultat suivant (voir [54] pour plus de détails) :

**Théorème 17** *[54] Le nombre $\mathbf{N}_s$ de séries de longueur s, présentes dans une séquence de $\mathcal{S}$ bits, pour $\mathcal{S}$ assez grand, est normalement distribué, c'est-à-dire que pour $\alpha$ et $\beta$ fixés, ($\alpha < \beta$), la probabilité que*

$$\frac{n}{\mu} + \alpha\sigma\sqrt{\frac{\mathcal{S}}{\mu^3}} < \mathbf{N}_{\mathcal{S}} < \frac{\mathcal{S}}{\mu} + \beta\sigma\sqrt{\frac{\mathcal{S}}{\mu^3}}$$

*tend vers* $\mathcal{B}(\beta) - \mathcal{B}(\alpha)$ *où* $\mathcal{B}$ *est la fonction de distribution normale et* $\mu$, $\sigma$ *les moyenne et écart-type de récurrence données par :*

$$\mu = \frac{1-p^s}{qp^s} \qquad \sigma^2 = \frac{1}{(qp^s)^2} - \frac{2s+1}{qp^s} - \frac{p}{q^2}$$

Ici, on considère les suites de zéros (pour le bruit) donc

$$q = P[b_t = 0] \qquad \text{et} \quad p = 1 - q = P[b_t = 1]$$

On veut déterminer $\mathcal{S}$, la taille nécessaire de la séquence interceptée de code (et donc de bruit l'affectant), de telle sorte que $\mathbf{N}_s > 1$. Le risque (probabilité qu'il n'y ait pas de plage propre de $s = L_{\min}$ bits) a été fixé à 0.001 ce qui donne $-\alpha = \beta = 3,29$. La table 8.2 donne quelques valeurs de $\mathbf{N}_s$ en fonction de la longueur de contrainte $K$, pour un codeur de taux $\frac{1}{2}$ ($\mathbf{N}_s^-$ désigne le nombre minimum de plages propres au risque 0.001, présentes et $\mathbf{N}_s^+$, le nombre maximum de ces plages).

| $K$ | $\mathcal{S}$(théo.) | $\mathcal{S}^*$ | $\mathbf{N}_s^-$ | $\mathbf{N}_s^+$ |
|---|---|---|---|---|
| 5 | 1100 | 100 | 1 | 38 |
| 10 | 4100 | 1000 | 1 | 27 |
| 15 | 21000 | 7000 | 1 | 24 |
| 20 | 90000 | 44000 | 1 | 26 |

TABLE 8.2 – Tailles $\mathcal{S}$ de séquence en fonction de $K$ (Codeur de taux $\frac{1}{2}$)

Les conditions fixées au départ (quelques kbits) sont donc suffisantes. Rappelons que les valeurs de $\mathcal{S}$, théoriques, sont des surestimations. Une étude plus fine, validée par des simulations, a donné des valeurs $\mathcal{S}^*$ plus précises (quoique encore surestimées : $\mathcal{S}^*$ ) :

**Proposition 45** *Une séquence de longueur*

$$\mathcal{S} = \frac{L_{\min}}{(1-p)^{L}_{\min}}$$

*contient avec un grande probabilité une sous-séquence de longueur* $L_{\min}$ *non bruitée, où* $p$ *est le paramètre de bruit du canal.*

*Preuve.*
Si un bit de bruit se produit, il se propage sur $L_{\min}$ positions, c'est-à-dire

perturbe $L_{\min}$ suites chevauchantes successives. Il faut donc $L_{\min}$ fois plus de suites soit

$$\frac{\mathcal{S}}{L_{\min}}.q^{L_{\min}} \geq 1$$

où $q^{L_{\min}} = (1-p)^{L_{\min}}$ est la probabilité d'avoir $L_{\min}$ bits consécutifs non bruités. ■

### 8.3.2 La reconstruction

Un algorithme similaire à l'algorithme 8.1 (de complexité encore en $\mathcal{O}(K^4)$) a été mis au point, en langage C (voir table 8.3). Dans le cas du taux $\frac{1}{n}$, $k$ sera fixé à 1 dans le cas général $k$ vaudra $n-1$. Afin de faciliter la présentation, je discuterai l'algorithme dans le cas de base d'un taux $\frac{1}{2}$. Le cas général se traite de la même manière. La détection se fait en deux phases.

#### Phase statistique

Le candidat proposé (ligne 9, couple de polynômes), à l'issue de la résolution d'un système de taille $(2K, 2K)$ est testé pour toutes les équations générées à partir de la séquence interceptée ; c'est-à-dire, on calcule pour les $t$ équations :

$$(\hat{c}_{2,N+i}, \ldots, \hat{c}_{2,i}, \hat{c}_{1,N+i}, \ldots, \hat{c}_{1,i}).^t(f_{1,0}, \ldots, f_{1,N}, f_{2,0}, \ldots, f_{2,N}) \qquad (8.10)$$

- Si le candidat est faux (hypothèse $\mathcal{H}_0$), l'équation (8.10) vaut 0, en moyenne, pour la moitié des équations soit $\frac{t}{2}$.
- Si le candidat est le bon (hypothèse $\mathcal{H}_1$), l'équation (8.10) vaut 0 pour toutes les équations non bruitées ( où $\hat{c}_i = c_i$) et pour la moitié des équations bruitées.

Si on dispose de $\mathcal{S}$ bits interceptés, on peut fabriquer $\mathcal{S} - 2K + 1$ équations. Sur les $\mathcal{S}$ bits, et si les bits faux sont répartis de la façon la plus défavorable (distance maximale entre deux bits faux consécutifs), ces derniers (au nombre de $\mathcal{S}p$) affectent chacun $K$ équations (consécutives). Alors, on peut évaluer le nombre $C_0$ de coïncidences à zéro pour (8.10) :

- Hypothèse $\mathcal{H}_0$ :

$$C_0 = \frac{S - 2K + 1}{2}$$

- Hypothèse $\mathcal{H}_1$ :

$$C_0 = S - 2K + 1 - \frac{SpK}{2}$$

Un test simple de comptage permet donc de distinguer les deux hypothèses et de détecter un bon candidat. Différentes simulations ont corroboré ces données, soulignant leur caractère pessimiste (en terme de longueur minimale de suite $\mathcal{S}$ nécessaire).

#### Phase algébrico-statistique

Utilisant les résultats de la partie 8.2.1, on teste, pour $h(x)$ donné (en pratique, $h(x) = 1 + x$ ou $h(x) = 1 + x^2$) et un candidat $(f_1(x), f_2(x))$ l'élément :

$$(h(x)f_1(x), h(x)f_2(x))$$

pour le test précédemment défini.

### 8.3.3 Essai systématique des motifs de poids borné

Une autre approche pour la reconstruction avec un canal bruité, est, connaissant le niveau de bruit du canal, c'est-à-dire le paramètre $p$, de tester tous les motifs d'erreurs possibles. Cette approche est particulièrement indiquée lorsque les codeurs ont des paramètres de petites valeurs (ce qui est souvent le cas en pratique) ou lorsque la séquence interceptée est vraiment très petite (ce qui devient très rare).

Bien sûr, le temps de calcul sera beaucoup plus important. Mais, dans la mesure où la reconstruction ne se fait qu'une fois, cela ne pose pas de problème. On peut admettre six mois de calcul pour un codeur dont la durée de vie sera de plusieurs années [3].

Si pour la reconstruction, une séquence de $L_{\min}$ est nécessaire et que le paramètre de bruit du canal est $p$ alors, en moyenne, sur la séquence interceptée, $e = p.L_{\min}$ bits seront faux. Ce qui signifie qu'il faut essayer tous les motifs de poids au plus $e$ (en fait $e + \epsilon$ où $\epsilon$ dépend du risque d'erreur que l'on se fixe). Il sont au nombre de

$$\mathcal{N}_e = \sum_{i=0}^{e} \binom{\min}{i}.$$

**Exemple 23** *La reconstruction d'un* $(2, 1, 10)$*-code convolutif nécessite*

$$L_{\min} = 62 \text{ } \textit{bits}$$

3. Cette approche n'est donc pas faisable pour des polynômes générés aléatoirement avant la transmission, quand la longueur de contrainte $K$ devient trop grande.

*Si $p = 0.05$, il y aura en moyenne entre 3 et 4 bits faux. Il faudra tester $\mathcal{N}_e = 600.000$ motifs. Cela prendra à peine quelques secondes de calcul.*

*Pour un $(4, 3, 8)$-code, il faut $L_{\min} = 283$ bits. Si $p = 0.01$, alors e vaut en moyenne 3. Cela donne $\mathcal{N}_e = 3.777.768$. Quelques dizaines de minutes de calcul suffisent sur un Pentium II.*

Cette recherche exhaustive peut être menée en parallèle sur plusieurs bouts de séquences.

## 8.4 Résultats numériques

Voici quelques résultats numériques de la reconstruction pour des codeurs de différents taux, lors d'une communication bruitée. Le paramètre de bruit est de 0.01 (table 8.5) et 0.05 (table 8.4). $L_{\min}$ désigne la longueur minimale de séquence interceptée nécessaire pour une communication sans bruit. $\mathcal{S}$ désigne la longueur minimale de séquence interceptée nécessaire pour une communication bruitée (voir proposition 45). $\mathcal{S}^*$ donne la longueur de suite pour une recherche exhaustive des motifs d'erreur.

Les temps de calcul ne sont pas donnés dans le cas général (suite de longueur $\mathcal{S}$). Ils ne dépassent pas l'heure. Pour le cas d'une suite de longueur $\mathcal{S}^*$, on donne le nombre approximatif de motifs $\mathcal{N}_e$ à tester (valeur moyenne).

Ces valeurs représentent encore des valeurs par excès. En effet, pour des applications réelles, le modèle de bruit est plus favorable que le modèle gaussien. Le niveau de bruit est en général assez faible. De plus les capacités d'interception modernes permettent de disposer de suites de plusieurs megabits. La technique décrite dans ce chapitre permet donc une reconstruction dans de bonnes conditions.

```
1 lecture du fichier code
2 pour n de 2 à nmax
3       pour K de 3 à Kmax
4             séparer les pistes de code
5             pour p de 2 à n
6                   fabriquer les S − 2K + 1 = e équations pour les pistes 1 et p
7                   pour s de 1 à e − 2K + 1
8                         fabriquer le (2K, 2K)-système S_s pour les pistes 1 et p
9                         trianguler le système S_s
10                        si rang(S_s) ≠ 1 aller en 7
11                        sinon calculer la solution p_{1,p,s} de S_s
12                        pour equ de 1 à e
13                              si ⟨p_{1,p,s}, Equ_{equ}⟩ = 0 compteur++
14                        finpour
15                        si compteur ≥ Seuil
16                           tester h(x)p_{1,p,s}
17                           si succès aller en 5
18                           sinon aller en 7
19                  finpour
20            finpour
21            si coincïdence des solutions p_{1,i} sur le polynôme f_1
22               impression des résultats
23               continuer
24            sinon aller en 3
25      finpour
26 finpour
```

TABLE 8.3 – Programme 2 de reconstruction

| $(n,k,\delta)$-code | $L_{\min}$ | $\mathcal{S}$ | $\mathcal{S}^*$ | $\mathcal{N}_e$ |
|---|---|---|---|---|
| $(2,1,5)$ | 32 | 166 | 32 | 529 |
| $(2,1,10)$ | 62 | 1.491 | 62 | 600.000 |
| $(3,1,5)$ | 48 | 563 | 48 | 18.473 |
| $(3,1,10)$ | 93 | 10.970 | 93 | $2^{25}$ |
| $(3,2,5)$ | 62 | 1.491 | 62 | 600.000 |
| $(3,2,10)$ | 122 | 63.691 | 122 | $2^{31}$ |
| $(4,1,5)$ | 64 | 1.706 | 64 | 679.121 |
| $(4,1,10)$ | 124 | 71.729 | 124 | $2^{32}$ |
| $(4,3,5)$ | 178 | 1.642.927 | 178 | $2^{49}$ |

TABLE 8.4 – Reconstruction : exemples ($p = 0.05$)

| $(n,k,\delta)$-code | $L_{\min}$ | $\mathcal{S}$ | $\mathcal{S}^*$ | $\mathcal{N}_e$ |
|---|---|---|---|---|
| $(2,1,5)$ | 32 | 45 | 32 | 33 |
| $(2,1,10)$ | 62 | 116 | 62 | 63 |
| $(3,1,5)$ | 48 | 78 | 48 | 49 |
| $(3,1,10)$ | 93 | 236 | 93 | 94 |
| $(3,2,5)$ | 62 | 116 | 62 | 63 |
| $(3,2,10)$ | 122 | 416 | 122 | 7504 |
| $(4,1,5)$ | 64 | 122 | 64 | 65 |
| $(4,1,10)$ | 124 | 432 | 124 | 7751 |
| $(4,3,5)$ | 178 | 1066 | 178 | 15932 |

TABLE 8.5 – Reconstruction : exemples ($p = 0.01$)

# Chapitre 9

# Le codes convolutifs poinçonnés

Les codes poinçonnés ont été introduits par J.B. Cain, G.C. Clark et J.M. Geist [17] en 1979. Le poinçonnage permet par une technique simple de générer de nouveaux codes convolutifs à partir d'anciens. Ils ont connus depuis un grand succès car leurs performances sont globalement aussi bonnes que celles des codes de mêmes paramètres et leur mise en œuvre est beaucoup plus facile que pour les codes convolutifs non poinçonnés. Un grand nombre de résultats ont depuis été obtenus, notamment dans la recherche de codes à haut débit pour le décodage de Viterbi ou le décodage séquentiel [6, 84, 174]. L'étude de codeurs catastrophiques a été également envisagée [133].

## 9.1 Génération des codes poinçonnés

### 9.1.1 Exemple introductif

Soit un codeur convolutif de taux $\frac{1}{2}$ de matrice génératrice

$$G(X) = (1 \oplus X \oplus X^2, 1 \oplus X^2)$$

(voir la figure 7.1 de la section 7.2, chapitre 7). A chaque coup d'horloge, un bit de message $u_t$ entre dans le registre et deux bits de codé sont produits en sortie $v_t^1, v_t^2$.

Ce cadencement de l'horloge est arbitraire et on pourrait très bien imaginer que l'horloge tourne moitié moins vite. Dans ce cas, le codeur accepte, à chaque nouveau coup d'horloge deux bits à la fois $u_{2t}$ et $u_{2t+1}$ et produira en sortie quatre bits de sortie $v_{2t}^1, v_{2t+1}^1, v_{2t}^2, v_{2t+1}^2$.

En clair, le changement de cadencement d'horloge revient à un regroupement des bits en blocs de taille donnée (ici 2) et sur cet exemple un $(2,1)$-code convolutif s'est transformé en $(4,2)$-code convolutif. Toutefois ce changement n'est qu'un problème de vision et n'a pas modifié le code. Les codes obtenus par regroupement par blocs seront appelés codes convolutifs regroupés.

Sur ce $(4,2)$ code, effaçons ou encore *poinçonnons* les bits $v_{2t+1}^1,\ t \geq 0$ de chaque blocs de quatre bits de sortie. Le codeur ne produit donc plus que trois bits en sortie, à chaque coup d'horloge. Autrement dit :

$$\begin{pmatrix} v_0^1 & v_1^1 & v_2^1 & v_3^1 & v_4^1 & v_5^1 & \cdots \\ v_0^2 & v_1^2 & v_2^2 & v_3^2 & v_4^2 & v_5^2 & \cdots \end{pmatrix} \Rightarrow \begin{pmatrix} v_0^1 & & v_2^1 & & v_4^1 & & \cdots \\ v_0^2 & v_1^2 & v_2^2 & v_3^2 & v_4^2 & v_5^2 & \cdots \end{pmatrix}$$

Le poinçonnage est donc réalisé selon le motif ou *matrice de poinçonnage* (encore dite matrice de perforation ) $P$ :

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

Ainsi le $(2,1)$-code convolutif de départ, une fois poinçonné avec la matrice $P$ est devenu un $(3,2)$-code convolutif. La distance minimale de ce code reste 3, identique à celle du code parent et le degré $\delta$ est de 2.

D'une manière générale, l'opération de poinçonnage peut-être résumée ainsi : à partir d'un $(n,k,\delta)$-code convolutif parent, on opère un regroupement par blocs de profondeur $M$, c'est-à-dire que l'on groupe les bits de message par blocs de taille $M$. Le code convolutif obtenu est un $(nM,kM,\delta)$-code convolutif de même distance libre que le code parent. On considère ensuite une matrice de perforation $P$ (matrice 0-1 de taille $n \times M$) de densité $N$ ($N$ valeurs non nulles égales à 1). Si on perfore tous les bits nuls du motif sur chaque bloc de sortie de taille $nM$, on obtient un $(N,kM,\delta)$-code convolutif.

Les codes obtenus ainsi sont généralement très performants (grande distance libre). De plus, ils abaissent la complexité du décodage de Viterbi par rapport à celle du code parent d'un facteur $2^{M-1}$ [115].

### 9.1.2 Résultats fondamentaux

Cette section présente les principaux résultats concernant la construction des codes convolutifs poinçonnés. Une description exhaustive et technique pourra être trouvée dans [115, chap. 12].

La construction algébrique est centrée autour de la notion de *décomposition polyphase* d'un polynôme ou d'une série formelle.

**Définition 47** *Soit $f(X)$ une série formelle (ou un polynôme) d'indéterminée $X$.*

$$f(X) = a_0 + a_1X + a_2X^2 + \dots$$

*Alors pour tout entier $M \geq 1$ la décomposition polyphase d'ordre $M$ de $f(X)$ est la liste des $M$ sous-séries (ou sous-polynômes) suivantes :*

$$(f_{0,M}(X), \dots, f_{M-1,M}(X)) = \left( \sum_{k\geq 0} a_{kM} X^k, \dots, \sum_{k\geq 0} a_{kM+M-1} X^k \right) \quad (9.1)$$

*La jème composante est appelée* composant polyphase *de rang $j$ et d'ordre $M$ de $f(X)$.*

En fait, le composant polyphase de rang $j$ et d'ordre $M$ n'est que la sous-série obtenue en partant du jème terme de la série $f(X)$ et en prenant un terme tous les $M$.

**Exemple 24** *Soit*

$$f(X) = 3 + X + 4X^2 + X^3 + 5X^4 + 9X^5 + 2X^6 + 6X^7$$

*alors la décomposition polyphase d'ordre 3 est donnée par :*

$$(3 + X + 2X^2, 1 + 5X + 6X^2, 4 + 9X)$$

Les composants polyphase de rang $j$ et d'ordre $M$ sont utilisés pour construire une matrice très importante, la *matrice pseudocirculante polycyclique* d'ordre $M$ associée à $f(X)$ (ou matrice PCPC).

**Définition 48** *Soit $f(X)$ une série formelle (ou un polynôme) d'indéterminée $X$.*

$$f(X) = a_0 + a_1X + a_2X^2 + \dots$$

*alors pour tout entier $M \geq 1$, si $(f_0, f_1, \dots, f_{M-1})$ est la décomposition polyphase d'ordre $M$ de $f(X)$, alors la matrice pseudocirculante polycyclique associée est la matrice $M \times M$ poynomiale $f^{[M]}(X)$ définie par :*

$$f^{[M]}(X) = \begin{pmatrix} f_0 & f_1 & \cdots & f_{M-1} \\ Xf_{M-1} & f_0 & \cdots & f_{M-2} \\ Xf_{M-2} & Xf_{M-1} & \cdots & f_{M-3} \\ \vdots & \vdots & \ddots & \vdots \\ Xf_1 & Xf_2 & \cdots & f_0 \end{pmatrix}$$

**Exemple 25** *En reprenant l'exemple 24, la matrice PCPC d'ordre 3 est*

$$f^{[3]}(X) = \begin{pmatrix} 3+X+2X^2 & 1+5X+6X^2 & 4+9X \\ 4X+9X^2 & 3+X+2X^2 & 1+5X+6X^2 \\ D+5X^2+6X^3 & 4X+9X^2 & 3+X+2X^2 \end{pmatrix}$$

Muni de ces outils, nous pouvons donner maintenant les résultats fondamentaux.

**Théorème 18** *Si $\mathcal{C}$ est un $(n,k)$-code convolutif, alors le regroupement par bloc d'ordre $M$ de $\mathcal{C}$, noté $\mathcal{C}^{[M]}$, est un $(nM,kM)$-code convolutif. Si $G(X) = (g_{i,j}(X))$ est une matrice polynomiale de taille $k \times n$, génératrice du code $\mathcal{C}$, alors une matrice génératrice du code $\mathcal{C}^{[M]}$, notée $G^{[M]}(X)$, peut être obtenue à partir de $G(X)$ en remplaçant chaque entrée $g_{i,j}(X)$ de $G(X)$ par la matrice PCPC d'ordre $M$ correspondante et en entrelaçant les colonnes avec une périodicité $M$.*

**Exemple 26** *Reprenons l'exemple du $(2,1,2)$-code convolutif de la section 9.1.1 de matrice génératrice $G(x) = (1 \oplus X^2, 1 \oplus X \oplus X^2)$ et de distance libre 5.*

*Prenons $M = 2$. La décomposition polyphase d'ordre 2 associée à $g_1(X)$ est $(1 \oplus X, 0)$, celle associée à $g_2(X)$ est $1 \oplus D, 1)$. Les matrices polyphase associées sont donc*

$$g_1^{[2]}(X) = \begin{pmatrix} 1 \oplus X & 0 \\ 0 & 1 \oplus X \end{pmatrix} \qquad g_2^{[2]}(X) = \begin{pmatrix} 1 \oplus X & 1 \\ X & 1 \oplus X \end{pmatrix}$$

*Alors la matrice bloc $(g_1^{[2]}(X) g_2^{[2]}(X))$ après entrelacement de périodicité 2, est donnée par*

$$(g_1^{[2]}(X) g_2^{[2]}(X)) = \begin{pmatrix} 1 \oplus X & 1 \oplus X & 0 & 1 \\ 0 & X & 1 \oplus X & 1 \oplus X \end{pmatrix}$$

Pour les propriétés des codes construits par regoupement d'ordre $M$, le théorème suivant est important.

**Théorème 19** *Si $G(X)$ est matrice polynomiale génératrice d'un code $\mathcal{C}$ alors $G^{[M]}(X)$ sera une matrice génératrice canonique pour $\mathcal{C}^{[M]}$. Le degré et la distance libre du code $\mathcal{C}^{[M]}$ seront les mêmes que ceux du code parent.*

### 9.1.3 Le poinçonnage

**Définition 49** *Si $G(X)$ est une $k \times n$ matrice polynomiale et si $P$ est une $n \times M$ matrice de 0 et de 1, les $nM$ entrées de $G^{[M]}(X)$ sont en relation biunivoque avec les entrées de $P$. La matrice $G_P(X)$, génératrice du code obtenu du code $\mathcal{C}^{[M]}$ par perforation selon la matrice $P$, est obtenue à partir de $G^{[M]}(X)$ en supprimant les colonnes correspondants aux entrées nulles de $P$. Le code obtenu, noté $\mathcal{C}_P$ est appelé version poinçonnée du code parent $\mathcal{C}$. C'est un $(N, kM, \delta', d'_{free})$-code convolutif, où $N$ est le poids de Hamming de la matrice $P$.*

Contrairement aux codes obtenus par regroupement, ceux poinçonnés n'ont pas forcément les mêmes degré $\delta$ et surtout distance libre que leur code parent. La détermination de ces paramètres se fait au cas par cas (voir [6, 49, 174]).

**Exemple 27** *Soit le code parent $(3, 1, 4, 9)$ de matrice génératrice*

$$G(X) = (1 \oplus X^2 \oplus X^3 \quad 1 \oplus X^3 \quad 1 \oplus X \oplus X^2 \oplus X^4).$$

*Ce code n'est pas optimal [5] car il existe un $(3, 1, 4, 12)$ code (voir annexe C). Prenons la matrice de perforation*

$$P = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}.$$

*La matrice $G^{[2]}(X)$ est*

$$G^{[2]}(X) = \left( \begin{pmatrix} 1 \oplus X & X \\ X^2 & 1 \oplus X \end{pmatrix}, \begin{pmatrix} 1 & X \\ X^2 & 1 \end{pmatrix}, \begin{pmatrix} 1 \oplus X \oplus X^2 & 1 \\ X & 1 \oplus X \oplus X^2 \end{pmatrix} \right)$$

*Par perforation selon le motif $P$, nous obtenons la matrice génératrice suivante :*

$$G_P(X) = \begin{pmatrix} 1 \oplus X & X & 1 \\ X^2 & 1 & 1 \oplus X \oplus X^2 \end{pmatrix}$$

*$G_P(X)$ est canonique donc le code poinçonné est un $(3, 2, 3)$ code convolutif. Son degré est stritement inférieur à celui du code parent. Sa distance libre est de 4. Au final, nous avons obtenu un $(3, 2, 3, 4)$-code convolutif.*

## 9.2 Résultats sur la reconstruction

La reconstruction des codes poinçonnés ne diffère donc en rien de celle des codeurs de taux $\frac{k}{n}$ (voir chapitre 8, section 8.2.4). J'ai publié les résultats dans [61].

Mais il est intéressant de voir comment la théorie conduisant à la construction des codes poinçonnés peut intervenir dans la reconstruction du cas général des codes convolutifs. En fait, lors de la reconstruction d'un $(n, k, \delta)$-code convolutif (non poinçonné), on ne retrouvera pas forcément le code réellement utilisé par l'opérateur. En vertu de la façon dont sont construits les codes regroupés :

- Si $\gcd(n, k) = k$, on retrouvera un $(\frac{n}{k}, 1, \delta')$-code.
- Si $\gcd(n, k) = d$ $(1 < d < k)$ on retrouvera un $(\frac{n}{d}, \frac{k}{d}, \delta'')$-code.

Cela revient à considérer un cadencement d'horloge $\gcd(n, k)$ fois plus rapide (voir section 9.1.1).

Pour les codes poinçonnés, la même constatation est théoriquement possible. Toutefois, les bons codes poinçonnés (ceux qu'utilisent en théorie les opérateurs) sont tous des codes de taux $\frac{n-1}{n}$ (voir annexe C). Une telle simplification est donc peu probable.

Ces résultats ont été publiés dans [61].

# Conclusion

J'ai développé une technique permettant, dans le cas d'une communication bruitée, de reconstruire le codeur utilisé. Le niveau de bruit peut être relativement important sans que cela rende cette technique inopérationnelle. Dans certain cas, non pas le codeur véritablement utilisé est retrouvé, mais un codeur équivalent. Dans tous les cas, le décodage par l'intercepteur devient alors possible.

La complexité de cette technique est polynomiale en le degré du codeur alors que son décodage est exponentiel en ce même paramètre. Cela assure à cette technique de fonctionner dans la plupart des cas.

Pour les codes convolutifs poinçonnés, en exploitant leur équivalence avec certains codes convolutifs non poinçonnés, la reconstruction fonctionne également, sans modifier la complexité générale de cette méthode.

Les temps de calcul sont de l'ordre de quelques secondes à quelques minutes, selon les paramètres du code à reconstruire. Si le canal est sans bruit, moins d'une seconde de calcul et une suite très courte sont nécessaires.

Pour une communication bruitée, deux approches ont été utilisées. L'une, exploitant la présence de plages de bits propres de longueur suffisante, permet avec une suite interceptée relativement peu importante, de reconstruire en quelques secondes à quelques dizaines de minutes, la totalité du codeur. Les moyens d'interception modernes permettent toujours en pratique de disposer d'une telle suite.

L'autre approche, plus gourmande en temps de calcul, explore tous les motifs d'erreur possibles (en fonction du paramètre de bruit du canal). Cela permet de traiter les cas où la suite disponible est très courte au prix d'une complexité de calcul qui peut-être très lourde (temps de calcul de quelques minutes à plusieurs mois). Il ne faut pas toutefois oublier que ce travail est fait une fois pour toute, ce qui autorise pour l'attaquant des temps de calcul qui seront de toute façon inférieurs à la durée de vie du codeur.

Comment lutter contre une telle technique de reconstruction ? Même si l'utilisation d'un code n'a pas pour objet de dissimuler de l'information, un

concepteur de chaîne de transmission n'a pas forcément envie de faciliter la vie des intercepteurs, surtout quand on sait que le codage est souvent utilisé pour des transmissions chiffrées.

La reconstruction des codes en blocs est déja pratiquement résolue par d'autres auteurs. Celle des codes convolutifs l'est totalement. Une perspective prometteuse semblerait être d'utiliser des turbo-codes. Consistant en deux codes convolutifs mis en parallèles, avec pour l'un des deux, application d'une permutation par blocs de grande taille sur le message en entrée, la reconstruction du codeur dans ce cas semble être plus difficile, si l'on veut retrouver la totalité du codeur et en particulier la permutation.

Toutefois, il est toujours possible de retrouver les codes convolutifs constituant le turbo-codes par la technique présentée dans cette thèse.

# Perspectives

Quel avenir pour la cryptanalyse ?

Celui de la cryptographie n'est pas véritablement aussi directement intéressant. En effet, d'abord parce que les connaissances dans ce domaine ont très souvent été déterminées, en réaction, par les attaques connues [1]. C'est la seule aune à laquelle mesurer la sécurité d'un système. D'autre part, avec ces connaissances et en tenant compte de l'évolution de la puissance de calcul, le cryptographe possède maintenant plusieurs longueurs d'avance sur le cryptanalyste. C'est dans cet esprit qu'a été lancé et s'est déroulé le concours AES. De plus, il est assez facile__ cela ne requiert que peu de connaissances, accessibles de nos jours dans tous les bons manuels ou sur Internet__ de concevoir un système offrant suffisamment de sécurité pour contrarier tout service de cryptanalyse.

Cryptanalyser un système n'est par contre pas une chose facile et cela requiert des connaissances qu'il faut souvent créer *ex nihilo*, du temps, de la puissance de calcul... et le goût du défi et de l'inconnu. C'est la raison pour laquelle, il demeure plus de systèmes inviolés opérationnellement parlant [2] que de systèmes véritablement cryptanalysés. Cela jette une lumière inquiétante sur la situation résultant en France d'une libéralisation inconsidérée parce ce que non préparée.

Alors quel est l'avenir de la cryptanalyse ? Est-ce un combat perdu d'avance ? Non ! L'ampleur de la tâche est bien sûr colossale mais le défi est passionnant. Il réclame un nouvel élan dans la recherche, comparable à celui des années 70 et 80 qui a connu les Meier, Staffelbach, Siegenthaler,...

Les techniques développées dans cette thèse se basent quasi-totalement

1. Il ne faut pas ignorer cependant que de très importants travaux théoriques, "en amont", ont permis de systématiser les connaissances et de leur donner un fondement théorique. La théorie de la complexité constitue le meilleur exemple.

2. On écartera les cryptanalyses fantaisistes qui pullulent dans les congrès de cryptologie et réclament une infinité de couples clair/chiffré et une complexité supérieure à la recherche exhaustive.

sur l'utilisation plus ou moins grande de propriétés de linéarité. L'utilisation et l'étude de la non-linéarité (équations de degré $> 1$ en premier lieu) devrait permettre de faire un grand bond en avant, si l'on parvient à créer les outils adéquats. Les bases de Gröbner semblent d'ores et déja prometteuses. La théorie des codes et ses outils très élaborés confirmera les progrès déja constatés (décodage de Sudan par exemple). La combinatoire devrait permettre de mieux comprendre ce qu'est vraiment un système de chiffrement par blocs ou une fonction de hachage.

Les reconstructions présentées dans cette thèse passent toutes par la manipulation et l'étude de polynômes. Beaucoup de propriétés restent encore à découvrir. Le problème de leur classification se pose encore.

Pour résumer, il est essentiel et urgent de lancer des études de fond dans différents domaines. Pour ma part, je n'en citerai que deux qui me semblent importants :

- l'étude des registres à rétroaction non linéaire. Comprendre ces objets permettra certainement de mieux comprendre le comportement non-linéaire en général et de développer de nouvelles attaques, plus générales et donc plus efficaces. Le point de départ pourrait être la thèse de Jansen [95].
- l'étude combinatoire des schémas par blocs. Là tout reste à faire. Il est par exemple urgent de convaincre définitivement que "trapper" de tels systèmes est possible, permettant ainsi une cryptanalyse facile. Encore trop peu de spécialistes en sont convaincus ou acceptent de l'être. La transformation de schéma pour ces systèmes par blocs est également une chose intéressante à étudier. Elle a récemment été initiée pour le DES [78]. Est elle systématiquement possible ?

Quoi qu'il en soit l'avenir de la cryptanalyse semble assuré, le travail ne manque pas.

# Annexe A

# Valeurs utiles $d$ pour l'attaque par décimation

Je donne ici les meilleures valeurs du facteur de décimation $d$ d'un registre permettant de simuler un registre de longueur $L^*$ avec $L^* \leq \frac{L}{2}$. Ces calculs ont été obtenus par une recherche exhaustive avec la librairie GPM et Magma 2.6 pour la phase de factorisation de $2^L - 1$.

| $L$ | $L^*$ | $d$ | $L$ | $L^*$ | $d$ |
|---|---|---|---|---|---|
| 4 | 2 | 5 | 20 | 10 | 1025 |
| 6 | 3 | 9 | 21 | 7 | 16513 |
| 8 | 4 | 17 | 22 | 11 | 2049 |
| 9 | 3 | 73 | 24 | 12 | 12291 |
| 10 | 5 | 33 | 25 | 5 | 1082401 |
| 12 | 6 | 65 | 26 | 13 | 8193 |
| 14 | 7 | 129 | 27 | 9 | 262657 |
| 15 | 5 | 1057 | 28 | 14 | 16385 |
| 16 | 8 | 257 | 30 | 15 | 32769 |
| 18 | 9 | 511 | 32 | 16 | 65537 |

TABLE A.1 – Valeurs pour $L \leq 32$

| $L$ | $L^*$ | $d$ | $L$ | $L^*$ | $d$ |
|---|---|---|---|---|---|
| 33 | 11 | 4196353 | 49 | 7 | 270549121 |
| 34 | 17 | 131073 | 50 | 25 | 33554433 |
| 35 | 7 | 270549121 | 51 | 17 | 17180000257 |
| 36 | 18 | 786435 | 52 | 26 | 67108865 |
| 38 | 19 | 524289 | 54 | 27 | 134217729 |
| 39 | 13 | 67117057 | 55 | 11 | 17600780175361 |
| 40 | 20 | 1048577 | 56 | 28 | 268435457 |
| 42 | 21 | 14680071 | 57 | 19 | 274878431233 |
| 44 | 22 | 4194305 | 58 | 29 | 536870913 |
| 45 | 15 | 1073774593 | 62 | 31 | 2147483649 |
| 46 | 23 | 8388609 | 63 | 21 | 4398048608257 |
| 48 | 24 | 16777217 | | | |

TABLE A.2 – Valeurs pour $33 \leq L \leq 63$

| $L$ | $L^*$ | $d$ | $L$ | $L^*$ | $d$ |
|---|---|---|---|---|---|
| 64 | 32 | 4294967297 | 80 | 40 | 3298534883331 |
| 65 | 13 | 4504149450301441 | 81 | 27 | 18014398643699713 |
| 66 | 33 | 8589934593 | 82 | 41 | 2199023255553 |
| 68 | 34 | 17179869185 | 84 | 42 | 4398046511105 |
| 69 | 23 | 70368752566273 | 85 | 17 | 295150156996346511361 |
| 70 | 35 | 34359738369 | 86 | 43 | 8796093022209 |
| 72 | 36 | 68719476737 | 87 | 29 | 288230376688582657 |
| 74 | 37 | 137438953473 | 88 | 44 | 17592186044417 |
| 75 | 25 | 1125899940397057 | 90 | 45 | 35184372088833 |
| 76 | 38 | 274877906945 | 91 | 13 | 302268352895954163081217 |
| 77 | 11 | 73823022692637345793 | 92 | 46 | 70368744177665 |
| 78 | 39 | 549755813889 | | | |

TABLE A.3 – Valeurs pour $64 \leq L \leq 92$

# Annexe B

# Tables de codes convolutifs optimaux

Cette annexe présente quelques codes convolutifs optimaux. Pour une liste plus exhaustive voir [98, 108, 109, 132]. Les notations sont les suivantes : $\delta$ représente le degré du code, $\Delta(n, k, \delta)$ est la distance libre maximale possible pour un $(n, k, \delta)$-code convolutif binaire. Chaque entrée polynomiale de la matrice est codée en octal (ex. la valeur octale 113 représente le polynôme $X^6 \oplus X^3 \oplus X \oplus 1$).

| $m$ | $\Delta(2, 1, m)$ | Matrice canonique $G(X)$ |
|---|---|---|
| 0 | 2 | (1 1) |
| 1 | 3 | (1 3) |
| 2 | 5 | (5 7) |
| 3 | 6 | (13 17) |
| 4 | 7 | (23 35) |
| 5 | 8 | (53 75) |
| 6 | 10 | (133 171) |
| 7 | 10 | (133 171) |
| 8 | 12 | (561 753) |
| 9 | 12 | (561 753) |
| 10 | 14 | (2335 3661) |

TABLE B.1 – Exemples de $(2, 1, m)$-codes convolutifs optimaux

| $m$ | $\Delta(3,1,m)$ | Matrice canonique $G(X)$ |
|---|---|---|
| 0 | 3 | (1 1 1) |
| 1 | 5 | (1 3 3) |
| 2 | 8 | (5 7 7) |
| 3 | 10 | (13 15 17) |
| 4 | 12 | (25 33 37) |
| 5 | 13 | (47 53 75) |
| 6 | 15 | (133 145 175) |
| 7 | 16 | (225 331 367) |
| 8 | 18 | (567 663 711) |
| 9 | 20 | (1117 1365 1633) |
| 10 | 22 | (2353 2671 3175) |

TABLE B.2 – Exemples de $(3,1,m)$-codes convolutifs optimaux

| $m$ | $\Delta(3,2,m)$ | Matrice canonique $G(X)$ | Indices de Forney |
|---|---|---|---|
| 0 | 2 | $\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$ | (0,0) |
| 1 | 2 | " | " |
| 2 | 3 | $\begin{pmatrix} 3 & 2 & 3 \\ 2 & 1 & 1 \end{pmatrix}$ | (1,1) |
| 3 | 4 | $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 1 & 7 \end{pmatrix}$ | (1,2) |
| 4 | 5 | $\begin{pmatrix} 7 & 4 & 1 \\ 2 & 5 & 7 \end{pmatrix}$ | (2,2) |
| 5 | 6 | $\begin{pmatrix} 3 & 6 & 7 \\ 14 & 1 & 17 \end{pmatrix}$ | (2,3) |
| 6 | 7 | $\begin{pmatrix} 13 & 6 & 13 \\ 6 & 13 & 17 \end{pmatrix}$ | (3,3) |
| 7 | 8 | $\begin{pmatrix} 3 & 6 & 15 \\ 34 & 31 & 17 \end{pmatrix}$ | (3,4) |
| 8 | 8 | " " | |
| 9 | 9 | $\begin{pmatrix} 25 & 30 & 17 \\ 50 & 7 & 65 \end{pmatrix}$ | (4,5) |
| 10 | 10 | $\begin{pmatrix} 63 & 54 & 31 \\ 26 & 53 & 43 \end{pmatrix}$ | (5,5) |

TABLE B.3 – Exemples de $(3,2,m)$-codes convolutifs optimaux

| $m$ | $\Delta(4,1,m)$ | Matrice canonique $G(X)$ |
|---|---|---|
| 0 | 4 | (1 1 1 1) |
| 1 | 7 | (1 3 3 3) |
| 2 | 10 | (5 7 7 7) |
| 3 | 13 | (13 15 15 17) |
| 4 | 16 | (25 27 33 37) |
| 5 | 18 | (53 67 71 75) |
| 6 | 20 | (135 135 147 163) |
| 7 | 22 | (235 275 313 357) |
| 8 | 24 | (463 535 733 745) |
| 9 | 27 | (1117 1365 1633 1653) |
| 10 | 29 | (2387 2353 2671 3175) |

TABLE B.4 – Exemples de $(4,1,m)$-codes convolutifs optimaux

| $m$ | $\Delta(4,3,m)$ | Matrice canonique $G(X)$ | Indices de Forney |
|---|---|---|---|
| 0 | 2 | $\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$ | (0,0,0) |
| 1 | 2 | " | " |
| 2 | 3 | $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 \\ 0 & 3 & 1 & 0 \end{pmatrix}$ | (0,1,1) |
| 3 | 4 | $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 3 & 2 & 1 \\ 0 & 2 & 5 & 5 \end{pmatrix}$ | (0,1,2) |
| 4 | 4 | " | " |
| 5 | 5 | $\begin{pmatrix} 3 & 2 & 2 & 3 \\ 4 & 3 & 0 & 7 \\ 0 & 2 & 5 & 5 \end{pmatrix}$ | (1,2,2) |
| 6 | 6 | $\begin{pmatrix} 3 & 4 & 0 & 7 \\ 6 & 1 & 4 & 3 \\ 2 & 6 & 7 & 1 \end{pmatrix}$ | (2,2,2) |
| 7 | 6 | " | " |
| 8 | 7 | $\begin{pmatrix} 7 & 6 & 2 & 1 \\ 14 & 5 & 0 & 15 \\ 10 & 4 & 17 & 1 \end{pmatrix}$ | (2,3,3) |
| 9 | 8 | $\begin{pmatrix} 1 & 14 & 16 & 3 \\ 10 & 13 & 2 & 7 \\ 16 & 0 & 3 & 13 \end{pmatrix}$ | (3,3,3) |

TABLE B.5 – Exemples de $(4,3,m)$-codes convolutifs optimaux

# Annexe C

# Tables de codes convolutifs poinçonnés

Cette annexe regroupe les différents résultats issus de la recherche exhaustive de bons codes convolutifs poinçonnés . Ces travaux ont été initiés par J.B. Cain [17] et *al.* et développés par Y. Yasuda et *al.* et J. Hagenauer [86] pour des codes de faible mémoire de contrainte ($K \leq 9$). Ces résultats ont été plus largement développés par D. Haccoun et G. Bégin et étendus à des mémoires plus grandes ($K \leq 24$) [6, 84].

Chacune des tables donne les générateurs $G_1$ et $G_2$ du code convolutif parent (de faible taux) en notation octale, la matrice de poinçonnage $P$ et la distance libre $d$ du code poinçonné obtenu. Ces codes sont les meilleurs pour le décodage par Viterbi et le décodage séquentiel.

| Code parent | | Code poinçonné | | Code parent | | Code poinçonné | |
|---|---|---|---|---|---|---|---|
| M | $G_1$ $G_2$ | P | d | M | $G_1$ $G_2$ | P | d |
| 2 | 5 - 7 | 1 0<br>1 1 | 3 | 3 | 15 - 17 | 1 1<br>1 0 | 4 |
| 4 | 23 - 35 | 1 1<br>1 0 | 4 | 5 | 53 - 75 | 1 0<br>1 1 | 6 |
| 6 | 133 - 171 | 1 1<br>1 0 | 6 | 7 | 247 - 371 | 1 0<br>1 1 | 7 |
| 8 | 561 - 753 | 1 1<br>1 0 | 7 | 9 | 1167 - 1545 | 1 1<br>1 0 | 7 |
| 10 | 2335 - 3661 | 1 0<br>1 1 | 8 | 11 | 4335 - 5723 | 1 1<br>1 0 | 9 |
| 12 | 10533 - 17661 | 1 1<br>1 0 | 9 | 13 | 21675 - 27123 | 1 1<br>1 0 | 10 |
| 14 | 55367 - 63121 | 1 1<br>1 0 | 10 | 15 | 111653 - 145665 | 1 1<br>1 0 | 10 |
| 16 | 347241 - 246277 | 1 1<br>1 0 | 12 | 17 | 506477 - 673711 | 1 0<br>1 1 | 12 |
| 18 | 1352755 - 1771563 | 1 1<br>1 0 | 12 | 19 | 2451321 - 3546713 | 1 1<br>1 0 | 12 |
| 19 | 2142513 - 3276177 | 1 1<br>1 0 | 13 | 20 | 6567413 - 5322305 | 1 1<br>1 0 | 12 |
| 21 | 15724153 - 12076311 | 1 1<br>1 0 | 13 | 22 | 33455341 - 24247063 | 1 1<br>1 0 | 14 |
| 23 | 55076157 - 75501351 | 1 1<br>1 0 | 15 | | | | |

TABLE C.1 – Codes poinçonnés de taux $\frac{2}{3}$

| Code parent | | Code poinçonné | | Code parent | | Code poinçonné | |
|---|---|---|---|---|---|---|---|
| M | $G_1$ $G_2$ | P | d | M | $G_1$ $G_2$ | P | d |
| 2 | 5 - 7 | 1 0 1<br>1 1 0 | 3 | 3 | 15 - 17 | 1 1 0<br>1 0 1 | 4 |
| 4 | 23 - 35 | 1 0 1<br>1 1 0 | 3 | 5 | 53 - 75 | 1 0 0<br>1 1 1 | 4 |
| 6 | 133 - 171 | 1 1 0<br>1 0 1 | 5 | 7 | 247 - 371 | 1 1 0<br>1 0 1 | 6 |
| 8 | 561 - 753 | 1 1 1<br>1 0 0 | 6 | 9 | 1167 - 1545 | 1 0 0<br>1 1 1 | 6 |
| 10 | 2335 - 3661 | 1 0 1<br>1 1 0 | 6 | 11 | 4335 - 5723 | 1 0 0<br>1 1 1 | 7 |
| 12 | 10533 - 17661 | 1 1 0<br>1 0 1 | 7 | 13 | 21675 - 27123 | 1 1 0<br>1 0 1 | 7 |
| 14 | 55367 - 63121 | 1 0 1<br>1 1 0 | 8 | 15 | 111653 - 145665 | 1 0 0<br>1 1 1 | 8 |
| 16 | 347241 - 246277 | 1 1 0<br>1 0 1 | 8 | 17 | 506477 - 673711 | 1 0 0<br>1 1 1 | 9 |
| 18 | 1352755 - 1771563 | 1 1 1<br>1 0 0 | 10 | 19 | 2451321 - 3546713 | 1 1 0<br>1 0 1 | 10 |
| 19 | 2142513 - 3276177 | 1 1 0<br>1 0 1 | 10 | 20 | 6567413 - 5322305 | 1 1 1<br>1 0 0 | 10 |
| 21 | 15724153 - 12076311 | 1 1 1<br>1 0 0 | 11 | 22 | 33455341 - 24247063 | 1 0 0<br>1 1 1 | 12 |
| 23 | 55076157 - 75501351 | 1 0 0<br>1 1 1 | 13 | | | | |

TABLE C.2 – Codes poinçonnés de taux $\frac{3}{4}$

| Code parent | | Code poinçonné | | Code parent | | Code poinçonné | |
|---|---|---|---|---|---|---|---|
| M | $G_1$ $G_2$ | P | d | M | $G_1$ $G_2$ | P | d |
| 2 | 5 - 7 | 1 0 1 1<br>1 1 0 0 | 2 | 3 | 15 - 17 | 1 0 1 1<br>1 1 0 0 | 3 |
| 4 | 23 - 35 | 1 0 1 0<br>1 1 0 1 | 3 | 5 | 53 - 75 | 1 0 0 0<br>1 1 1 1 | 4 |
| 6 | 133 - 171 | 1 1 1 1<br>1 0 0 0 | 4 | 7 | 247 - 371 | 1 0 1 0<br>1 1 0 1 | 5 |
| 8 | 561 - 753 | 1 1 0 1<br>1 0 1 0 | 5 | 9 | 1167 - 1545 | 1 1 1 1<br>1 0 0 0 | 4 |
| 10 | 2335 - 3661 | 1 0 0 1<br>1 1 1 0 | 5 | 11 | 4335 - 5723 | 1 0 1 1<br>1 1 0 0 | 6 |
| 12 | 10533 - 17661 | 1 0 1 1<br>1 1 0 0 | 6 | 13 | 21675 - 27123 | 1 0 1 1<br>1 1 0 0 | 7 |
| 14 | 55367 - 63121 | 1 1 1 0<br>1 0 0 1 | 7 | 15 | 111653 - 145665 | 1 0 1 0<br>1 1 0 1 | 8 |
| 16 | 347241 - 246277 | 1 0 0 0<br>1 1 1 1 | 8 | 17 | 506477 - 673711 | 1 1 0 1<br>1 0 1 0 | 8 |
| 18 | 1352755 - 1771563 | 1 0 1 1<br>1 1 0 0 | 8 | 19 | 2451321 - 3546713 | 1 1 1 0<br>1 0 0 1 | 9 |
| 19 | 2142513 - 3276177 | 1 0 1 1<br>1 1 0 0 | 9 | | | | |

TABLE C.3 – Codes poinçonnés de taux $\frac{4}{5}$

| Code parent | | Code poinçonné | | Code parent | | Code poinçonné | |
|---|---|---|---|---|---|---|---|
| M | $G_1$ $G_2$ | P | d | M | $G_1$ $G_2$ | P | d |
| 2 | 5 - 7 | 1 0 1 1 1<br>1 1 0 0 0 | 2 | 3 | 15 - 17 | 1 0 1 0 0<br>1 1 0 1 1 | 3 |
| 4 | 23 - 35 | 1 0 1 1 1<br>1 1 0 0 0 | 3 | 5 | 53 - 75 | 1 0 0 0 0<br>1 1 1 1 1 | 4 |
| 6 | 133 - 171 | 1 1 0 1 0<br>1 0 1 0 1 | 4 | 7 | 247 - 371 | 1 1 1 0 0<br>1 0 1 1 0 | 4 |
| 8 | 561 - 753 | 1 0 1 1 0<br>1 1 0 0 1 | 5 | 9 | 1167 - 1545 | 1 0 1 1 1<br>1 1 0 0 0 | 5 |
| 10 | 2335 - 3661 | 1 1 0 0 0<br>1 0 1 1 1 | 5 | 11 | 4335 - 5723 | 1 1 0 0 0<br>1 0 1 1 1 | 5 |
| 12 | 10533 - 17661 | 1 0 0 1 1<br>1 1 1 0 0 | 6 | 13 | 21675 - 27123 | 1 1 0 1 0<br>1 0 1 0 1 | 6 |
| 14 | 55367 - 63121 | 1 1 1 1 1<br>1 0 0 0 0 | 6 | 15 | 111653 - 145665 | 1 0 1 0 1<br>1 1 0 1 0 | 7 |
| 16 | 347241 - 246277 | 1 1 0 0 0<br>1 0 1 1 1 | 7 | 17 | 506477 - 673711 | 1 0 0 0 0<br>1 1 1 1 1 | 7 |
| 18 | 1352755 - 1771563 | 1 1 1 1 1<br>1 0 0 0 0 | 8 | 19 | 2451321 - 3546713 | 1 1 0 1 0<br>1 0 1 0 1 | 8 |
| 19 | 2142513 - 3276177 | 1 1 1 1 0<br>1 0 0 0 1 | 8 | | | | |

TABLE C.4 – Codes poinçonnés de taux $\frac{5}{6}$

| Code parent | | Code poinçonné | | Code parent | | Code poinçonné | |
|---|---|---|---|---|---|---|---|
| M | $G_1$ $G_2$ | P | d | M | $G_1$ $G_2$ | P | d |
| 2 | 5 - 7 | 1 0 1 1 1 1<br>1 1 0 0 0 0 | 2 | 3 | 15 - 17 | 1 0 0 0 1 1<br>1 1 1 1 0 0 | 2 |
| 4 | 23 - 35 | 1 0 1 0 1 0<br>1 1 0 1 0 1 | 3 | 5 | 53 - 75 | 1 1 0 1 1 0<br>1 0 1 0 0 1 | 3 |
| 6 | 133 - 171 | 1 1 1 0 1 0<br>1 0 0 1 0 1 | 3 | 7 | 247 - 371 | 1 0 1 0 0 1<br>1 1 0 1 1 0 | 4 |
| 8 | 561 - 753 | 1 1 0 1 1 0<br>1 0 1 0 0 1 | 4 | 9 | 1167 - 1545 | 1 1 1 1 0 0<br>1 0 0 0 1 1 | 5 |
| 10 | 2335 - 3661 | 1 1 1 1 0 0<br>1 0 0 0 1 1 | 5 | 11 | 4335 - 5723 | 1 1 0 1 0 1<br>1 0 1 0 1 0 | 5 |
| 12 | 10533 - 17661 | 1 1 1 0 1 1<br>1 0 0 1 0 0 | 6 | 13 | 21675 - 27123 | 1 1 0 0 0 0<br>1 0 1 1 1 1 | 6 |
| 14 | 55367 - 63121 | 1 1 1 1 0 1<br>1 0 0 0 1 0 | 6 | 15 | 111653 - 145665 | 1 0 1 0 0 0<br>1 1 0 1 1 1 | 6 |
| 16 | 347241 - 246277 | 1 1 1 1 0 1<br>1 0 0 0 1 0 | 7 | 17 | 506477 - 673711 | 1 0 1 1 0 1<br>1 1 0 0 1 0 | 7 |
| 18 | 1352755 - 1771563 | 1 1 1 1 1 1<br>1 0 0 0 0 0 | 7 | 19 | 2451321 - 3546713 | 1 0 1 1 1 0<br>1 1 0 0 0 1 | 7 |
| 19 | 2142513 - 3276177 | 1 1 0 1 1 0<br>1 1 0 0 0 1 | 7 | | | | |

TABLE C.5 – Codes poinçonnés de taux $\frac{6}{7}$

| Code parent | | Code poinçonné | | Code parent | | Code poinçonné | |
|---|---|---|---|---|---|---|---|
| M | $G_1$ $G_2$ | P | d | M | $G_1$ $G_2$ | P | d |
| 2 | 5 - 7 | 1 0 1 1 1 1 1<br>1 1 0 0 0 0 0 | 2 | 3 | 15 - 17 | 1 0 0 0 0 1 0<br>1 1 1 1 1 0 1 | 2 |
| 4 | 23 - 35 | 1 0 1 0 0 1 1<br>1 1 0 1 1 0 0 | 3 | 5 | 53 - 75 | 1 0 1 1 1 0 1<br>1 1 0 0 0 1 0 | 3 |
| 6 | 133 - 171 | 1 1 1 1 0 1 0<br>1 0 0 0 1 0 1 | 3 | 7 | 247 - 371 | 1 0 1 0 1 0 0<br>1 1 0 1 0 1 1 | 4 |
| 8 | 561 - 753 | 1 1 0 1 0 1 1<br>1 0 1 0 1 0 0 | 4 | 9 | 1167 - 1545 | 1 0 1 0 0 1 1<br>1 1 0 1 1 0 0 | 4 |
| 10 | 2335 - 3661 | 1 0 1 0 1 1 1<br>1 1 0 1 0 0 0 | 4 | 11 | 4335 - 5723 | 1 0 0 1 1 0 1<br>1 1 1 0 0 1 0 | 5 |
| 12 | 10533 - 17661 | 1 1 0 1 0 0 1<br>1 0 1 0 1 1 0 | 5 | 13 | 21675 - 27123 | 1 0 1 1 0 0 1<br>1 1 0 0 1 1 0 | 5 |
| 14 | 55367 - 63121 | 1 0 0 0 0 0 0<br>1 1 1 1 1 1 1 | 6 | 15 | 111653 - 145665 | 1 0 0 0 0 1 1<br>1 1 1 1 1 0 0 | 6 |
| 16 | 347241 - 246277 | 1 1 0 1 0 0 1<br>1 0 1 0 1 1 0 | 6 | 17 | 506477 - 673711 | 1 0 1 0 1 0 0<br>1 1 0 1 0 1 1 | 6 |
| 18 | 1352755 - 1771563 | 1 1 0 1 1 0 1<br>1 0 1 0 0 1 0 | 7 | 19 | 2451321 - 3546713 | 1 1 0 0 0 1 0<br>1 0 1 1 1 0 1 | 7 |
| 19 | 2142513 - 3276177 | 1 1 0 1 1 0 1<br>1 0 1 0 0 1 0 | 7 | | | | |

TABLE C.6 – Codes poinçonnés de taux $\frac{7}{8}$

# Bibliographie


[1] http ://www.nist.org/aes/

[2] ARTIN (E.) .- *Algebraic Numbers and Algebraic Functions*. Gordon and Breach, New York, 1969.

[3] ASSMUS (E.) et KEY (J.D.) .- *Designs and their Codes*. Cambridge University Press, 1992.

[4] The BALTIMORE SUN : *No Such Agency, Part 4 : Rigging the Game*, n$^o$ du 4 décembre 1995.

[5] BEGIN (G.) et HACCOUN (D.) .- High-rate punctured convolutional codes : structure properties and construction techniques. *IEEE Trans. on Communications*, COMM-37, pp 1381-1385, Dec. 1989.

[6] BÉGIN (G.), HACCOUN (D.) et PAQUIN (C.) .- Further results on high-rate punctured convolutional codes for Viterbi and sequential decoding. *IEEE Transactions on Communications*, Vol. 38, No. 11, Novembre 1990.

[7] BERLEKAMP (E.R.) .- *Algebraic Coding Theory*. McGraw-Hill, New-York, 1968.

[8] BETH (T.), JUNGNICKEL (D.) et LENZ) (H.) .- *Design Theory*. Cambridge University Press, Cambridge, 1986.

[9] BEUTELSPACHER (A.) .- *Classical Geometries*. In Handbook of Combinatorial Designs, C. J. Colbourn, J. H. Dinitz eds., CRC Press, 1996.

[10] BIHAM (E.) et SHAMIR (A.) .- Differential cryptanalysis of DES-like cryptosystems. In : *Advances in Cryptology - CRYPTO'90*, Lecture Notes in Computer Science 537, pp 2–21, Springer Verlag, 1991.

[11] BIHAM (E.) et SHAMIR (A.) .- Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, Vol. 4, Nr 1, pp 3–72, 1991.

[12] BIHAM (E.) et SHAMIR (A.) .- Differential cryptanalysis of the full 16-round DES. In : *Advances in Cryptology - CRYPTO'92*, Lecture Notes in Computer Science 740, pp 487–496, Springer Verlag, 1992.

[13] BIHAM (E.) et SHAMIR (A.) .- *Differential Cryptanalysis of the Data Encryption Standard*. Springer Verlag, 1993.

[14] BLAHUT (R.E.) .- *Decoding of cyclic codes and codes on curves*. In Handbook of Coding Theory, V. S. Pless and W. C. Huffman Editors, North-Holland, 1998.

[15] BRASSARD (G.) .- *Cryptologie comtemporaine*.- Masson, 1993, Coll. *Logique, Mathématiques et Informatique*.

[16] BRUER (J.) .- On pseudo-random sequences as crypto-generators. *Proceedings of the International Zurich Seminar on Digital Communications*, pp 157–161, 1984.

[17] CAIN (J.B.), CLARK (G.C.) et GEIST (J.M.) .- Punctured convolutional codes of Rate $\frac{n-1}{n}$ and simplified maximum likekihood decoding. *IEEE Transactions on Information Theory*, Vol. IT-25, No. 1, Janvier 1979.

[18] CAMERON (P.J.) .- *Combinatorics : Topics, Techniques, Algorithms*. Cambridge University Press, 1996.

[19] CAMERON (P.J.) et VANLINT (J.H.) .- *Designs, Graphs, Codes and their Links*. London Mathematical Society, Students Text 22, Cambridge University Press, 1991.

[20] CAMION (P.), CARLET (C.), CHARPIN (P.) et SENDRIER (N.) .- On correlation-immune functions. In : *Advances in Cryptology - CRYPTO'91*, Lecture Notes in Computer Science 576, Springer Verlag, 1992.

[21] CANTEAUT (A.) .- *Attaques de cryptosystèmes à mots de poids faibles et construction de fonctions t-résilientes*. Thèse de Doctorat, Université Paris VI, 1996.

[22] CANTEAUT (A.), CARLET (C.), CHARPIN (P.) et FONTAINE (C.) .- On cryptographic properties of the cosets of R(1,m). A paraitre dans *IEEE Transactions on Information Theory*.

[23] CANTEAUT (A.) et FILIOL (E.) .- Ciphertext Only Reconstruction of Stream Ciphers based on Combination Generators. In : *Proceedings of Fast Software Encryption 2000*, éd. par SCHNEIER (B.), Lecture Notes in Computer Science 1978, Springer Verlag, 2001.

[24] CANTEAUT (A.) et FILIOL (E.) .- Ciphertext Only Reconstruction of LFSR-based Stream Ciphers. *Rapport de Recherche INRIA*, n$^o$ 3887, Février 2000 - Institut National de Recherche en Informatique et Automatique, Domaine de Voluceau, B.P. 105, 78153 Le Chesnay cédex.

[25] CANTEAUT (A.) et TRABBIA (M.) .- Improved Fast Correlation Attacks using Parity-check Equations of weight 4 and 5. In : *Advances in Cryptology - EUROCRYPT'2000*, Lecture Notes in Computer Science 1807, Springer Verlag, 2000.

[26] CARLET (C.) .- Partially Bent Functions. *Designs, Codes and Cryptography*, Vol. 3, 135-145, 1993.

[27] CARLET (C.) .- On the propagation criterion of degree $l$ and order $k$. In : *Advances in Cryptology - EUROCRYPT'98*, Lecture Notes in Computer Science 1403, Springer Verlag, 1998.

[28] CARLET (C.) .- Generalized Partial Spreads. *IEEE Transactions on Information Theory*, Vol. IT-41, Nr 5, pp 1482–1487, 1995.

[29] CARLET (C.) .- Two New Classes of Bent Functions. In : *Advances in Cryptology - EUROCRYPT'93*, Lecture Notes in Computer Science 765, pp 77–101, Springer Verlag, 1994.

[30] CARLET (C.) .- A Construction of Bent Functions. In : *Third Conference on Finite Fields and Applications, Glasgow, Grande-Bretagne*, London Mathematical Society, Lecture Series 233, Cambridge University Press, pp 47–58, 1996.

[31] CARLET (C.) .- Hyper-Bent Functions. *PRAGoCRYPT'96 Conference*, Czech Technical University Publishing House, pp 145–155, Prague, 1996.

[32] CARLET (C.) .- Recent Results on Bent Functions. *International Conference on Combinatorics, Information Theory and Statistics'97*, 1998.

[33] CARLET (C.) et DUBUC (S.) .- On Generalized Bent and q-ary Perfect Nonlinear Functions. In : *Fifth Conference on Finite Fields and Applications*, à paraitre, 2000.

[34] CARLET (C.) et GUILLOT (P.) .- A Characterization of Binary Bent Functions. *Journal of Combinatorial Theory, Series A*, Vol. 76, Nr. 2, pp 328–335, 1996.

[35] CARLET (C.) et GUILLOT (P.) .- An Alternate Characterization of the Bentness of Binary Functions with Uniqueness. *Designs, Codes and Cryptography*, Vol. 14, Nr 2, pp 133–140, Mai 1998.

[36] CARLET (C.) et GUILLOT (P.) .- A new Representation of Boolean Functions. In : *Proceedings of AAECC'13*, Lecture Notes in Computer Science 1719, Springer Verlag, 1999.

[37] CARLET (C.) et GUILLOT (P.) .- Bent, Resilient Functions and the Numerical Normal Form. A paraitre dans *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*.

[38] CHAKRABARTY (K.) et HAYES (J.P.) .- Balanced Boolean Functions. *IEE Proc.-Comput. Digit. Tech.*, Vol. 145, No. 1, January 1998.

[39] CHAKRABARTY (K.) et HAYES (J.P.) .- Balance testing and balance-testable design of logic circuits. *J. Electron. Testing : Theory Appl.*, 1996, 8, pp 81–86.

[40] CHAKRABARTY (K.) et HAYES (J.P.) .- Cumulative balance testing of logic circuits.*IEEE Trans. VLSI Syst.*, 1995, 3, pp 72–83.

[41] CHEPYZHOV (V.), JOHANSSON (T.) et SMEETS (B.) .- A Simple Algorithm for Fast Correlation Attacks on Stream Ciphers. In :*Proceedings of Fast Software Encryption 2000*, éd. par SCHNEIER (B.), à paraitre dans Lecture Notes in Computer Science, Springer Verlag.

[42] CHEPYZHOV (V.) et SMEETS (B.) .- On a Fast Correlation Attack on Stream Ciphers. In : *Advances in Cryptology - EUROCRYPT'91*, Lecture Notes in Computer Science 547, Springer Verlag, 1991.

[43] COHEN (G.D.), GODLEWSKI (P.) et MERKX (F.) .- Linear binary codes for write-once memories. *IEEE Transactions in Information Theory*, Vol. IT-32, pp 697–700, 1986.

[44] COLBURN (C.J.) et DINITZ (J.H.) éditeurs .- *The CRC Handbook of Combinatorial Designs*. CRC Press, 1996.

[45] http ://www-rocq.inria.fr/codes/Eric.Filiol/English/COS/COS.html

[46] Data Encryption Standard, *National Bureau of Standards, Federal INformation Processing Standard*, Vol. 46, U.S.A., Janvier 1977.

[47] DIFFIE (W.) et HELLMAN (M.E.) .- New directions in cryptography. *IEEE Transactions in Information Theory*, Vol. IT-22, Nr. 6, 1976, pp. 644–654.

[48] DILLON (J.F.) .- *Elementary Hadamard Difference Sets*. Thèse de Doctorat, Uiversité of Maryland, 1974.

[49] DHOLAKIA (A.) .- *Introduction to Convolutional Codes with Applications*. Kluwer, Boston, 1994.

[50] DOBBERTIN (H.) .- Construction of Bent Functions and Balanced Boolean Functions with High Nonlinearity. *Fast Software Encryption*, Lecture Notes in Computer Sciences 1008, Springer Verlag, 1994.

[51] DOBBERTIN (H.), BOSSELAERS (A.) et PRENEEL (B.) .- Ripemd-160 : a Strengthened Version of RIPEMD. In : *Fast Software Encryption 1995* Lecture Notes in Computer Sciences 1039, Springer Verlag, 1996.

[52] ELIAS (P.) .- Coding for noisy channels. *IRE Conv. Rec.*, Part. 4, pp. 37-46, 1955.

[53] FEISTEL (H.) .- Cryptography and computer privacy. *Scientific American*, Vol. 228, Nr. 5, pp 15–23, 1973.

[54] FELLER (W.) .- *An Introduction to Probability Theory.* Vol1., Wiley, 1966.

[55] FIAT (A.) et SHAMIR (A.) .- How to prove yourself : practical solutions to identification and signature problems. In : *Advances in Cryptology - CRYPTO'86*, Lecture Notes in Computer Science 263, pp. 186–194, Springer Verlag, 1986.

[56] http ://www-rocq.inria.fr/codes/Eric.Filiol/Attaques/index.html

[57] FILIOL (E.) .- Reconstruction of Convolutional Encoders over $GF(q)$. In : *Proceedings of the 6th IMA Conference on Cryptography and Coding*, éd. par DARNELL (M.), Lecture Notes in Computer Science, n$^o$ 1355, pp. 100-110. - Springer Verlag, 1997.

[58] FILIOL (E.) et FONTAINE (C.) .- Highly Nonlinear Balanced Boolean Functions with a good Correlation Immunity. In : *Advances in Cryptology - EUROCRYPT'98*, Lecture Notes in Computer Science, n$^o$ 1403, pp. 475-488. - Springer Verlag, 1998.

[59] FILIOL (E.) .- Designs, Intersecting Families and Weight of Boolean Functions. In : *Proceedings of the 7th IMA Conference on Cryptography and Coding*, éd. par WALKER (M.), Lecture Notes in Computer Science, n$^o$ 1746, pp. 70-81. - Springer Verlag, 1999.

[60] FILIOL (E.) .- Designs, Intersecting Families and Weight of Boolean Functions : Construction and Characterization .- Rapport technique CRECSC - Décembre 1999.

[61] FILIOL (E.) .- Reconstruction of Punctured Convolutional Encoders. In : Proceedings of the 2000 International Symposium on Information Theory and Applications, éd. par FUJIWARA (T.), SITA et IEICE Publishing, pp. 4–7, 2000.

[62] FILIOL (E.) .- Decimation Attack of Stream Ciphers. In : *Proceedings of the First International Conference in India - INDOCRYPT'2000*, Lecture Notes in Computer Science 1977, pp 31–42, Springer Verlag, 2000.

[63] FIPS 180-1 .- Secure Hash Standard. Federal Information Processing Standards Publication 180-1, U.S. Dept. of Commerce/NIST, National Technical Information Service, Springfield, Virginia, April 17, 1995.

[64] FISHER (Sir R.A.) .- An examination of the different possible solutions of a problem in incomplete blocks. *Ann. Eugenics*, Vol. 10, pp 52–75, 1940.

[65] FONTAINE (C.) .- *Contribution à la recherche de fonctions booléennes hautement non linéaires, et au marquage d'images en vue de la protection des droits d'auteur*. Thèse de Doctorat, Université Paris VI, 1998.

[66] FORNEY G.D.) .- Convolutional codes I : Algebraic structure. *IEEE Transactions on Information Theory*, IT-16, pp. 720–738, Nov. 1970.

[67] FORNEY G.D.) .- *Algebraic Structure of Convolutional Codes and Algebraic System Theory*. Mathematical System Theory, A.C. Antoulas editor, Springer Verlag, Berlin, 1991.

[68] FORRE (R.) .- A fast correlation attack on nonlinearly feedforward filtered shift-register sequences. In : *Advances in Cryptology - EUROCRYPT'89*, Lectures Notes in Computer Science 434, pp 586–595, Springer Verlag, 1990.

[69] FRANKL (P.) .- *Extremal Set Systems*. In Handbook of Combinatorics, R. L. Graham, M. Grötschel and L. Lovász (eds), Elsevier, 1995.

[70] GALLAGER (R.G.) .- Low-density parity-check codes. *IRE Trans. Inform. Theory*, Vol. IT-8 :21-28, 1962.

[71] GEFFE (P.R.) .- How to protect Data with Ciphers that are Really Hard to Break. *Electronics*, pp 99–101, 1973.

[72] Colonel GIVIERGE (M.) .- *Cours de Cryptographie*. Berger-Levrault, Paris, 1925.

[73] GOLDWASSER (S.), MICALI (S.) et RACKOFF (C.) .- The knowledge complexity on interactive proof systems. In : *Proceedings of the 14th ACM Symposium on Theory of Computing*, pp 291–304, 1985.

[74] GOLIĆ (J.) .- On the linear complexity of functions of periodic GF(q) sequences. *IEEE Transactions on Information Theory*, Vol. IT-35, Nr 1, pp 69–75, 1989.

[75] GOLIĆ (J.) .- On the security of nonlinear filter generators. In ; *Fast Software Encryption, Third International Workshop*, Lecture Notes in Computer Science 1039, pp 173–188, Springer Verlag, 1996.

[76] GOLOMB (S.W.) .- *Shift Register Sequences.*Aegean Park Press, 1982.

[77] GOLOMB (S.W.) .- On the Cryptanalysis of Nonlinear Sequences. In : *Proceedings of the 7th IMA Conference on Cryptography and Coding*, éd. par WALKER (M.), Lecture Notes in Computer Science, n$^o$ 1746, pp. 236-242. - Springer Verlag, 1999.

[78] GONG (G.) et GOLOMB (S.W.) .- Transform Domain Analysis of DES. *IEEE Transactions on Information Theory*, vol. 45, No.6, September 1999, pp. 2065-2073.

[79] GOPALAKRISHNAN (K.) et STINSON (D.R.) .- Three characterizations of non-binary correlation-immune and resilient functions. *Designs, Codes and Cryptography*, Vol. 5, pp 241–251, 1995.

[80] GÖTTFERT (R.) et NIDERREITTER (H.) .- On the Minimal Polynomial of the Product of Linear Recurring Sequences. *Finite Fields and Their Applications*, 1(2), pp 204–218, 1995.

[81] GRABBE (J. O.) .- NSA, Crypto AG and the Iraq-Iran Conflict. In : *http ://www.cryptosoft.com/snews/nov97/03119700.htm.*

[82] GUILLOT (P.) .- *Fonctions courbes binaires et transformation de Möbius.* Thèse de Doctorat, Université de Caen, 1999.

[83] GUILLOT (P.) .- The Completed Class of $\mathcal{GPS}$ Covers all Bent Functions. Preprint. Paper presented at ISIT'98, Cambridge, USA, 1998.

[84] HACCOUN (D.) et BÉGIN (G.) .- High-rate punctured convolutional codes for Viterbi and sequential decoding. *IEEE Transactions on Communications*, Vol. 37, No. 11, Novembre 1989.

[85] HAGELBARGER (D.W.) .- Recurrent codes : Easily mechanized, burst-correcting binary codes. *Bell. Syst. Tech. J.*, Vol. 38, pp 969–984, July 1959.

[86] HAGENAUER (J.) .- Rate compatible punctured convolutional codes and their applications. *IEEE Transactions on Communications*, Vol. 36, Avril 1988.

[87] HAWKES (P.) et O'CONNOR (L.) .- Xor and non-xor differential probabilities. In : *Advances in Cryptology - EUROCRYPT'99*, Lectures Notes in Computer Science 1592, Springer Verlag, 1999.

[88] HERLESTAM (T.) .- On the complexity of functions of linear shift register sequences. In : *IEEE Symposium on Information Theory*, 1982.

[89] HERLESTAM (T.) .- On the complexity of certain crypto generators. In : *Security, IFIP/Sec'83*, North Holland, 1983.

[90] HERLESTAM (T.) .- On Functions of Linear Shift Register Sequences. In : *Advances in Cryptology - EUROCRYPT'85*, Lecture Notes in Computer Science 219 pp 119–129, Springer Verlag, 1986.

[91] HIRSHFELD (J.W.P.) .- *Projective Geometries over Finite Fields*. Oxford University Press, 1979.

[92] D. G. HOFFMAN (D.G.) and al. .- *Coding Theory - The Essentials*. Dekker, 1991.

[93] HOU (X.D.) .- On the Coefficients of Binary Bent Functions. In : *Proceedings of the American Mathematical Society*, S 0002-9939(99)05146-1. Publié électroniquement le 17 août 1999.

[94] HOU (X.D.) et LANGEVIN (P.) .- Results on Bent Functions. *Journal of Combinatorial Theory, Series A*, Vol. 80, pp 232–246, 1997.

[95] JANSEN (C.J.A.) .- *Investigations on Nonlinear Stream Ciphers Systems : Construction and Evaluation Methods*. Thèse de Doctorat, Technische Universiteit Delft, 1989.

[96] JENNINGS (S.M.) .- Multiplexed sequences : some properties on the minimum polynomial. *Cryptography Proceedings of the Workshop on Cryptography, Burg Feuerstein*, Lecture Notes in Computer Science 149, pp 189–206, Springer Verlag, 1983.

[97] JIAXI (L.) .- *Coolected Works on Combinatorial Designs*. Inner Mongolia People's Press, Hunhot, Mongolia, 1990.

[98] JOHANNESSON (R.) et ZYGANGIROV (K. Sh.) .- *Fundamentals of Convolutional Coding*, - IEEE Press, 1999.

[99] JOHN (P.W.M.) .- *Experimental Designs*. In Studies in Statistics 19, R.V. Hogg ed., Mathematical Association of America, 1978.

[100] JOHANSSON (T.) et JÖNSSON (F.) .- Improved Fast Correlation Attack on stream Ciphers via Convolutional Codes. In : *Advances in Cryptology - EUROCRYPT'99*, Lecture Notes in Computer Science 1592, pp 347–362, Springer Verlag, 1999.

[101] JOHANSSON (T.) et JÖNSSON (F.) .- Fast Correlation Attack based on Turbo Codes Techniques. In : *Advances in Cryptology - CRYPTO'99*, Lecture Notes in Computer Science 1666, pp 181–197, Springer Verlag, 1999

[102] KAHN (D.) .- *The Codebreakers : the Story of Secret Writing* - MacMillan Publishing, 1967.

[103] KERCKHOFFS (A.) .- *La cryptologie militaire*. Louis BAUDOIN éd., Paris, 1883.

[104] KEY (E.) .- An analysis of the structure and complexity of nonlinear binary sequence generators. *IEEE Transactions on Information Theory*, IT-22, pp 732–736, 1976.

[105] KIRKMAN (T.P.) .- On a problem in combinatorics. *Cambridge and Dublin Maths Journal*, Vol. 2, pp 191-204, 1850.

[106] LAI (X.) .- Condition for the nonsingularity of a feedback shift register over a general finite field. *IEEE Transactions on Information Theory*, Vol IT-33, pp 747–749, 1987.

[107] LAI (X.), MASSEY (J.L.) et MURPHY (S.) .- Markov ciphers and differential cryptanalysis. In : *Advances in Cryptology - EUROCRYPT'91*, Lecture Notes in Computer Science 547, pp 17–38, Springer Verlag, 1991.

[108] LARSEN (K.J.) .- Short convolutional codes with maximal free distance for rates $\frac{1}{2}$, $\frac{1}{3}$ and $\frac{1}{4}$. *IEEE Transactions on Information Theory*, IT-18, pp. 371–372, May 1973.

[109] LAUER (G.S.) .- Some optimal partial-unit-memory codes. *IEEE Transactions on Information Theory*, IT-25, pp. 240-243, March 1979.

[110] LIDL (R.) et NIDERREITER (H.) .- *Finite Fields*. Encyclopedia of Mathematics and its Applications, Cambridge University Press, 1997.

[111] LIDL (R.) et NIDERREITER (H.) .- *Introduction to Finite Fields and their Applications*. Cambridge University Press, 1994.

[112] LIN (S.) et COSTELLO (D.J.) .- *Error Control Coding : Fundamentals and Applications* - Prentice Hall, Englewood Cliffs, N.J., 1983.

[113] LUCAS (E.) .- *Sur les congruences des nombres eulériennes et des coefficients différentiels des fonctions trigonométriques, suivant un module premier*. Bull. Soc. Math. France, 6, 49–54, 1878.

[114] MACWILLIAMS (F.J.) - SLOANE (N.J.A.) .- *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, 1977.

[115] MCELIECE (R.J.) .- *The algebraic theory of convolutional codes*. In Handbook of Coding Theory, V.S. Pless and W.C. Huffman editors, North-Holland, 1998.

[116] MADSEN (W.) .- Crypto AG : The NSA's Trojan Whore ?
*http ://www.mediafilter.org/caq/crytogate*.

[117] MAITRA (S.) et SARKAR (P.) .- Constructions of non linear Boolean functions with important cryptographic properties. In : *Advances in Cryptology - EUROCRYPT'00*, Lecture Notes in Computer Science 1807, Springer Verlag, 2000.

[118] MAITRA (S.) et SARKAR (P.) .- New directions in design of resilient Boolean functions. *Advances in Cryptology - CRYPTO'00*, Lecture Notes in Computer Science 1880, 2000.

[119] MANTEL (W.) .- Residues of recurring series. *Nieuw. Arch. Wisk.*, Vol. 2, Nr. 1, pp 172–184, 1894.

[120] MASSEY (J.L.) .- *Threshold Decoding* - MIT Press, Cambridge Mass., 1963.

[121] MASSEY (J.L.) .- Codes, automata and continuous systems : explicit interconnections. *IEEE Trans. Automatic Control*, **AC-12**, pp644-649, 1967.

[122] MASSEY (J.L.) .- Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, Vol. IT-15, pp 122–127, 1969.

[123] MASSEY (J.L.) .- Some applications of coding theory in cryptography. In : *Codes and Cyphers : Cryptography and Coding IV*, P. G. Farrel editor, pages 33–47, 1995.

[124] MATSUI (M.) .- Linear cryptanalysis method for DES cipher. In : *Advances in Cryptology - EUROCRYPT'93*, Lectures Notes in Computer Science 765, Springer Verlag, 1994.

[125] MEIER (W.) et STAFFELBACH (O.) .- Fast Correlation Attack on certain Stream Ciphers. *J. of Cryptology*, pp 159–176, 1989.

[126] MEIER (W.) et STAFFELBACH (O.) .- Nonlinearity criteria for cryptographic functions. In : *Advances in Cryptology - EUROCRYPT'89*, Lecture Notes in Computer Science 434, pp 549–562, Springer Verlag, 1990.

[127] MENEZES (A.J.), VAN ORSCHOT (P.C.) et VANSTONE (S.A.) .- *Handbook of Applied Cryptography*. CRC Press, 1997.

[128] MICHELSON (A.M.) et LEVESQUE (A.H.) .- *Error-Control Techniques for Digital Communication* - Wiley, New York, 1985.

[129] MIHALJEVIC (M.) et GOLIĆ (J.) .- A Fast Iterative Algorithm for a Shift-Register Initial State Reconstruction given the Noisy Output Sequence.*Proc. Auscrypt'90*, Lecture Notes in Computer Science 453, Springer Verlag, 1990.

[130] MONTGOMERY (D.C.) .- *Designs and and Analysis of Experiments*, Wiley, 1996.

[131] http ://www.cryptonessie.org/

[132] PAASKE (E.) .- Short binary convolutional codes with maximal free distance. *IEEE Transactions on Information Theory*, IT-20, pp. 683–688, Sept. 1974.

[133] O'DONOGHUE (C.) et BURKLEY (C.) .- New algorithm to identify rate $\frac{k}{n}$ catastrophic punctured convolutional encoders. In : *Proceedings of the Workshop in Coding and Cryptography 99*, Paris, 1999.

[134] PALIT (S.) et ROY (B.) .- Cryptanalysis of a LFSR-encrypted codes with unknown combining function. In : *Advances in Cryptology - ASIACRYPT'99*, Lecture Notes in Computer Science, n$^o$ 1716 - Springer Verlag, 1999.

[135] PAPADIMITRIOU (C.H.) .- *Computational Complexity*. Addison-Wesley, 1994.

[136] PENZHORN (W.) .- Correlation Attacks on Stream Ciphers : Computing low Weight Parity Checks based on Error-Correcting Codes. In : *Proceedings of FSE'96*, Lecture Notes in Computer Science 1039, Springer Verlag, 1996.

[137] PLANQUETTE (G.) .- *Identification de train binaires codés* - Thèse de doctorat, Université de Rennes I, Décembre 1996.

[138] PLESS (V.) .- Encryption schemes for computer confidentiality. *IEEE Transaction on Computer*, Vol. 26, pp 1133–1136, 1977.

[139] PLESS (V.S.) et HUFFMAN (W.C.) .- *Handbook of Coding Theory*. North-Holland, 1998.

[140] PREENEL (B.) .- *ANalysis and Design of Cryptographic Hash Functions*. Thèse de Doctorat, Katholieke Universiteit Leuven, Belgique, 1993.

[141] B. PREENEL (B.), LEEKWIJCK (W.V.), LINDEN (L.V.), GOVAERTS (R.), VANDEWALLE (J.) .- Propagation Characteristics of Boolean Functions. In : *Advances in Cryptology - EUROCRYPT'90*, Lecture Notes in Computer Science 437, Springer Verlag, 1991.

[142] RAO (C.R.) .- Factorial experiments derivable from combinatorial arrangements of arrays. *J. Roy. Statist.*, Vol. 9, pp 128–139, 1947.

[143] RAY-CHAUDHURI (D.K.) et WILSON (R.M.) .- Solution of Kirkman's schoolgirl problem. *Proc. Symp. Pure Math. Amer. Math. Soc.*, Vol.19, pp 187–204, 1971.

[144] RIVEST (R.L.), SHAMIR (A.) et ADLEMAN (L.M.) .- A method for obtaining digital signatures and public-key cryptosystems. *Communications of the A.C.M.*, Vol. 21, Nr. 3, 1978, pp. 120–126.

[145] RICE (B.) .- Determining the parameters of a rate $\frac{1}{n}$ convolutional encoder over $GF(q)$. In *Third International Conference on Finite Fields and Applications*, Glasgow, 1995.

[146] ROTHAUS (O.S.) .- On bent functions. *Journal of Combinatorial Theory*, Vol. 20, pp 300–305, 1976.

[147] RUBIN (F.) .- Decrypting a stream cipher based on J-K flip-flops. *IEEE Transaction on Computers*, Vol. 28, pp 483–487, 1979.

[148] RUEPPEL (R.A.) .- *Analysis and Design of Stream Ciphers*. Springer Verlag, 1986.

[149] RUEPPEL (R.A.) et STAFFELBACH (O.) .- Products of Linear Recurring Sequences with Maximum Complexity. *IEEE Transactions on Information Theory*, Vol. 33 (1), pp 124–131, 1987.

[150] SCHNEIER (B.) .- *Applied Cryptography*. Wiley, 1996, Second Edition.

[151] SELMER (E.S.) .- *Linear Recurrence Relation over Finite Fields*, Ph. D Thesis, University of Bergen, Norway, 1966.

[152] SHANNON (C.E.) .- A mathematical theory of communication. *Bell System Journal*, Vol. 27 pp. 379-423 (Part I) et pp. 623-656 (Part II), 1948.

[153] SHANNON (C.E.) .- Communication Theory of Secrecy Systems.*Bell System Technical Journal*, Vol. 28, Nr.4, pp 656–715, 1949.

[154] SIEGENTHALER (T.) .- Correlation immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, IT-35 Nr. 5, September 1984, pp 776–780.

[155] SIEGENTHALER (T.) .- Decrypting a class of stream ciphers using ciphertext only. *IEEE Transactions on Computers*, C-34, 1, pp 81–84, 1985.

[156] SIEGENTHALER (T.) .- Cryptanalysis representation of nonlinearly filtered ML-sequences. In : *Advances in Cryptology - EUROCRYPT'85*, Lecture Notes in Computer Science 219, pp 103–110, Springer Verlag, 1986.

[157] SIMPSON (L.), DAWSON (E.), GOLIĆ (J.) et MILLAN (W.) .- LILI keystream generator. In *Proceedings of the 7th Workshop on Selected Areas in Cryptology - SAC'2000*, à paraitre dans Lecture Notes in Computer Science, Springer Verlag, 2000.

[158] SIMPSON (L.), DAWSON (E.), GOLIĆ (J.) et MILLAN (W.) .- The LILI-128 keystream generator. In *Proceedings of the First NESSIE Conference 2000*, Leuwen, 2000.

[159] SKOLEM (T.) .- Some remarks on the triple systems of Steiner. *Math. Scand.*, Vol. 6, pp 273–280.

[160] Der SPIEGEL *Wer ist der befugte Vierte ?*, n$^o$ 36, 1996. Version électronique : *http ://jya.com/cryptoag.htm.*

[161] STEINER (J.) .- Combinatorische Aufgabe. *Journal für die reine und angewandte Mathematik*, Vol. 45, pp 181–182, 1853.

[162] TARANNIKOV (Y.) .- On resilient Boolean functions with maximal possible nonlinearity. *Http ://eprint.iacr.org/2000/005.ps.gz*

[163] TARDY-CORFDIR (A.) et GILBERT (H.) .- A known plaintext attack on FEAL-4 and FEAL-6. In : *Advances in Cryptology - CRYPTO'91*, Lecture Notes in Computer Science, n$^o$ 576, Springer Verlag, 1991.

[164] THAS (J.A.) .- *Partial Geometries.* In Handbook of Combinatorial Designs, C. J. Colbourn, J. H. Dinitz eds., CRC Press, 1996.

[165] VALEMBOIS (A.) *Détection, reconnaissance et décodage de codes linéaires binaires*, Thèse de doctorat, Université de Limoges, Septembre 2000.

[166] VAN LINT (J.H.) .- *Introduction to Coding Theory.* Springer Verlag, New York, 1982.

[167] VERNAM (G.S.) .- Cipher printing telegraph systems for secret wire and radio telegraphic communications. *Journal of The American Institute for Electrical Engineers*, Vol. 55, pp 109–115, 1926.

[168] VITERBI (A.J.) .- Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, Vol. IT-13, pp. 260-269, 1967.

[169] WEBSTER (A.F.) et TAVARES (S.E.) .- On the design of Sboxes. In : *Advances in Cryptology - CRYPTO'85*, Lecture Notes in Computer Science 218, Springer Verlag, 1986.

[170] L'affaire Bühler est détaillée dans *Res Strehle : der Fall Hans Bühler*, Werd Verlag, Zürich, 1994.

[171] WOOLHOUSE (W.S.B.) .- Prize questions 1733. *Lady's and Gentleman's Diary*, 1844.

[172] WOZENCRAFT (J.M.) .- *Sequential Decoding* - MIT Press, Cambridge Mass., 1961.

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

[174] YASUDA (Y.), KASHIKI (K.) et HIRATA (Y.) .- High-rate punctured convolutional codes for soft decision Viterbi decoding. *IEEE Transactions on Communications*, Vol. 32, No. 11, Mars 1984.

[175] ZENG (K.) et HUANG (M.) .- On the linear syndrome method in cryptanalysis. In : *Advances in Cryptology - CRYPTO'88*, Lecture Notes in Computer Science 405, Springer Verlag, 1990.

[176] ZENG (K.), YANG (C.H.) et RAO (T.R.) .- An improved linear syndrome algorithm in cryptanalysis with applications. In : *Advances in Cryptology - CRYPTO'90*, Lecture Notes in Computer Science 537, Springer Verlag, 1991.

[177] ZIERLER (N.) .- Linear recurring sequences.*J. Soc. Indus. Appl. Math.*, Vol. 7, pp 147–157, 1959.

[178] ZIERLER (N.) et MILLS (W.H.) .- Products of linear recurring sequences. *Journal of Algebra*, Vol. 27, pp 147–157, 1973.

[179] ZENG (Y.), ZHANG (X.M.) et IMAI (H.) .- Restriction, Terms and Nonlinearity of Boolean Functions. A paraitre dans le numéro spécial Cryptographie, Theoretical Computer Science, en l'honneur du professeur Arto Salomaa.

# Table des figures

# Liste des tableaux

# Table des matières