UNIVERSITÉ DE **PARIS 7 - Denis Diderot**

La Sécurité des Primitives Cryptographiques

THÈSE DE DOCTORAT *Avril 1995*

Président du jury :
Philippe Flajolet

Rapporteurs :
Jean-Jacques Quisquater
Claus Peter Schnorr

Examinateurs :
Guy Cousineau
Claude Crépeau
Michel Minoux
Adi Shamir
Brigitte Vallée

Directeur de thèse :
Jacques Stern

Serge Vaudenay
Laboratoire d'Informatique de
l'Ecole Normale Supérieure

A la mémoire
d'Edmond Besse.

Remerciements

Je souhaite tout d'abord exprimer ma gratitude envers tous mes professeurs qui m'ont enseigné les mathématiques et l'informatique. J'ai tenté dans cette thèse de refléter la qualité de leur enseignement. Je suis tout particulièrement reconnaissant envers Jacques Stern qui m'a successivement enseigné la théorie des ensembles, la logique et la théorie de la complexité, qui m'a accueilli dans son équipe de recherche et qui a bien voulu me guider jusqu'à l'accomplissement du présent travail. Cette thèse n'aurait pu être menée à terme sans ses (nombreuses) critiques. J'espère m'être montré digne de la rigueur de son exigence.

Je suis également reconnaissant envers les autres chercheurs français avec qui j'ai souvent eu d'intéressantes discussions, notamment Paul Camion, Pascale Charpin et les autres membres du projet CODE de l'INRIA, Stéphane Boucheron, Claude Carlet, Marc Girault, Jacques Patarin, Miklos Santha, Philippe Toffin, Brigitte Vallée et ceux que j'aurait involontairement oublié. J'adresse aussi mes remerciements aux nombreux chercheurs étrangers qui m'ont aidé dans mes travaux : Ross Anderson, László Babai, Antoon Bosselaers, Joel Friedman, Carlo Harpes, Gyula Katona, James Massey, Ueli Maurer, Bart Preneel, Jean-Jacques Quisquater, Adi Shamir et plus particulièrement Don Coppersmith et Claus Schnorr avec qui j'ai eu l'honneur de partager des publications.

Mes travaux de recherche ont été effectués dans le Groupe de Recherche en Complexité et Cryptographie, petite équipe conviviale dans laquelle on trouve toujours une aide spontanée de tout ordre. De nombreuses discussions fructueuses ont initié plusieurs de mes travaux. Je remercie donc chaleureusement toutes les personnes ayant fait partie de cette équipe tout au long de mes recherches : Philippe Béguin, Ludovic Caudal, Florent Chabaud, Jean-Marc Couveignes, Claude Crépeau, Antonella Cresti, Jean-Bernard Fischer, Louis Granboulan, Philippe Hoogvorst, Antoine Joux, Lars Knudsen, David Pointcheval et Jacques Stern.

A plusieurs reprises, dans cette équipe, j'ai participé à des travaux de recherche contractuels. Si les résultats (à une exception près) ne s'inscrivent pas dans le cadre de ce mémoire, il m'est toutefois agréable de remercier collectivement les partenaires du groupe. Je mentionne notamment David M'Raïhi, David Naccache et Patrice Peyret de l'entreprise GEMPLUS qui ont toujours témoigné l'intérêt d'une telle collaboration. D'ailleurs, ils sont, en quelque sorte, initiateurs de mes travaux de recherche puisque j'ai découvert le coté réel de la cryptographie au sein

de leur entreprise, dans le cadre d'un stage de DEA. De plus, je suis spécialement reconnaissant envers le CELAR de m'avoir autorisé à publier le résultat d'un travail de recherche effectué à sa demande et qui est l'objet du chapitre 7 de ce mémoire.

Je remercie toutes les personnes qui m'ont aidé de quelque manière possible, mes parents, Hervé Brönnimann et Myriam Maumy. Un grand merci à Félix Hassine et David M'Raïhi pour avoir accepté de relire ce mémoire. Je remercie les élèves, secrétaires, chercheurs, enseignants de l'Ecole Normale Supérieure et plus particulièrement du Département de Mathématiques et d'Informatique qui entretiennent dans l'école un cadre de recherche fertile, ainsi que le Service des Prestations Informatiques pour son aide inestimable. Je suis notamment redevable envers Julien Cassaigne du lemme 57, suite à une demande faite sur les tableaux électroniques de l'école.

J'aimerais enfin exprimer ma profonde gratitude envers les membres du jury :

- le président Philippe Flajolet dont la présence m'honore;
- les rapporteurs Jean-Jacques Quisquater et Claus Schnorr pour leurs précieuses remarques constructives, leur soutien, l'accueil chaleureux qu'ils m'ont réservés dans leur laboratoire respectif;
- Guy Cousineau, Claude Crépeau, Michel Minoux et Brigitte Vallée pour leur sympathique participation spontanée et leur aide au cours de mes études;
- Adi Shamir, en précisant que j'attache une importance particulière à l'honneur qu'il me fait par sa présence et son soutien répété, pour la distance qu'il a bien voulu parcourir, son acceptation à assister à une soutenance en français autant que pour le fait qu'il représente à lui seul les plus importants progrès de la cryptologie moderne
- et bien sûr Jacques Stern pour son ultime soutien.

Pour conclure, je tiens à souligner l'ambiance cordiale qui règne dans la communauté internationale des chercheurs en cryptographie, bien qu'il soit naturel de penser qu'elle est *a priori* conflictuelle. Ceci ajouté à l'enthousiasme que j'éprouve à travailler dans un domaine qui nécessite des concepts théoriques pour résoudre des problèmes concrets me permettra sans doute de rappeler le leitmotiv que j'ai propagé dans l'Ecole Normale Supérieure jusque dans la plaquette du Département de Mathématiques et d'Informatique : la crypto c'est rigolo !

Résumé

Dans les années cinquante, Claude Shannon initia la théorie des primitives cryptographiques. Il définit les notions de diffusion et de confusion. Toutefois, cette théorie a très peu évolué jusqu'à nos jours. Récemment, la cryptanalyse différentielle et la cryptanalyse linéaire marquèrent un progrès significatif dans l'analyse des primitives. Des critères de sécurité sur les boîtes de confusion, essentiellement des critères de non-linéarité, furent proposés.

Dans cette thèse, on montre comment construire une notion de complexité sur la structure de graphe des primitives et comment l'étudier. Ceci fournit des critères de sécurité sur le réseau de calcul. On propose également de nouveaux critères sur la diffusion. On unifie enfin les deux types de cryptanalyse en les débarrassant de leur aspect linéaire par une approche statistique.

Abstract

In the early fifties, Claude Shannon initiated the theory of cryptographic primitives. He defined the notion of diffusion and confusion. However, this theory did not develop very much until now. Recently, the differential cryptanalysis and the linear cryptanalysis gave a significant advance in the analysis of the primitives. Security criteria for confusion, essentially nonlinearity criteria, has been proposed.

In this thesis, we show how to define a notion of complexity on the graph structure of the primitives and how to study it. This gives security criteria of the computational network. We propose new criteria for diffusion. Finally, we unify the two types of cryptanalysis, by getting rid of their linear aspects by a statistical approach.

Chapitre 1

Introduction

L'histoire de la cryptologie, racontée par David Kahn, est jalonnée d'anecdotes souvent décousues [4]. La *cryptologie conventionnelle* a toujours fait appel à des techniques hétéroclites : les méthodes de construction (en cryptographie) sont indépendantes, et les méthodes d'attaque (en cryptanalyse) aussi.

D'après certains auteurs, l'article de Whitfield Diffie et Martin Hellman de 1976 marque la naissance de la *cryptologie moderne* [3, 1]. Depuis, des objets nouveaux comme les *systèmes à clefs publiques* sont apparus et de nouvelles théories ont été construites, mais le succès de la recherche rencontré dans ces domaines entraîne un désintérêt des précédents. Dans le présent travail, on se démarque de ce point de vue en revenant aux sources. Bien qu'elles furent désertées, on cherche à montrer qu'elle sont toutes aussi fertiles que les systèmes "modernes" construits dans une infrastructure mathématique complexe.

La cryptologie "conventionnelle", dans son aspect actuel, a été initié par Claude Shannon [18]. Les recherches dans ce domaine ont souvent été secrètes, et l'on constate aujourd'hui un manque de théorie. L'ambition de ce mémoire est de montrer qu'il est possible d'en construire une en complétant les quelques résultats disponibles et en proposant de nouvelles bases.

Les principaux objets primitifs utilisés en cryptologie sont bien définis. On utilise des *fonctions de chiffrement* pour résoudre le problème de la confidentialité, les *fonctions de hachage* pour le problème de l'intégrité, et les *générateurs pseudo-aléatoires* pour introduire du hasard dans certains protocoles. Pour construire de telles primitives, la méthode la plus utilisée semble empirique. Elle consiste à définir un réseau où chaque sommet représente une boîte de calcul, ce réseau devant être *assez* inextricable, et les boîtes *suffisamment* irrégulières pour que l'on ne parvienne pas à avoir une vue d'ensemble de la fonction qui nuirait à la sécurité ! Le présent mémoire est entièrement consacré aux primitives construites ainsi.

La fonction de chiffrement DES, dont l'étude est restée secrète, proposée comme standard par le gouvernement américain, semble *a priori* suivre ce prin-

cipe de construction [6]. Cependant, d'autres fonctions s'inspirant de ce modèle ont été cassées par de multiples méthodes, dont la *cryptanalyse différentielle* proposée par Eli Biham et Adi Shamir et la *cryptanalyse linéaire* proposée par Mitsuru Matsui, qui ont à peine ébranlé la sécurité de DES. On peut donc supposer que les experts du gouvernement américain avaient une idée sur les critères de sécurité de telles fonctions et cela donne l'espoir de trouver une théorie dans la recherche publique, ... 20 ans après.

Le présent mémoire se compose de deux chapitres d'introduction et de deux parties consacrées, l'une à l'étude de la structure géométrique des primitives, l'autre à celle des boîtes. Il présente quelques résultats, la plupart d'entre eux ayant été présentés dans des colloques internationaux et publiés dans leurs actes [42, 26, 56, 73, 43]. D'autres travaux effectués parallèlement dans le domaine des algorithmes à clefs publiques ne sont pas reproduits ici [90, 91, 92].

Dans la présente introduction, on expose la problématique de la cryptologie conventionnelle en définissant les primitives cryptographiques, ainsi que la notion de sécurité. Le chapitre 2 est plus consacré à l'état de l'art : on y trouve les principales techniques utilisées dans la construction ou l'attaque des primitives construites sur un réseau de calcul.

Dans la première partie, on présente une approche de l'analyse des primitives à partir de la géométrie du circuit. Ainsi, dans le chapitre 3, on définit un formalisme utile dans l'étude de la sécurité et le chapitre 4 développe dans ce cadre l'étude d'une classe générique d'algorithmes qui généralise les notions de *recherche exhaustive* et de *paradoxe des anniversaires*, principaux outils d'analyse. On montre notamment que la sécurité vis-à-vis de cette classe est liée aux propriétés spectrales du graphe de calcul. Le chapitre 5 illustre une notion intuitive de *contraction de graphe* qui consiste à grouper plusieurs boîtes ayant un comportement global représentable en une seule. Elle permet de casser la fonction de hachage FFT Hash II proposée par Claus Schnorr [25]. Cela motive la recherche de boîtes de comportement global "inextricable". Les idées de FFT Hash II ont été ensuite reprises dans un travail avec Claus Schnorr pour proposer une nouvelle famille de fonctions de hachage [26, 73]. Le chapitre 6 utilise les méthodes d'analyse génériques présentées plus haut pour montrer la sécurité de cette famille.

Dans la seconde partie, le rôle des boîtes dans la sécurité des primitives est étudié. Tout d'abord, dans le chapitre 7, on met en évidence le danger dû à l'absence de critère de sécurité en matière de *diffusion*. On illustre ce danger par une cryptanalyse partielle de MD4, fonction de hachage proposée par Ronald Rivest [27]. Dans le chapitre 8, on définit un tel critère : la notion de *multipermutation*. On étudie la relation de cette notion avec d'autres objets combinatoires déjà existant, ainsi que les méthodes pour en construire. Dans le chapitre 9, un exemple de *cryptanalyse linéaire* appliqué à la fonction de chiffrement SAFER proposée

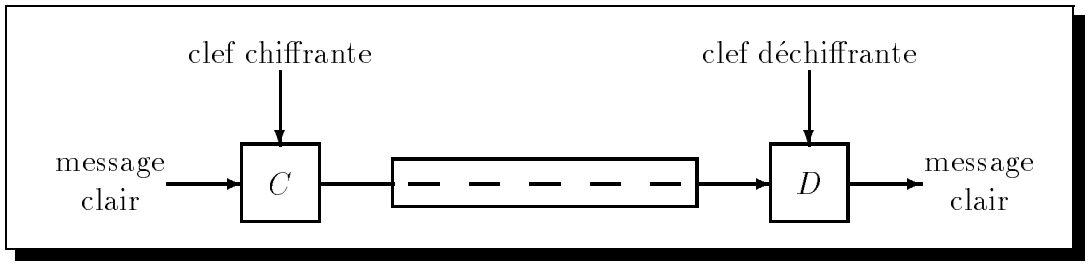

Figure 1.1 : Confidentialité dans un canal peu sûr.

par James Massey est présenté [11]. Dans le chapitre 11, une vision de la cryptanalyse des fonctions de chiffrement englobant la cryptanalyse différentielle et linéaire est présentée. On montre comment améliorer la meilleur attaque connue de DES et comment la retrouver sans aucun aspect linéaire. Enfin, le chapitre 10, résultat d'un travail avec Florent Chabaud, présente le lien entre l'étude de la sécurité des boîtes vis-à-vis de la *cryptanalyse différentielle* et celle vis-à-vis de la cryptanalyse linéaire.

1.1 Les primitives cryptographiques

On donne ici la définition de quelques objets cryptographiques que l'on appellera *fonctions primitives*. On illustre leur usage dans des problèmes simples.

Chacun de ces objets spécifie un problème supposé difficile contre lequel il doit offrir une certaine *sécurité*. En général, on ne sait pas la minorer de manière intéressante. Le rôle des cryptanalystes est de la majorer de manière aussi spectaculaire que possible. La sécurité correspond à la complexité, c'est-à-dire le coût en temps, en espace de travail ou en matériel de calcul, de la résolution de problèmes contre lesquels on souhaite se protéger.

1.1.1 Fonctions de chiffrement

La confidentialité est historiquement le premier problème posé à la cryptographie. Il se résout par la notion de *chiffrement* : un *message clair* est préalablement chiffré par une fonction de chiffrement C à l'aide d'une *clef chiffrante*. Un *déchiffrement* semblable s'opère au moyen d'une clef qui peut être différente et d'une fonction de déchiffrement D (voir figure 1.1).

Les principales méthodes de chiffrement sont présentées dans le livre de Dorothy Denning [2].

Les fonctions de chiffrement sont supposées rendre impossible le *décryptage*, c'est-à-dire la récupération d'un message clair sans la clef. *A fortiori*, elles doivent protéger le secret des clefs. On distingue plusieurs modèles d'attaque :

- **attaque à texte chiffré connu** : on dispose d'une liste de textes chiffrés;
- **attaque à texte clair connu** : on dispose d'une liste de couples formés d'un texte clair et de son texte chiffré;
- **attaque à texte clair choisi** : on dispose d'une boîte noire qui chiffre à l'aide de la clef secrète, ce qui permet de produire une liste de couples clairs/chiffrés avec un contrôle sur les textes clairs.

Chiffrement symétrique

Dans la cryptographie conventionnelle, les clefs sont identiques, et donc gardées secrètes. Le procédé de chiffrement est dit *symétrique*. Une méthode démontrée sûre pour effectuer du chiffrement symétrique existe, elle est d'ailleurs employée dans le *téléphone rouge* : le chiffre de Gilbert Vernam, connu en anglais sous le nom de *one-time pad* [16]. Elle est cependant très lourde d'emploi, donc trop coûteuse.

La méthode la plus employée pour réaliser un procédé de chiffrement symétrique consiste à concevoir un circuit suffisamment compliqué et avec des boîtes de calcul suffisamment irrégulières pour que son analyse soit difficile, en respectant des critères de sécurité. Le problème fondamental dans ce type de construction est de déterminer ces critères.

Le gouvernement américain a développé en 1977 la fonction DES (*Data Encryption Standard*) sur ce type de construction [6]. Cette fonction résiste encore à l'acharnement des cryptanalystes, ce qui laisse supposer que les inventeurs de cette fonction disposaient de bons critères. Malheureusement, ils ont gardé secrètes les études ayant conduit à cette fonction.

Une autre fonction répandue est la fonction IDEA (*International Data Encryption Algorithm*) développée par Xuejia Lai, James Massey et Sean Murphy [50, 17]. Une étude complète de cette fonction est disponible dans la thèse de Xuejia Lai [17]. On y trouve notamment des arguments qui justifient la structure de IDEA.

Chiffrement asymétrique

Dans la cryptographie à clef publique, les clefs sont différentes. La clef chiffrante est publique et la clef déchiffrante secrète. Le procédé est *asymétrique*.

La notion même de cryptographie à clef publique fut un défi mathématique lancé par Whitfield Diffie et Martin Hellman [3]. Ronald Rivest, Adi Shamir et Leonard Adleman furent les premiers à apporter une solution présumée aussi sûre que la factorisation est difficile [14]. Leur méthode, appelée RSA et aujourd'hui largement utilisée, repose sur une propriété de certains calculs mathématiques : il y a des calculs simples dont la démarche à contre-sens est difficile. Par exemple, il est simple de multiplier deux nombres, mais il est difficile de factoriser le résultat.

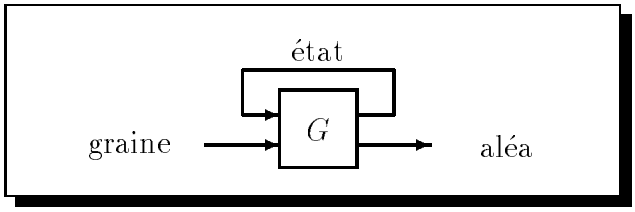

Figure 1.2 : Générateur pseudo-aléatoire.

La méthode RSA repose sur une complexité du même ordre : le problème de retrouver la clef secrète est équivalent au problème de la factorisation.

Ceci a été le premier pas vers le développement de procédés cryptographiques à clef publique. Ces procédés présentent l'avantage d'être implémentables et d'avoir une sécurité conditionnée à un problème sur lequel on a maintenant une vaste expertise. Ils présentent aussi le défaut d'être coûteux en terme d'implémentation et de reposer sur des sécurités équivalentes : si l'on découvre un procédé de factorisation efficace, la sécurité de tous ces procédés s'écroule ! Des progrès récents dans le domaine de l'ordinateur quantique précisent cette crainte. Peter Shor a montré qu'il est possible qu'une application de la mécanique quantique permette un jour de construire une machine qui factorise de grands nombres [72]. On ne sais pas à l'heure actuelle construire une machine mettant en application l'algorithme de Peter Shor, mais il est possible que les obstacles technologiques soient un jour surmontés. Cela relance donc l'intérêt pour la cryptographie à clef secrète.

1.1.2 Générateurs pseudo-aléatoires

Lorsqu'une personne choisit sa clef secrète, elle doit faire intervenir le hasard. De même, certains protocoles cryptographiques nécessitent un facteur de hasard supposé imprédictible par les opposants éventuels. Les différents procédés demandent donc l'aptitude à fabriquer du hasard.

L'objet cryptographique en rapport avec ce problème est le *générateur pseudo-aléatoire*. Il se décrit comme un automate fini dont l'état initial est la *graine* (ou *semence*). Son état évolue à chaque génération en produisant une quantité qualifiée d'*aléatoire* (voir figure 1.2). On formalise ainsi le générateur pseudo-aléatoire comme une fonction d'un ensemble fini (l'ensemble des états) dans un ensemble plus grand (l'ensemble des couples formés d'un aléa et d'un état qui définit le nouveau).

Plusieurs solutions utilisant des propriétés de la théorie des nombres ont été proposées [34, 33, 32, 35]. La méthode la plus couramment utilisée est encore la méthode empirique qui consiste à fabriquer un circuit dédié assez compliqué pour résister à l'expertise du cryptanalyste.

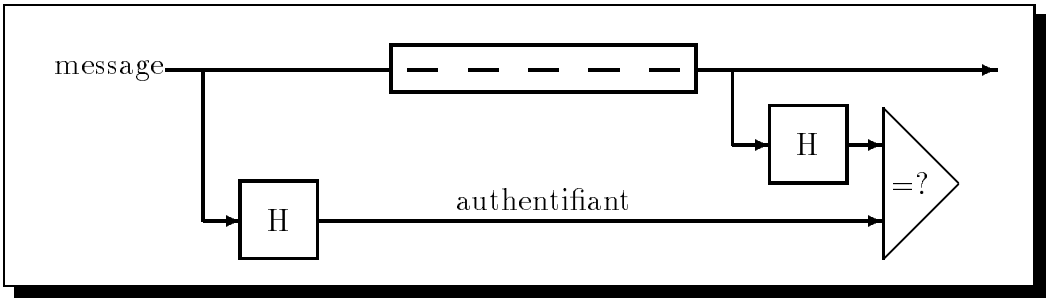

Figure 1.3 : Intégrité dans un canal peu sûr.

Les générateurs pseudo-aléatoires sont supposés être *imprédictibles*. Le modèle d'attaque courant consiste à observer les aléas engendrés pendant un certain nombre de générations pour deviner la suite, voire l'état du générateur.

1.1.3 Fonctions de hachage

Le problème de l'intégrité des données intervient dans différentes situations :

1. lorsque l'on partage des ressources de mémoire sur des ordinateurs : si l'on laisse des données sur une mémoire, on aimerait pouvoir être sûr qu'elles n'ont pas été volontairement modifiées quand on y accède à nouveau;

2. lorsque l'on communique avec une autre personne au travers d'un canal peu sûr : on aimerait que le destinataire soit convaincu que le message n'a pas été altéré volontairement;

3. lorsque dans un protocole, on doit choisir une donnée sans la révéler et s'engager à ne pas la modifier (par exemple, si plusieurs participants doivent choisir simultanément une valeur sans que le choix des uns ne dépende de celui des autres).

Une *fonction de hachage* est une fonction qui réduit une liste de données de taille arbitraire en une donnée de taille fixe. Dans le problème 1, on peut noter le résultat *haché* des données en mémoire sur un support sécurisé (son propre agenda par exemple). Dans le problème 2, si l'on dispose d'un canal sécurisé (mais plus coûteux) en parallèle, on peut communiquer le résultat haché par l'intermédiaire de ce canal (voir figure 1.3). Dans le problème 3, on *met en gage* le résultat haché de la donnée choisie.

Les fonctions de hachage ont été étudiées de manière exhaustive dans la thèse de Bart Preneel [31]. les méthodes de construction utilisées y sont dégagées. On trouve également de nombreux commentaires sur les cryptanalyses existantes.

Dans les problèmes d'intégrité, la structure de la fonction de hachage doit rendre difficile la fabrication de deux données différentes qui se hachent en la même valeur. On dit alors que la fonction de hachage est *sans collision*. Cette notion a été définie par Ivan Damgård [30].

Dans le problème de mise en gage, on demande de plus la difficulté d'obtenir des informations sur une donnée à partir uniquement de son résultat haché. On dit alors que la fonction est *à sens unique*. Cette notion a été proposée par Whitfield Diffie et Martin Hellman [3].

Le fait d'être sans collision et à sens unique définit deux critères qui sont en général les spécifications minimales des fonctions de hachage, en cryptographie, mais il existe de multiples autres spécifications de sécurité, qui dépendent de leur application, comme cela est montré par Ross Anderson [29]. Par exemple, on dit qu'une fonction est *correlation-free* s'il est difficile de trouver deux données telles que leur résultat haché diffère de peu de bits.

1.2 Analyse de la sécurité

1.2.1 Types de sécurité

Actuellement, on constate un saut entre les procédés destinés à un usage pratique, dont on ne sait pas montrer la sécurité autrement que par une expertise, et les procédés théoriques. Diverses notions de sécurité existent.

Sécurité parfaite. Cette notion, proposée par Claude Shannon, est liée à l'incapacité théorique de casser le problème (au sens de la théorie de l'information) [18]. Les rares exemples de protocoles qui offrent une sécurité parfaite, comme le chiffre de Gilbert Vernam par exemple, sont hélas trop coûteux [16].

Sécurité asymptotique. En général, un "procédé théorique" est une famille infinie de procédés associés à un paramètre (dit *paramètre de sécurité*) [88, 87, 85]. On arrive parfois à étudier le comportement asymptotique de la complexité des problèmes sous-jacents en fonction du paramètre ou à la réduire à celle d'un problème dont on a une vaste expertise. Par exemple, on dira qu'un procédé théorique est *sûr* si la complexité croît exponentiellement, ou si elle est équivalente à celle d'un algorithme qui résout un problème NP-complet. Une expertise *a posteriori* des cryptanalystes donne une idée du paramètre à utiliser en pratique. Hélas, ce paramètre est souvent trop grand.

Sécurité empirique. Un "procédé pratique" (ou une instance d'un procédé théorique pour un paramètre donné) est sûr si la complexité de son problème sous-jacent est supérieure à un seuil donné. Pour des applications militaires, on pourra fixer un seuil correspondant à l'exploitation des ressources de calcul dans

le monde entier pendant un siècle. Pour des applications domestiques, on pourra fixer un seuil correspondant à l'utilisation de machines coûteuses pendant une semaine.

1.2.2 Quelques outils d'analyse

Il existe des outils de base qui fonctionnent de manière universelle dans la cryptanalyse des primitives cryptographiques. La plupart des astuces algorithmiques de ces méthodes est décrite dans [71, 69].

Recherche exhaustive

Le premier outil est la *recherche exhaustive*. Pour chercher une solution à une équation, on "essaye" toutes les solutions potentielles jusqu'à en trouver une. On peut également essayer des solutions candidates en les engendrant de manière pseudo-aléatoire. Si la probabilité qu'un candidat soit solution est $\frac{1}{N}$, la complexité de cette attaque est en $O(N)$. Si le problème possède certaines symétries, on peut utiliser des astuces liées à la théorie des probabilités pour obtenir une complexité $O(\sqrt{N})$ par les méthodes détaillées dans la suite.

Attaque des anniversaires

Pour chercher une solution à l'équation $f(x) = f(y)$ avec $x \neq y$ où x et y sont les inconnues (c'est la recherche de collisions sur f), on utilise le *paradoxe des anniversaires* (voir [84]) : si l'on effectue k tirages aléatoires $x_1, \ldots, x_k$ dans l'espace de départ, si N est le cardinal de l'espace d'arrivée, en admettant que les $f(x_i)$ sont indépendants et uniformément distribués dans un espace de taille N, lorsque k est voisin de $\sqrt{N}$, la probabilité d'avoir une collision dans cette liste est voisine de

$$1 - e^{-\frac{k^2}{N}}.$$

Cela rend possible la recherche de collisions avec une complexité $O(\sqrt{N})$ en espace et en temps. En général, on réduit la complexité en espace d'un facteur constant en gérant des tables de hachage des $f(x_i)$ [71, 69].

Méthode ρ

On note que lorsque l'espace d'arrivée et de départ de f sont identiques, une astuce due à John Pollard (que l'on appelle méthode ρ) permet d'obtenir une complexité en espace constante et une complexité en temps en $O(\sqrt{N})$ [70]. Si f est une fonction aléatoire uniformément distribuée, en prenant un x_0 quelconque, la suite des itérés de x_0 par f, nécessairement ultimement périodique, possède une collision $f(x_k) = f(x_{k+T})$ où T est la période. Si k est le plus petit possible, on montre que l'espérance de $k + T$ est $O(\sqrt{N})$ [67].

Attaque dans le milieu

Pour rechercher une solution à l'équation $f(x) = g(y)$ où x et y sont les inconnues (ce que l'on appellera la recherche d'une *pince*, de l'anglais *claw*), on généralise naturellement la méthode des anniversaires, comme cela est étudié par Marc Girault, Robert Cohen et Mireille Campana [68]. On gère deux listes de $f(x_i)$ et de $g(y_j)$. Lorsque les deux listes sont de taille k et k', la probabilité qu'elles possèdent une pince est

$$1 - e^{-\frac{kk'}{N}}.$$

Lorsque l'équation est de la forme $g(f(x), y) = c$ où c est un paramètre et g est inversible connaissant y, cette méthode prend le nom d'*attaque dans le milieu* [66].

Conclusion

La demande de sécurité dans la vie courante montre le besoin de fonctions cryptographique primitives.

La théorie des nombres apporte des solutions à la plupart de ces besoins, avec ses avantages et ses défauts. L'avantage réside principalement dans la possibilité de ramener la preuve de sécurité à des problèmes sur lesquels on dispose d'une expertise très étendue. Le principal défaut, outre le problème du coût d'implémentation qui tend à devenir secondaire, est que la sécurité de toutes ces solutions repose sur des problèmes que l'on pense pouvoir résoudre simultanément dans l'avenir. Cela place la sécurité de leurs applications en porte-à-faux.

Dans le domaine des primitives cryptographiques, la seule méthode efficace (du point de vue de son coût d'implémentation, de son coût d'utilisation et de sa sécurité) consiste à concevoir des circuits dédiés : la fonction est décrite par un réseau de calcul sur lequel sont placées des boîtes. La sécurité, exclusivement empirique, repose uniquement sur une expertise dédiée à la primitive.

On constate l'absence de liens entre les diverses expertises, ce qui conduit à des cryptanalyses indépendantes de certaines fonctions. Ces attaques, à de rares exceptions près, reposent exclusivement sur des astuces complétées par de la puissance de calcul brutale.

Dans ce mémoire, on trouvera des exemples de telles attaques, sur des fonctions primitives dédiées. De plus, on verra une nouvelle approche d'étude systématique de la sécurité d'une fonction dédiée par une double analyse des propriétés locales des boîtes utilisées et des propriétés globales du circuit.

Chapitre 2

Etat de l'art

Do we need two theories?
Peter Landrock

La plupart des primitives cryptographiques proposées dans la littérature se décrivent par un réseau de calcul. On dénombre quelques méthodes classiques de construction, adaptées à tel ou tel type de primitive, ce qui donne lieu à plusieurs théories indépendantes. De plus, on commence à avoir des critères de sécurité sur les boîtes du réseau. On se propose ici de dresser l'état des connaissances sur le sujet.

2.1 Formalisation des primitives

2.1.1 Définitions

On considère une fonction primitive comme étant une fonction f d'un ensemble Z^p dans un ensemble Z^q, pour un alphabet Z. Cette fonction peut éventuellement être paramétrée par une clef k. Pour une fonction de chiffrement, on a $p = q$, pour un générateur pseudo-aléatoire, on a $p < q$, et pour une fonction de compression, on a $p > q$. Chacune de ces primitives admet des spécifications de sécurité.

Fonction de chiffrement

La fonction f_k, qui doit être réversible, est paramétrée par une clef secrète k. Elle doit rendre difficile la récupération de la clef lorsque l'on dispose de f_k comme boîte noire. Le procédé de fabrication des fonctions f étant supposé connu, l'espace des clefs doit être suffisamment grand pour rendre la recherche exhaustive impossible. En général, on utilise un espace d'au moins 2^{64} clefs.

Générateurs pseudo-aléatoires

On écrit Z^q comme étant $Z^p \times Z^{q-p}$. Si le générateur est dans l'état s, et si $f(s) = (s', x)$, le générateur passe dans le nouvel état s' en engendrant l'aléa x. f doit rendre difficile la récupération d'un état s lorsque l'on dispose d'une suite de x consécutifs. L'espace des états doit donc être suffisamment grand pour rendre l'utilisation d'une attaque de type anniversaire impossible : si l'on dispose de $O(Card(Z)^{\frac{p}{2}})$ valeurs x consécutives et si l'on essaye autant d'états de manière aléatoire, on pourra détecter une séquence connue dans la suite. En général, on utilisera donc 2^{128} états différents codés sur 128 bits.

Fonctions de compression

On écrit Z^p comme étant $Z^q \times Z^{p-q}$. Une telle fonction permet de construire une fonction de hachage, suivant un procédé itératif. Pour hacher un message représenté comme un mot m sur l'alphabet Z, on complète m pour obtenir un mot m' de longueur multiple de $p-q$. Cette étape préliminaire doit être injective. Par exemple, on peut utiliser le procédé proposé indépendamment par Ralph Merkle [22] et Ivan Damgård [20] qui consiste à ajouter un petit nombre de 0 (une lettre de Z), un 1 (une autre lettre), et la longueur de m codée sur l'alphabet Z. Le nombre de 0 est ajusté pour obtenir la longueur multiple de $p - q$.

Le mot m' de longueur $n(p-q)$ est découpé en blocs $m' = m_1 \dots m_n$. Le hachage s'effectue alors de manière itérative. h_0 est une *valeur initiale*. On pose $h_{i+1} = f(h_i, m_{i+1})$. Le résultat du hachage est h_n.

La fonction de hachage doit rendre difficile la recherche de collisions. Pour rendre une attaque de type anniversaire impossible, on prendra donc, en général, $Card(Z)^p = 2^{128}$.

La spécification induite sur la fonction de compression n'est pas clairement établie. Il est évident que la difficulté de la recherche de collisions sur f est suffisante, mais elle ne semble pas nécessaire. Inversement, la difficulté de trouver une solution de $f(a, x) = f(b, y)$ pour a et b fixés est nécessaire, mais n'est peut-être pas suffisante.

2.1.2 Structure des primitives

Réseau de calcul

On s'intéresse aux fonctions primitives définies par un réseau de calcul : on a un graphe (V, E) orienté sans cycle composé de p sommets d'entrée V_i (ceux qui n'ont pas de prédécesseurs), de q nœuds de sortie V_o (ceux qui n'ont pas de successeurs), chaque arête e correspond à une valeur d'un domaine D_e (le plus souvent, $D_e = Z$), et chaque sommet interne v correspond à une boîte de calcul R_v (que l'on appellera *relation*), c'est-à-dire une fonction du produit des domaines des arêtes entrantes dans le produit des domaines des arêtes sortantes.

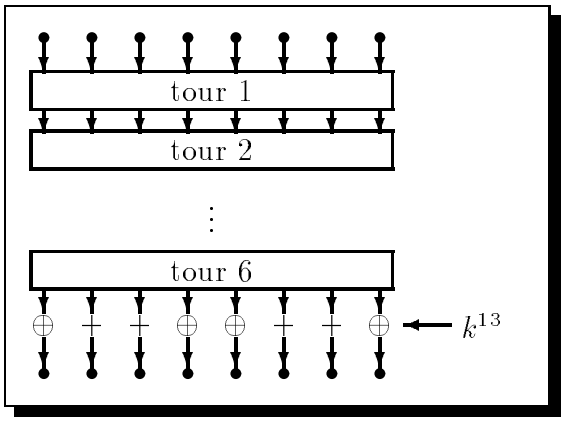

Figure 2.1 : Vue générale du réseau de SAFER.

Ce modèle de primitive est celui qui est le plus utilisé en pratique. Souvent, le réseau d'une primitive est décrit comme l'itération d'un petit réseau. Le nombre d'itérations est appelé *nombre de tours*. C'est ce nombre qui détermine la sécurité.

Depuis la théorie introduite par Claude Shannon en 1949 [18], on distingue deux types de boîtes :

- les boîtes de **confusion** (les boîtes à une seule entrée); ce sont des permutations qui servent à cacher toute structure remarquable, tant sur le plan algébrique que statistique;
- les boîtes de **diffusion** (les boîtes à plusieurs entrées), qui permettent de fusionner plusieurs informations de telle sorte que chacune soit correctement diffusée.

Exemple de SAFER

La fonction de chiffrement SAFER (*Secure And Fast Encryption Routine*) introduite par James Massey illustre bien ces notions [11]. L'alphabet utilisé est l'ensemble des octets, soit l'ensemble $Z = \{0, 1, \ldots, 255\}$. La fonction est décrite par 6 tours complétés par une opération finale. Une vue d'ensemble du réseau de SAFER est illustrée sur la figure 2.1. Le réseau d'un tour est représenté sur la figure 2.2. Les boîtes de confusion sont :

- l'addition + modulo 256 ou le *ou*-exclusif bit-à-bit $\oplus$ avec une information sur la clef secrète;
- une permutation P et sa réciproque Q.

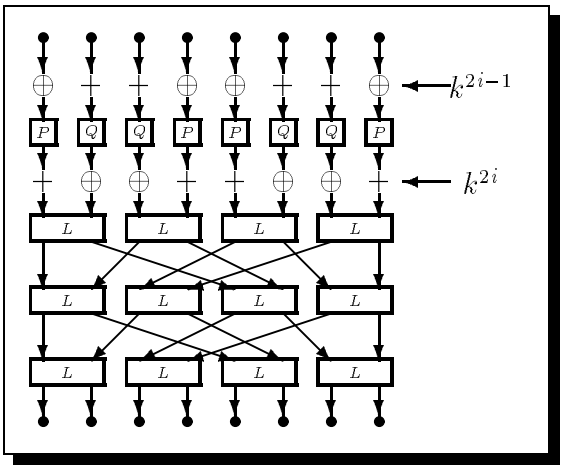

Figure 2.2 : Tour i dans SAFER.

La permutation P est définie par

$$P(a) = (45^a \bmod 257) \bmod 256.$$

Il n'y a qu'un seul type de boîte de diffusion : la boîte L définie par

$$L(a, b) = (2a + b, a + b) \pmod{256}.$$

L'opération finale (après les 6 tours) est une couche supplémentaire de confusion $\oplus/+$ avec une clef, comme cela se ferait dans un septième tour.

On constate que la boîte de diffusion est relativement simple, puisqu'elle est linéaire. De même, les boîtes de confusion utilisant la clef sont également très simples, mais on peut les voir comme des boîtes de diffusion entre le message et la clef. En revanche, le rôle des boîtes de confusion P et Q est de "casser" les structures linéaires. Dans ce cas, on peut trouver une structure algébrique aux fonctions P et Q (précisément, une structure d'homomorphisme entre les groupes $(\mathbb{Z}/257\mathbb{Z})^*$ et $\mathbb{Z}/256\mathbb{Z}$), mais l'enchevêtrement de ces structures différentes rend la fonction sûre, *a priori*. Une étude de la sécurité de SAFER est présentée dans le chapitre 9.

2.2 Construction des primitives

2.2.1 Construction de fonctions de chiffrement

La plupart des fonctions de chiffrement sont construites par itération de *tours* qui utilisent des *sous-clefs*. Un procédé annexe de *diversification de clef* (construit

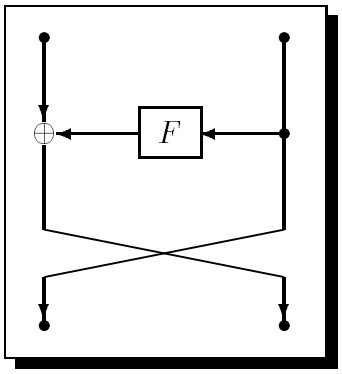

Figure 2.3 : Schéma de Horst Feistel.

également à l'aide d'un réseau de calcul) permet de produire des sous-clefs qui sont utilisées dans chaque tour. Dans le cas de SAFER, un procédé de diversification de clef produit 13 sous-clefs de 8 octets à partir d'une clef maître.

Chaque tour doit représenter une fonction bijective. Plusieurs constructions sont utilisées.

Utilisation de boîtes réversibles

La méthode la plus évidente consiste à utiliser des boîtes localement réversibles (donc avec autant d'entrées que de sorties). Elle est cependant difficile à mettre en œuvre, car la construction de telles boîtes vérifiant une liste de critères de non linéarité imposent des contraintes lourdes. Cette méthode est celle qui est utilisée dans la fonction SAFER [11].

Schéma de Horst Feistel

Dans les années 70, le gouvernement américain commanda le développement d'une fonction de chiffrement à IBM. La fonction Lucifer imaginée par Horst Feistel fut tout d'abord proposée, mais une expertise la rejeta [9]. Chaque tour est basé sur un schéma simple qui fut largement repris par la suite. Il utilise une loi de groupe (à l'origine, le *ou*-exclusif bit-à-bit $\oplus$) et une fonction quelconque F (voir figure 2.3).

Une seconde proposition conduisit au standard de chiffrement DES, basé lui aussi sur ce schéma [6]. La fonction F utilisée dans DES est illustrée sur la figure 2.4. Les fonctions $\oplus$ utilisent la clef. Il est d'usage de considérer les S-boîtes comme des fonctions de confusion dont l'entrée est codée sur 6 bits et la sortie sur 4.

Il existe de nombreuses fonctions basées sur ce schéma : FEAL-N [15, 13], REDOC [8], LOKI [7], KHUFU et KHAFRE [12].

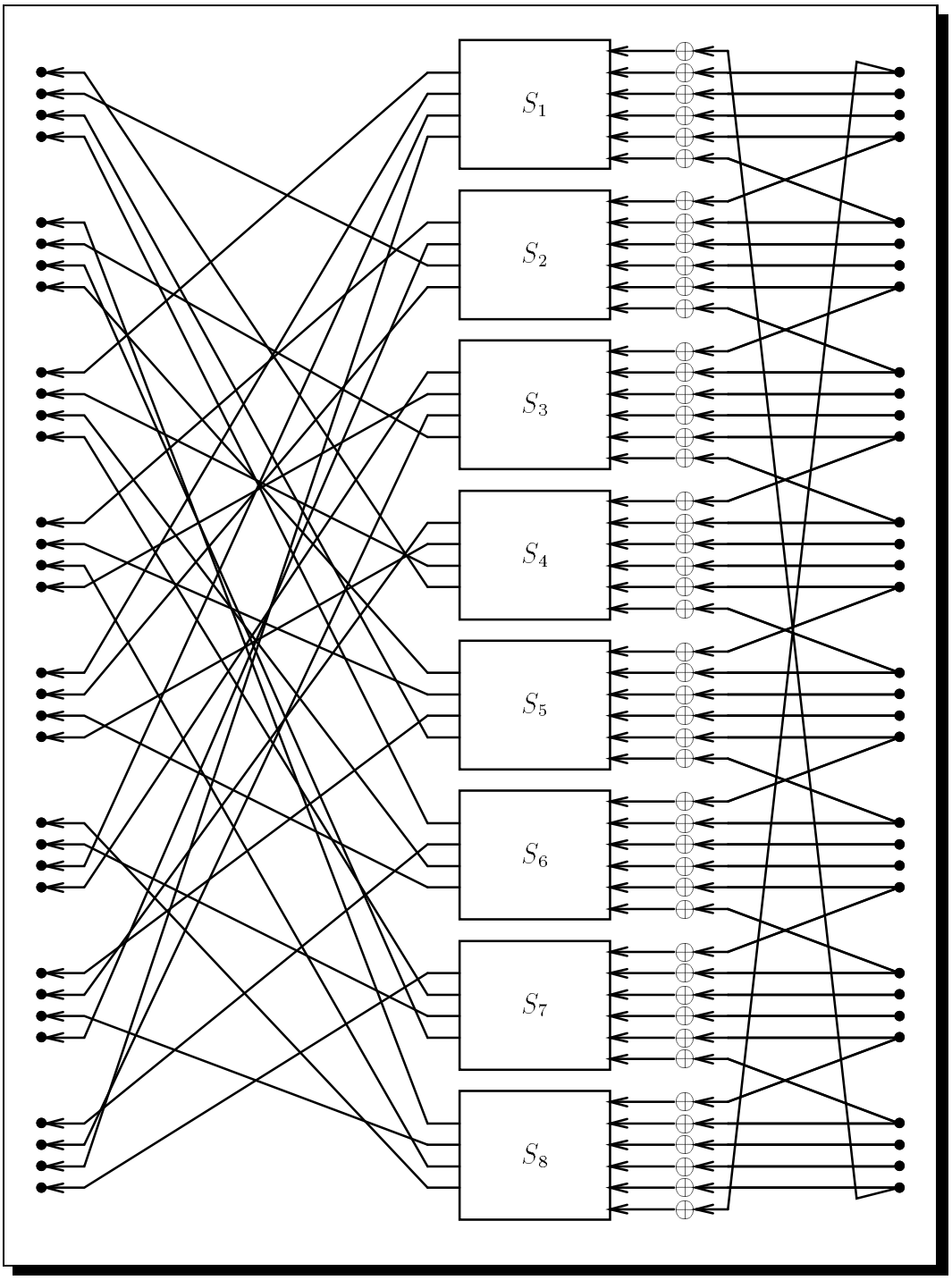

Figure 2.4 : Fonction F de DES.

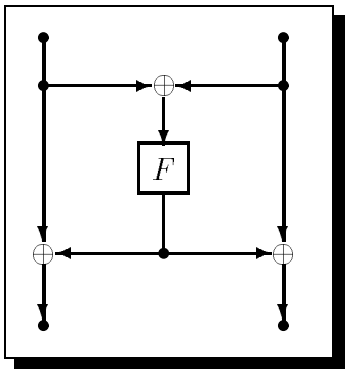

Figure 2.5 : Schéma de Xuejia Lai et James Massey.

Schéma de Xuejia Lai et James Massey

Un autre schéma fut proposé par Xuejia Lai et James Massey [10]. Il utilise la fonction $\oplus$ (voir figure 2.5).

La fonction IDEA s'inspire de ce schéma [17]. Elle utilise 8 tours, et une transformation finale (la couche $\cdot/+$). La figure 2.6 illustre un tour. Les flèches qui partent de rien utilisent une information sur la clef. Ici, l'alphabet utilisé est $Z = \{0, 1, \ldots, 65535\}$. $+$ est l'addition modulo 65536. $\cdot$ est défini par

$$a \cdot b = (ab \bmod 65537) \bmod 65536.$$

En fait, c'est la loi du groupe multiplicatif de $\mathbb{Z}/65537\mathbb{Z}$.

2.2.2 Construction de fonctions de compression

En général, une fonction de compression est construite à partir d'une fonction de chiffrement. Il n'existe toutefois pas de lien entre la sécurité des deux primitives. Ce constat, établi par Peter Landrock, implique qu'une bonne fonction de chiffrement peut donner une mauvaise fonction de compression.

Schéma de Donald Davies et Carl Meyer

Une construction fut proposée indépendamment par Donald Davies [21] et Carl Meyer [23] : si f_k est une fonction de chiffrement utilisant la clef k, on considère la fonction de compression $(h, m) \mapsto f_m(h) \oplus h$. L'irréversibilité provient de la double action de h.

Ronald Rivest a construit une série de fonctions de hachage construites suivant le procédé d'itération (voir page 18) à partir de fonctions de compression sur ce schéma. MD4 est une fonction utilisant 3 tours [27]. Des analyses, l'une non publiée effectuée par Ralph Merkle, l'autre effectuée par Bert den Boer et

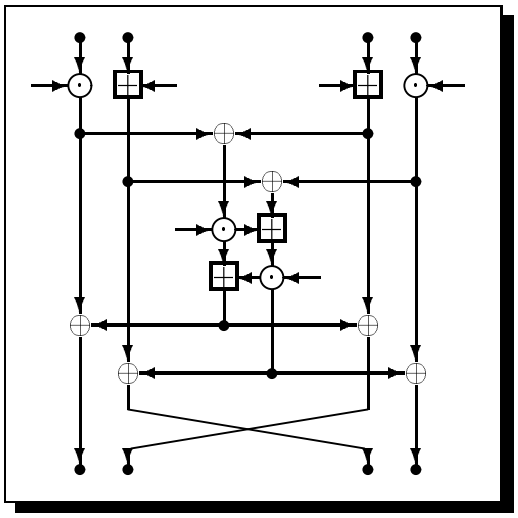

Figure 2.6 : Un tour de IDEA.

Antoon Bosselaers ont montré qu'il était facile de construire des collisions si l'on supprimait le premier ou le dernier tour [36]. Une attaque semblable est proposée dans le chapitre 7. MD5 est une fonction sur 4 tours possédant en outre une réparation de la faiblesse mise en évidence dans MD4 [28]. Cependant, cette réparation a donné lieu à la possibilité de construire des collisions pour la fonction de compression (mais pas pour la fonction de hachage), ce qui pouvait se révéler peu souhaitable [37].

Sur le même modèle, une nouvelle fonction SHA, sur 4 tours, a été adoptée comme standard de hachage par le gouvernement américain [19]. Cette fonction propose en outre de nouvelles idées. Aucune attaque de SHA n'est connue à l'heure actuelle.

Schéma de Claus Schnorr

Une autre construction fut proposée par Claus Schnorr dans la fonction FFT Hashing : si f est une permutation de $(Z^q)^2$, et si $f(h,m) = (h',x)$, le résultat compressé de h et m est h' [24]. Ainsi, l'irréversibilité s'obtient par l'"oubli" de x.

Un défaut de cette construction est que l'on ne peut pas défier de construire des collisions sur la fonction de compression, puisque cela est très facile. Cela ne rend bien sûr pas la fonction de hachage faible pour autant.

Thierry Baritaud, Henri Gilbert et Marc Girault ont montré que l'on pouvait construire des collisions pour la fonction de hachage FFT Hashing [38]. Une fonction FFT Hash II a été proposée [25]. Elle a été cassée par Serge Vaudenay (voir la chapitre 5) [42]. Cette construction de fonction de compression a toutefois été

reprise par Claus Schnorr et Serge Vaudenay conduisant à une nouvelle fonction de structure interne symétrique et hautement parallélisable [26]. La sécurité de cette nouvelle fonction est étudiée dans le chapitre 6.

2.3 Analyse de DES

Depuis le début de l'histoire de la cryptologie moderne, les principales théories développées en cryptanalyse en matière de primitives cryptographiques ont été motivées par le défi lancé par le gouvernement américain de casser la fonction DES [6]. Des techniques d'analyses ont été dégagées, et l'on dispose maintenant de nombreux critères de sécurité sur les fonctions de confusion.

2.3.1 Critères de non linéarité

Dans la fonction DES, il est difficile de représenter le réseau de calcul à l'aide d'un autre alphabet que $\{0,1\}$. Ainsi, les boîtes de confusion (autre que celles qui diffusent la clef) sont des S-boîtes avec une entrée de 6 bits et une sortie de 4 bits. Une S-boîte est donc un ensemble de 4 fonctions booléennes. Il pourra être intéressant, parfois, d'étudier des combinaisons linéaires de ces fonctions, ce qui donne onze autres fonctions booléennes.

Plusieurs critères de sécurité existent sur les fonctions booléennes. Ces critères ont largement été inspirés par les études menées sur le chiffrement en chaîne.

On définit quelques notations :

- **Distance de Hamming.** Pour un vecteur x, on note $w(x)$ son poids de Hamming, c'est-à-dire le nombre de ses coefficients non nuls. La distance induite se note $d(x,y) = w(y-x)$. De même, pour deux fonctions f et g, la distance $d(f,g)$ est le nombre d'éléments x tels que $f(x) \neq g(x)$.

- **Fonctions logiques.** Pour des vecteurs x et y à coefficients 0 ou 1, on note $x \wedge y$ le *et* bit-à-bit, $x \vee y$ le *ou* bit-à-bit, $x \oplus y$ le *ou* exclusif bit-à-bit (qui est aussi la somme sur le corps $\mathbb{Z}/2\mathbb{Z}$) et $\bar{x}$ la négation bit-à-bit.

- **Fonctions booléennes.** Si f est une fonction booléenne (donc à valeurs 0 ou 1), on note $\chi_f = (-1)^f = 1 - 2f$ sa représentation avec des ± 1.

- **Transformation de Hadamard-Walsh.** Pour toute fonction numérique f d'un vecteur sur le corps $\mathbb{Z}/2\mathbb{Z}$, on note

$$\hat{f}(x) = \sum_y f(y)(-1)^{x \cdot y}$$

où la somme est étendue sur tous les vecteurs y et où $x \cdot y$ est le produit scalaire de x par y. On remarque que $(-1)^{x \cdot y} = (-1)^{w(x \wedge y)}$. $\hat{f}$ est la transformée de Fourier discrète de f.

- **Produit de convolution.** Pour des fonctions numériques f et g sur des vecteurs sur le corps $\mathbb{Z}/2\mathbb{Z}$, on note le produit de convolution

$$(f \otimes g)(x) = \sum_y f(y)g(x \oplus y).$$

On rappelle les propriétés classiques :

$$\begin{aligned} \widehat{(\hat{f})} &= 2^n f \\ \widehat{f \otimes g} &= \hat{f}.\hat{g} \end{aligned}$$

où n est le nombre des bits d'entrée de f. En particulier, on a $\widehat{f \otimes f} = \hat{f}^2$.

Ordre de non linéarité

Si l'on écrit une fonction booléenne sous forme algébrique normale (c'est-à-dire sous forme de somme minimale de monômes dans lesquels chaque variable présente est de multiplicité 1), le degré du plus grand monôme est l'ordre de non linéarité de la fonction. Ce critère a été étudié par Willi Meier et Othmar Staffelbach [58].

Distance à une fonction affine

La distance de Hamming d'une fonction à l'ensemble $\mathcal{A}$ des fonctions affines est un critère de non linéarité. Les fonctions qui maximisent cette distance sont, dans certains cas, les fonctions courbes [79]. On note qu'elle est directement liée au maximum de la transformée de Fourier discrète de la fonction. En effet, en notant $\ell_{a,b}$ la fonction affine définie par $\ell_{a,b}(x) = (a \cdot x) \oplus b$, on a

$$2^n - 2.d(f, \ell_{a,b}) = \widehat{\chi_f}(a)(-1)^b$$

où a est un vecteur et b vaut 0 ou 1, donc

$$d(f, \mathcal{A}) = \min_{g \in \mathcal{A}} d(f, g) = 2^{n-1} - \frac{1}{2} \max_x |\widehat{\chi_f}(x)|.$$

Distance à une structure linéaire

Une fonction booléenne g admet une structure linéaire $a \neq 0$ si l'on a

$$g(x \oplus a) = g(x)$$

pour tout x. On note $\mathcal{S}_a$ l'ensemble des fonctions admettant a comme structure linéaire et $\mathcal{S}$ la réunion des $\mathcal{S}_a$. La distance de f à $\mathcal{S}_a$ est le plus petit nombre de

valeurs $f(x)$ à inverser pour obtenir la structure linéaire, soit la moitié du nombre de x tels que $f(x \oplus a) \neq f(x)$. On a donc

$$2^n - 4.d(f, S_a) = (\chi_f \otimes \chi_f)(a)$$

donc

$$d(f, \mathcal{S}) = \min_{g \in \mathcal{S}} d(f, g) = 2^{n-2} - \frac{1}{4} \max_{x \neq 0} (\chi_f \otimes \chi_f)(x).$$

Les fonctions qui maximisent ce nombre sont, dans certains cas, les fonctions parfaitement non linéaires (qui sont équivalentes aux fonctions courbes) [58]. Cette notion est duale de la distance à une fonction affine, car on a $\chi_f \widehat{\otimes} \chi_f = \widehat{\chi_f}^2$.

Immunité aux corrélations

Dans [63], on dit qu'une fonction est immunisée aux corrélations à l'ordre t si pour tout choix d'au plus t variables, si l'on fixe ces variables, la distribution de la fonction obtenue est toujours la même. Dans [65], on montre que cela revient à dire que la transformée de Fourier discrète de la fonction s'annule en tout point non nul de poids de Hamming au plus t. Le critère est donc

$$t = \min_{\hat{f}(x) \neq 0} w(x) - 1.$$

Critère de propagation

Dans [31], on dit qu'une fonction vérifie le critère de propagation à l'ordre t si le fait de changer au plus t entrées fait changer la sortie avec probabilité $\frac{1}{2}$. Cela revient à dire que la fonction d'auto-corrélation s'annule en tout point non nul de poids de Hamming au plus t. Le critère est donc

$$t = \min_{x \neq 0, (f \otimes f)(x) \neq 0} w(x) - 1.$$

C'est donc une notion duale de l'immunité aux corrélations (au sens de la transformée de Fourier discrète).

Ce critère généralise le critère de stricte avalanche [64] ($t = 1$) et la notion de non linéarité parfaite [58] (t est le nombre d'entrées).

2.3.2 Cryptanalyse différentielle

Une méthode d'attaque proposée par Eli Biham et Adi Shamir permet d'envisager une attaque à texte clair choisi des primitives basées sur le schéma de Horst Feistel [44, 48]. Elle consiste à étudier la propagation d'une différence $m \oplus m'$ entre deux textes clairs choisis dans le réseau de calcul. Elle utilise la distance à une structure linéaire de certaines fonctions. Cette méthode à permis de casser

un certain nombre de primitives [45, 46] et a conduit à la première cryptanalyse de DES plus rapide qu'une recherche exhaustive [47].

Dans [50], on trouve une approche théorique de ce type de cryptanalyse. La notion de *chiffre de Markov* a été proposée pour définir la classe des fonctions de chiffrement pour lesquelles les méthodes heuristiques développées par Eli Biham et Adi Shamir sont rigoureusement établies. Ces fonctions présentent l'avantage d'avoir une sécurité plus facilement démontrable vis-à-vis de cette attaque.

2.3.3 Cryptanalyse linéaire

Une méthode imaginée par Anne Corfdir et Henri Gilbert permet d'envisager une attaque à texte clair connu des primitives basées sur le schéma de Horst Feistel [54]. Elle a inspiré Mitsuru Matsui qui a proposé la cryptanalyse linéaire de DES avec une complexité analogue à celle de la meilleurs attaque de Eli Biham et Adi Shamir [52]. Elle a conduit à la première attaque expérimentale de DES [53].

Le principe de cette attaque consiste à enchaîner des corrélations entre les entrées et les sorties des boîtes pour obtenir une propriété statistiquement observable. Elle utilise la distance à une fonction affine.

Conclusion

Les primitives cryptographiques semblent liées, de part leur réduction mutuelle. Cependant, il n'existe pas de théorème pratique qui lie leur sécurité.

Il est important de faire la distinction entre les boîtes de confusion et les boîtes de diffusion. On note qu'il existe de nombreux critères de sécurité relatifs aux boîtes de confusion. Ce sont ceux qui assurent la résistance vis-à-vis des cryptanalyses différentielles ou linéaires.

On constate l'absence de critère de sécurité sur les boîtes de diffusion, ainsi que sur la structure du réseau.

Partie I

Le graphe des primitives cryptographiques

Chapitre 3

Graphes d'équation

Une primitive cryptographique se décrit souvent par un réseau sur lequel chaque sommet représente une boîte de calcul. De nombreux critères de sécurité existent sur la structure de ces boîtes, mais pas sur celle du réseau lui-même. Dans cette partie, on étudie l'implication de celle-ci dans la sécurité. Tout d'abord, dans ce chapitre, on définit un nouveau cadre permettant d'étudier la complexité des primitives dans lesquelles les boîtes ne sont pas explicitement définies. Chaque définition de l'ensemble des boîtes fournit une *interprétation*, et on fait alors une étude en moyenne sur toutes les interprétations possibles.

La fonctionnalité d'une primitive engendre des spécifications de sécurité. Chacune d'elles s'exprime, en général, par la difficulté de résoudre certaines équations. Par exemple, une fonction f est à sens unique si l'équation $f(x) = a$ est difficile à résoudre. Puisque f est décrite par un réseau de calcul, on définit ainsi un *graphe d'équation*. Ceci n'est rien d'autre qu'un système d'équations dans lequel chaque inconnue correspond à une arête du réseau, et chaque équation correspond à une boîte.

Si l'on cherche des critères sur les graphes pour analyser la complexité de la résolution de l'équation, il faut oublier la description exacte des équations (c'est-à-dire l'interprétation des boîtes) pour ne retenir que les relations d'interdépendance locales entre les inconnues, et le *degré de liberté* défini par la contrainte d'une équation. Ceci correspond à la notion de *graphe d'équation muet*. On peut imaginer un type de résolution qui cherche à satisfaire successivement chaque contrainte par une recherche exhaustive ou une attaque de type anniversaire. Cette idée de cryptanalyse est apparue dans l'article de Claus Schnorr et Serge Vaudenay sous le nom de *cryptanalyse par boîtes noires* [73]. On l'étend ici en définissant une classe générique d'algorithmes de résolution.

On montre dans ce chapitre comment formaliser de telles notions en utilisant des outils de l'informatique théorique et la notion de *laplacien* de graphe. On démontre également que le problème de l'étude de la complexité minimale d'un algorithme générique se ramène à un problème isopérimétrique sur les graphes. Les conséquences de cela sont étudiées dans le chapitre 4.

3.1 Définitions

3.1.1 Graphes d'équation

Graphes d'équation muets

Définition 1. Un *graphe d'équation muet* $G = (V, E, r)$ est la donnée :

- d'un graphe non orienté (V, E);
- d'une application r de $V \cup E$ dans $\mathbb{N}$ qui définit le *rang*.

On définit le *degré* d'un sommet v comme étant la somme $d(v)$ des rangs des arêtes adjacentes.

Chaque arête correspond à une valeur. Chaque sommet correspond à une relation sur les valeurs de ses arêtes adjacentes. On remarque que dans une telle définition, une valeur ne peut intervenir que dans deux relations différentes. Des systèmes plus généraux correspondent à la notion d'hypergraphe qu'il ne sera pas utile d'introduire, car il est facile de transformer un hypergraphe en graphe par l'adjonction de relations qui recopient les valeurs souhaitées.

Pour une arête e, $r(e)$ représente la dimension du domaine de la valeur correspondante. Pour un sommet v, $r(v)$ représente le *degré de liberté* de la relation correspondante, c'est-à-dire le nombre de valeurs que l'on peut fixer arbitrairement pour déterminer les autres.

Graphes d'équation et rangs

Définition 2. Un *graphe d'équation* $G = (V, E, r, k, D)$ est la donnée :

- d'un graphe d'équation muet (V, E, r);
- d'un entier k qui définit la *base*;
- d'une fonction D qui à une arête e de E associe un domaine D_e de cardinal $k^{r(e)}$.

En général, les arêtes sont toutes associées au même domaine Z de cardinal k, donc de même rang 1. En fait, une arête de rang plus élevé peut se représenter par une arête multiple. Ainsi, si le graphe d'équation provient d'un réseau de calcul, le rang d'un sommet est égal au nombre d'arêtes entrantes de la boîte correspondante, avant que l'on oublie l'orientation et la description de la boîte. Le degré est le nombre d'arêtes totales (entrantes ou sortantes).

Dans toute la suite, on pourra mesurer la taille des ensembles A par leur *rang*

$$rg(A) = \log_k Card(A).$$

Tuples et relations

Pour formaliser la notion d'équation, on utilise un vocabulaire analogue à celui des bases de données relationnelles :

Définition 3. Etant donné une famille d'ensembles $(D_e)_{e\in E}$:

- un *schéma* est une partie de E;
- un *tuple* $t = (t_e)_{e\in S}$ sur un schéma S est une fonction qui à $e \in S$ associe un élément t_e de D_e;
- une *relation* sur un schéma S est un ensemble de tuples sur ce schéma.

Graphes d'équation définis

Un graphe d'équation s'interprète en précisant les relations.

Définition 4. Un *graphe d'équation défini* $G = (V, E, r, k, D, R)$ est la donnée d'un graphe d'équation (V, E, r, k, D) et d'une *interprétation* R, c'est-à-dire d'une fonction qui à un sommet v de V associe une relation $R(v)$ de rang $r(v)$ dont le schéma est l'ensemble des arêtes adjacentes à v.

Intuitivement, une relation $R(v)$ est une contrainte explicitée par l'ensemble des solutions locales qui la respectent. Le problème de résoudre l'équation consiste à trouver un tuple sur le schéma complet E qui respecte toutes les relations, ou encore de trouver l'ensemble de ces tuples.

On considérera également des relations aléatoires. On les distinguera des autres en les notant par une lettre majuscule.

3.1.2 Algèbre de résolution

Pour formaliser la notion de cryptanalyse par boîtes noires, on définit des opérations de manipulation de relations. On utilise principalement deux opérations : la jointure $\bowtie$ et le contrôle γ. On pourra aussi envisager d'autres opérations dans une extension de ce type de cryptanalyse : la réunion $\cup$, la projection π, la sélection σ et la différence $-$.

Une *formule relationnelle* sur un graphe muet $G = (V, E, r)$ est un terme utilisant des opérations $\bowtie$, γ, $\cup$, π, σ et $-$ et les variables V suivant des règles de formation. On attribue à chaque terme F bien formé un schéma $E(F)$. Pour chaque opération, on a une règle de formation portant sur les schémas des entrées définissant celui de la sortie. L'interprétation R du graphe définit de manière naturelle la *sémantique* des formules.

Jointure

Pour deux relations x et y de schémas respectifs $E(x)$ et $E(y)$ quelconques, on note $x \bowtie y$ la relation de schéma $E(x) \cup E(y)$ obtenue en concaténant les tuples compatibles de x et y : si t est un tuple de schéma $E(x) \cup E(y)$, on a

$$t \in x \bowtie y \Longleftrightarrow t|_{E(x)} \in x \text{ et } t|_{E(y)} \in y$$

où $t|_S$ est la restriction de t au schéma S. $\bowtie$ est une opération associative et commutative sur les relations, la relation sur le schéma vide $\emptyset$ est neutre pour $\bowtie$.

Contrôle

Pour une relation x de schéma quelconque et un réel positif ρ, on définit la relation $\gamma_\rho(x)$ de même schéma comme étant un sous-ensemble aléatoire de x dans lequel chaque tuple de x est *pris* avec probabilité $k^{-\rho}$ de manière indépendante, k étant la base considérée. On a

$$t \in \gamma_\rho(x) \Longleftrightarrow t \in x \text{ et } t \text{ pris par } \gamma_\rho.$$

C'est un exemple de sélection non déterministe. Cette opération permet de *contrôler* la taille des relations dans des classes de résolution d'équations probabilistes.

Autres opérations

D'autres opérations de l'algèbre relationnelle pourront être envisagées :

Union. Pour deux relations x et y de même schéma, on définit la réunion $x \cup y$ des ensembles x et y, de même schéma.

Projection. Pour une relation x et un sous-schéma S, on définit la relation $\pi_S(R)$ qui est l'ensemble des tuples de x restreints au schéma S.

Sélection. Pour une relation x et un prédicat $P(t)$ portant sur un tuple t, on définit la relation $\sigma_P(R)$ de même schéma par l'ensemble des tuples de x qui satisfont le prédicat P.

Différence. Pour deux relations x et y de même schéma, on définit la relation $x - y$ obtenu à partir de x en enlevant les tuples de y.

3.2 Résolution de graphes d'équation

3.2.1 Algorithmes de résolution

Définition 5. Soit un graphe d'équation $G = (V, E, r, k, D)$. Un *algorithme de résolution* de G est un algorithme A qui admet en entrée une interprétation R du graphe et qui retourne en sortie une relation de schéma E telle que

$$\forall R \quad A(R) \subseteq \bowtie_{v \in V} R(v).$$

Lorsqu'il y a égalité, on dit que la résolution est *complète*.

A certaines formules relationnelles correspondent un algorithme de résolution. Par exemple, pour des formules constituées uniquement d'opérations $\bowtie$ et γ, le fait de résoudre le graphe d'équation équivaut au fait de contenir tous les sommets.

Définition 6. Soit un graphe d'équation $G = (V, E, r, k, D)$. La classe des formules relationnelles utilisant les opérations f_1, ..., f_n définissant un algorithme de résolution pour G est notée $\mathcal{A}(G, f_1, \ldots, f_n)$.

On étudiera la complexité des graphes vis-à-vis de certaines classes, en moyenne sur la distribution des interprétations possibles.

3.2.2 Distribution des interprétations

Dans l'analyse des algorithmes de résolution, il est nécessaire de choisir une distribution des interprétations d'un graphe d'équation $G = (V, E, r, k, D)$.

Distribution localement uniforme

On considère des distribution présentant des propriétés d'uniformité :

Définition 7. Soit un graphe d'équation $G = (V, E, r, k, D)$. On dit qu'une distribution des interprétations est *localement uniforme* si :

- pour tout sommet v, les événements $t \in R(v)$ ont même probabilité pour tous les tuples t sur schéma de v;
- pour tout tuple de schéma E, les événements $t|_{S(v)} \in R(v)$ sont indépendants pour tous les sommets v, $S(v)$ étant le schéma de v.

On a le lemme suivant :

Lemme 1. *Dans une distribution des interprétations R localement uniforme, pour tout sommet v et tout tuple t de schéma des arêtes adjacentes à v, on a*

$$\underset{R}{Prob}[t \in R(v)] = k^{r(v)-d(v)}.$$

Preuve. Soit p cette probabilité. Elle est indépendante de t. On a

$$\begin{aligned} k^{d(v)}p &= \sum_t \underset{R}{Esp}\left[1_{t\in R(v)}\right] \\ &= \underset{R}{Esp}[Card(R(v))] \\ &= k^{r(v)}. \end{aligned}$$

□

Classes de relations stables

La distribution uniforme de toutes les relations possibles est une distribution localement uniforme. Toutefois, dans un contexte où ces relations représentent des graphes de boîtes de calcul, elles présentent une dépendance fonctionnelle vis-à-vis du schéma constitué des arêtes d'entrée. Il existe donc des relations qui ne sont pas envisageables dans ce contexte. On peut donc s'intéresser à la distribution uniforme des interprétations possibles sur certaines classes de relations.

Soit $\mathcal{R}(S)$ l'ensemble des relations sur le schéma S. Le produit $\prod_{e\in S}\mathcal{S}_{D_e}$ des groupes symétriques sur les domaines D_e agit sur $\mathcal{R}(S)$ par $\sigma * R = \{\sigma(t); t \in R\}$ où

$$\sigma(t) = (\sigma_e(t_e))_{e\in S}$$

si $\sigma = (\sigma_e)_{e\in S}$ et $t = (t_e)_{e\in S}$. On a le lemme suivant :

Lemme 2. *Soit un graphe d'équation $G = (V, E, r, k, D)$. A tout sommet v de V on associe une classe $\mathcal{R}_v$ de relations de rang $r(v)$ sur le schéma des arêtes adjacentes à v stable par l'action des permutations. La distribution uniforme des interprétations sur $\prod_{v\in V}\mathcal{R}_v$ est localement uniforme.*

Preuve. L'indépendance des $t|_{S(v)}$ sur la distribution produit est triviale. Il suffit de montrer que si $\mathcal{R}_v$ est stable par l'action des permutations, le nombre de ses relations R telles que $t \in R$ est indépendant de t.

Si t et t' sont deux tuples sur le schéma de v, et si σ est un produit de permutations telles que $\sigma(t) = t'$, on a

$$t \in R \Longleftrightarrow t' \in \sigma(R)$$

pour tout $R \in \mathcal{R}_v$. Comme σ défini une permutation de $\mathcal{R}_v$, on a le résultat. □

Exemples

Dans un réseau, une boîte de calcul est une fonction du produit des domaines des arêtes entrantes vers le produit des domaines des arêtes sortantes. La classe des relations fonctionnelles sur un schéma $S(v)$ (la dépendance fonctionnelle étant

vis-à-vis d'un sous-schéma particulier) étant stable par l'action des permutations, la distribution uniforme sur toutes les boîtes possibles est localement uniforme.

Dans le chapitre 8, on étudie la notion de relation qui présente des dépendances fonctionnelles vis-à-vis de tous leurs sous-schémas de taille égale au rang : les *multipermutations*. On montrera que la classe de ces relations est également stable par l'action des permutations.

3.2.3 Classes d'algorithmes de résolution

Algorithmes de rang déterminé

Pour une distribution des interprétations R et un algorithme de résolution A, $A(R)$ est une relation aléatoire. On peut envisager son rang moyen.

$$\mathrm{rg}(A) = \log_k \underset{R}{Esp}\, Card(A(R)).$$

Ceci permet de définir la classe suivante :

Définition 8. Soit un graphe d'équation $G = (V, E, r, k, D)$ muni d'une distribution des relations. On note $\mathcal{A}^\rho(G, f_1, \ldots, f_n)$ la sous-classe des algorithmes de $\mathcal{A}(G, f_1, \ldots, f_n)$ de rang moyen au moins ρ.

Ainsi, $\mathcal{A}^0(G, f_1, \ldots, f_n)$ est la classe des algorithmes qui retournent au moins une solution en moyenne.

Algorithmes linéaires

Une formule est dite *linéaire* si chaque nœud binaire admet au moins un fils dont aucun descendant n'est binaire (en général, ce fils doit être une feuille). Ces algorithmes s'écrivent $((\gamma_{a_0}(v_0)P_1\gamma_{a_1}(v_1))P_2 \ldots)P_n\gamma_{a_n}(v_n)$ pour des opérations binaires $P_1, \ldots, P_n$, des entiers $a_1, \ldots, a_n$ et des sommets $v_0, \ldots, v_n$.

Définition 9. Soit un graphe d'équation $G = (V, E, r, k, D)$. La sous-classe des formules linéaires utilisant les opérations $f_1, \ldots, f_n$ est notée $\mathcal{L}(G, f_1, \ldots, f_n)$.

De même, on note $\mathcal{L}^\rho(G, f_1, \ldots, f_n)$ la sous-classe des formules linéaires de rang moyen au moins ρ pour une distribution des relations donnée.

La classe d'algorithmes la plus simple est la classe $\mathcal{L}(G, \bowtie)$ des algorithmes linéaires avec exclusivement des opérations $\bowtie$. Ces algorithmes permettent d'exprimer la notion de *recherche exhaustive* pour la recherche de *toutes* les solutions. La classe $\mathcal{L}(G, \bowtie, \gamma)$ exprime la recherche exhaustive pour la recherche de *quelques* solutions.

Considérons par exemple le réseau (V, E) de la figure 3.1. Le graphe d'équation muet associé donne 1 pour rang aux arêtes et 2 aux sommets représentant

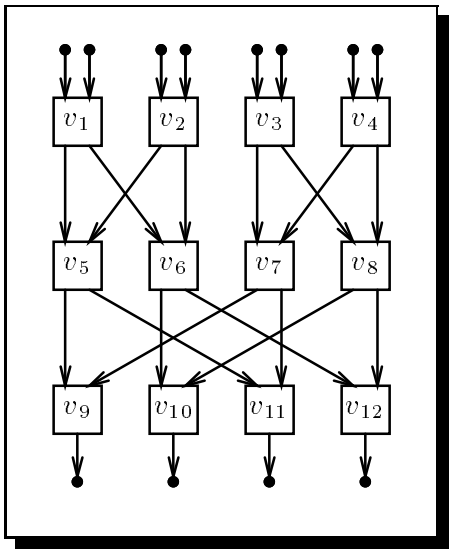

Figure 3.1 : Exemple de primitive.

une boîte de calcul (elles ont deux entrées). Dans le problème du calcul direct de la primitive, on connaît les valeurs d'entrées, donc, on donne 0 pour rang aux sommets d'entrée $v_1^i, \ldots, v_8^i$ (les relations correspondantes possèdent un unique tuple), et 1 aux sommets de sortie $v_1^o, \ldots, v_4^o$ (les relations correspondantes admettent tous les tuples possibles). La jointure successive des relations dans l'ordre défini par

$$v_1^i < \ldots < v_8^i < v_1 < \ldots < v_{12} < v_1^o < \ldots < v_4^o$$

possède à chaque instant un unique tuple dont le schéma augmente jusqu'à être le schéma complet.

Dans le problème de l'inversion, on donne au contraire 0 pour rang aux sommets v_j^o et 1 aux sommets v_j^i. La jointure dans le même ordre obtient, par recherche exhaustive sur toutes les entrées possibles, l'ensemble des tuples solutions. Une même résolution contrôlée permet d'obtenir une seule solution : l'algorithme

$$((((\gamma_{r(v_1^i)}(v_1^i) \bowtie \ldots)\gamma_{r(v_4^i)}(v_4^i)) \bowtie v_5^i) \bowtie \ldots) \bowtie v_4^o$$

obtient en moyenne une solution, comme le montrera un théorème ultérieurement. Intuitivement, cela revient à fixer arbitrairement les 4 premières entrées et à chercher de manière exhaustive les 4 autres pour obtenir les bonnes sorties.

L'intérêt des algorithmes arborescents est de modéliser les attaques de type anniversaire. Ainsi, dans le problème de recherche de collisions, on utilise un graphe dupliqué (voir figure 3.2). Les entrées et les sorties ont pour rang 1. Soient les termes

$$t \quad = \quad ((((\gamma_{r(v_1^i)}(v_1^i) \bowtie \ldots)\gamma_{r(v_6^i)}(v_6^i)) \bowtie v_7^i) \bowtie \ldots) \bowtie v_4^o$$

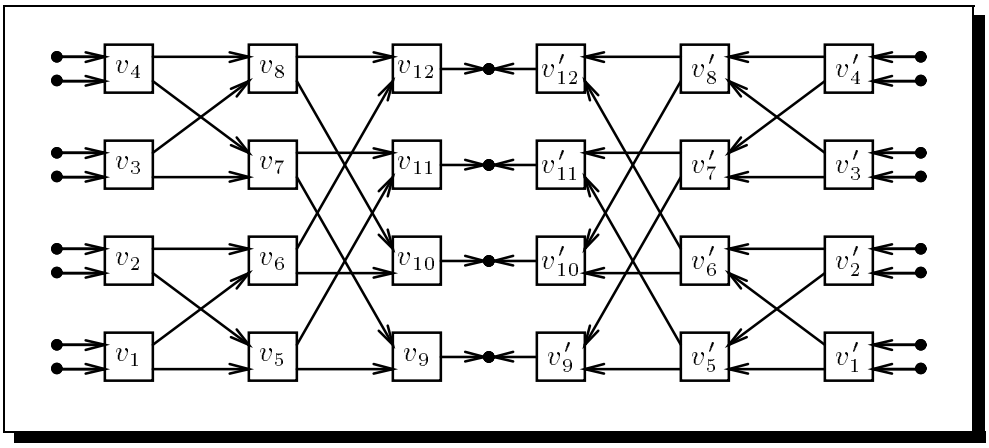

Figure 3.2 : Recherche de collisions.

$$t' \quad = \quad ((((\gamma_{r(v_1^i)}(v'^i_1) \bowtie \ldots)\gamma_{r(v_6^i)}(v'^i_6)) \bowtie v'^i_7) \bowtie \ldots) \bowtie v'^o_4.$$

Le terme t fournit une liste de k^2 tuples possibles (k étant la base) en ne laissant libre que les deux entrées v_7^i et v_8^i. Le terme t' fournit une liste analogue pour la partie droite. L'algorithme $t \bowtie t'$ modélise le paradoxe des anniversaires, qui donne en moyenne une solution. Ainsi, $\mathcal{A}(G, \bowtie, \gamma)$ généralise les attaques de type anniversaire.

On note que la figure 3.2 ne modélise le problème de la recherche de collisions que si la distribution des interprétations impose que les relations associées à gauche et à droite soient les mêmes. Si elles sont indépendantes, la figure modélise le problème de la recherche de pinces sur deux fonctions différentes.

3.3 Complexité des algorithmes

Si un modèle de calcul permet de définir la complexité d'un algorithme, on définit la complexité d'un graphe d'équation suivant une classe par la plus petite complexité en moyenne d'un algorithme de cette classe :

Définition 10. Soient un graphe d'équation $G = (V, E, r, k, D)$, une distribution des relations, une classe de résolution $\mathcal{A}$ et un modèle de calcul. On définit la *complexité logarithmique* du graphe d'équation

$$pC[\mathcal{A}] = \min_{A \in \mathcal{A}} pC(A)$$

où

$$pC(A) = \log_k \underset{R}{Esp}\, C(A, R)$$

où $C(A, R)$ est la complexité de l'algorithme A sur l'interprétation R.

On s'intéressera principalement à la complexité vis-à-vis des classes d'algorithmes qui généralisent les notions de recherche exhaustive, de paradoxe des anniversaires, et d'attaque dans le milieu : $pC[\mathcal{L}(G, \bowtie, \gamma)]$ et $pC[\mathcal{A}(G, \bowtie, \gamma)]$.

3.3.1 Modèle de calcul

On utilise un modèle de calcul "idéal", c'est-à-dire un modèle qui donne une complexité inférieure à la réalité. Cela permettra de donner des bornes inférieures.

Une opération de calcul relationnel devant fournir un résultat, la complexité de son calcul est supérieure au cardinal du résultat. Par exemple, le calcul de $x \bowtie y$ nécessite en général l'emploi de tables de hachage, ou de tri qui ne peuvent se faire qu'en temps $\Omega(n \log n)$ où n est la taille des entrées. La taille de la sortie est donc un bon estimateur de la complexité : on considère que la complexité logarithmique de $x \bowtie y$ est égale au rang de $x \bowtie y$.

On propose la définition suivante :

Définition 11. La *complexité* d'une formule relationnelle A est :

$$pC(A) = \max_{B \leq A} \mathrm{rg}(B)$$

où le maximum est étendu sur toutes les sous-formules B de la formule A.

Ainsi, on a la proposition immédiate suivante :

Proposition 3. *Etant donnés un graphe d'équation $G = (V, E, r, k, D)$, une distribution des interprétations, une classe de résolution $\mathcal{A}$ de formules relationnelles, on a :*

$$pC[\mathcal{A}] = \min_{A \in \mathcal{A}} \max_{B \leq A} \mathrm{rg}(B).$$

3.3.2 Forme quadratique du graphe d'équation

Le problème de résolution d'un système d'équations se ramène à la recherche d'une stratégie de résolution qui progresse en utilisant le plus petit nombre d'inconnues possible. Si l'on appelle *périmètre* le nombre d'inconnues dans un ensemble d'équations qui interviennent également dans des équations extérieures à cet ensemble, on observe une analogie avec les problèmes isopérimétriques en géométrie analytique.

En théorie des graphe, on définit une forme quadratique associée au graphe qui intervient dans de nombreuses applications, comme le dénombrement des arbres recouvrant [76]. Comme elle rappelle les propriété de l'opérateur de Laplace dans le calcul différentiel, certains chercheurs l'appellent *laplacien* du graphe. De manière analogue, pour les graphes d'équation, on définit :

Définition 12. Soit $G = (V, E, r)$ un graphe d'équation muet. La *forme quadratique* de G, notée également G, est définie sur l'algèbre libre $\mathbb{Z}[V]$ par

$$G\left(\sum_{v\in V} \lambda_v.v\right) = \sum_{v\in V} \lambda_v^2 r(v) - \sum_{\{v,w\}\in E} \lambda_v\lambda_w r(\{v,w\})$$

pour tous réels λ_v.

La matrice de G indexée sur V s'écrit :

$$G_{v,w} = \begin{cases} r(v) & \text{si} \quad v = w \\ -\frac{1}{2}r(\{v,w\}) & \text{si} \quad v \neq w \end{cases}$$

en convenant que le rang d'une arête inexistante est 0.

Dans la suite, on confondra une partie A de V avec la somme de ses éléments dans $\mathbb{Z}[V]$. Ainsi, on a

$$G(A) = \sum_{v\in A} r(v) - \sum_{\substack{v,w\in A \\ \{v,w\}\in E}} r(\{v,w\}).$$

L'intérêt de la forme quadratique est motivé par le théorème suivant :

Théorème 4. *Soit un graphe d'équation $G = (V, E, r, k, D)$ muni d'une distribution localement uniforme des relations. Pour toute formule A avec des opérations $\bowtie$, γ et des sommets de V, le rang moyen de A est :*

$$\mathrm{rg}(A) = G\left(\sum_{v\in A} v\right) - \sum_{\gamma_a\in A} a$$

où les sommes sont étendues aux occurrences des sommets et des opérations γ_a dans la formule A.

On remarque que ce résultat est indépendant de la base k. La complexité dans $\mathcal{A}(G, \bowtie \gamma)$ est donc indépendante de la base k. C'est une notion purement liée à la structure du graphe d'équation muet.

Preuve. Pour une formule B, on note $E(B)$ son schéma. Pour un sommet v, $E(v)$ est l'ensemble de ses arêtes adjacentes. Pour tout tuple t de schéma S et toute partie T de S, on note $t|_T$ le tuple restreint à T.

On remarque que pour toutes formules B et C et tout tuple aléatoire T de schéma $E(B) \cup E(C)$, on a

$$\{T \in B \bowtie C\} = \{T|_{E(B)} \in B\} \cap \{T|_{E(C)} \in C\}.$$

De même, pour toute formule $\gamma_a(B)$ et tout tuple T de schéma $E(B)$, on a

$$\{T \in \gamma_a(B)\} = \{T \in B\} \cap \{T \text{ pris par } \gamma_a / T \in B\}.$$

Ainsi, pour tout tuple T de schéma $E(A)$, on a

$$\{T \in A\} = \bigcap_{v \in A} \{T|_{E(v)} \in v\} \cap \bigcap_{\gamma_a(B) \subseteq A} \{T|_{E(B)} \text{ pris par } \gamma_a / T \in B\}.$$

On note que ces événements sont indépendants.

Pour toute relation X de distribution localement uniforme et pour tout tuple T de schéma $E(X)$, on a $\log_k(Prob[T \in B]) = rg(B) - d(E(B))$, donc

$$Prob[T \in A] = \prod_{v \in A} k^{r(v)-d(v)} \times \prod_{\gamma_a(B) \subseteq A} k^{-a}$$

d'où

$$\begin{aligned} rg(A) &= d(E(A)) + \sum_{v \in A} (r(v) - d(v)) - \sum_{\gamma_a \in A} a \\ &= \sum_{v \in A} r(v) - \sum_{v,w \in A} r(\{v,w\}) - \sum_{\gamma_a \in A} a \\ &= G\left(\sum_{v \in A} v\right) - \sum_{\gamma_a \in A} a. \end{aligned}$$

□

On note, dans le cas où le rang des arêtes est 1 et le rang des sommets est la moitié de leur degré (intuitivement, cela signifie que pour chaque boîte du graphe, le nombre d'entrées est égal au nombre de sorties), pour tout A, $G(A)$ est égal à la moitié du nombre d'arêtes au bord de A (que l'on appelle *périmètre* de A). On traitera ce cas particulier dans l'étude de la *réversibilité locale*.

Conclusion

La notion de graphe d'équation offre un cadre formel à l'étude de la complexité des spécifications des primitives cryptographiques, donc à l'étude de leur sécurité. Elle permet de définir des classes d'algorithmes qui généralisent les méthodes usuelles telles la recherche exhaustive et l'utilisation du paradoxe des anniversaires. On constate de plus que la complexité est fortement liée aux propriétés d'une forme quadratique définie intrinsèquement par la structure géométrique de la primitive. Dans toute la suite de ce mémoire, on tente d'élaborer une théorie de la sécurité des primitives cryptographiques dans ce cadre.

Chapitre 4

Complexité des graphes d'équation

Les graphes d'équation définis au chapitre 3 offrent un cadre formel aux spécifications des primitives cryptographiques définies par un réseau de calcul. La complexité d'un graphe représente le coût de la résolution de l'équation spécifiée par la primitive lorsque ses boîtes sont aléatoires. Cela représente donc la sécurité apportée par la structure du graphe.

On étudie dans ce chapitre la complexité des graphes d'équation vis-à-vis des classes d'algorithmes linéaires ou arborescentes utilisant les opérations $\bowtie$ et γ définies dans le chapitre 3. Dans la suite, G désigne un graphe d'équation (V, E, r, k, D). On considérera une distribution localement uniforme sur ses interprétations. Une forme quadratique associée à G est définie (voir la définition 12). On la note également G. Elle joue un rôle fondamental dans la complexité du graphe. On voit notamment que la complexité, dans certains cas, est proportionnelle à la seconde valeur propre de G.

4.1 Cas de réversibilité locale

On propose la définition suivante :

Définition 13. Un graphe d'équation muet $G = (V, E, r)$ est dit *localement réversible* si pour tout sommet v, le degré de v est égal à $2r(v)$.

On rappelle que dans un graphe d'équation muet, le degré d'un sommet est la somme des rangs des arêtes adjacentes. Cette définition modélise la notion de boîte réversible, qui a autant d'entrées que de sorties.

Les propriétés de la forme quadratique G qui correspond au *laplacien* peuvent être étudiées avec les techniques utilisées dans la théorie des *graphes d'expansion*. Ces graphes, liés à de nombreux problèmes combinatoires, apparaissent dans la

littérature sous plusieurs noms différents. On utilise ici le lien observé entre le *rapport d'expansion* du graphe et le spectre de son laplacien [80, 75, 74].

En s'inspirant de ces techniques, on montre :

Théorème 5. *Pour un graphe d'équation muet $G = (V, E, r)$ connexe localement réversible, la forme quadratique G est positive. Si A est une partie de V de cardinal c, on a :*

$$G(A) \geq \lambda_2 c \left(1 - \frac{c}{n}\right)$$

où n est le cardinal de V et λ_2 est la plus petite valeur propre non nulle de G.

Preuve. On note M la matrice de la forme quadratique G. Si X est un vecteur propre, et si $M.X = \ell.X$, soit i l'indice de la plus grande coordonnée de X (en valeur absolue). On a :

$$\begin{aligned} \ell &= M_{i,i} + \sum_{j \neq i} M_{i,j} \frac{X_j}{X_i} \\ &\geq M_{i,i} - \sum_{j \neq i} |M_{i,j}| \\ &= \sum_i M_{i,j} = 0 \end{aligned}$$

donc $\ell \geq 0$: G est positive. De plus, on note que si $\ell = 0$, tous les $\frac{X_j}{X_i}$ valent 1 dés que $M_{i,j}$ est non nul, donc X est constant sur la composante connexe de i dans le graphe G, donc sur V.

Soit $v_1, \ldots, v_n$ une base orthonormale de vecteurs propres pour G telle que $M.v_i = \lambda_i.v_i$ et $\lambda_1 = 0$. v_1 est donc le vecteur constant dont les composantes sont $\pm\frac{1}{\sqrt{n}}$.

Si A est un vecteur caractéristique (avec des 0 et des 1 uniquement) de poids c, on a :

$$G(A) = \sum_{i=1}^{n} \lambda_i (A \cdot v_i)^2 \geq \lambda_2 \left(\sum_{i=2}^{n} (A \cdot v_i)^2 \right).$$

Donc, on a $G(A) \geq \lambda_2.((A \cdot A) - (A \cdot v_1)^2)$. On a $A \cdot A = c$ et $(A \cdot v_1)^2 = \frac{c^2}{n}$, ce qui montre le résultat. □

On note que le théorème 5 se généralise à un graphe quelconque en :

Théorème 6. *Pour un graphe d'équation muet $G = (V, E, r)$, si n est le cardinal de V, si $\lambda_1 \leq \ldots \leq \lambda_n$ est la liste des valeurs propres de la forme quadratique G, si m est un entier tel que $\lambda_m \geq 0$ et si $v_1, \ldots, v_n$ est une base orthonormale de vecteurs propres adaptée à la liste, pour toute partie A de V de cardinal c, on a :*

$$G(A) \geq c \left[\lambda_m - c \sum_{i=1}^{m-1} (\lambda_m - \lambda_i) ||v_i||_\infty^2 \right]$$

où $||.||_\infty$ est la plus grande coordonnée en valeur absolue.

Dans le cas où $m = 2$, $\lambda_1 = 0$ et v_1 est le vecteur constant, on retrouve le théorème 5. Dans le cas où $m = 1$ et $\lambda_1 > 0$, on obtient la minoration triviale $G(A) \geq \lambda_1 c$. Cette inégalité peut cependant se révéler trop grossière.

Preuve. On a

$$\begin{aligned} G(A) &= \sum_{i=1}^{n} \lambda_i (A \cdot v_i)^2 \\ &\geq \sum_{i=1}^{m-1} \lambda_i (A \cdot v_i)^2 + \lambda_m \sum_{i=m}^{n} (A \cdot v_i)^2 \\ &= \sum_{i=1}^{m-1} \lambda_i (A \cdot v_i)^2 + \lambda_m c - \lambda_m \sum_{i=1}^{m-1} (A \cdot v_i)^2 \\ &= \lambda_m c - \sum_{i=1}^{m-1} (\lambda_m - \lambda_i)(A \cdot v_i)^2 \end{aligned}$$

et l'inégalité $(A \cdot v_i)^2 \leq c^2 ||v_i||_\infty^2$ achève la preuve. □

4.2 Résolution linéaire

4.2.1 Résolution dans $\mathcal{L}(G, \bowtie)$

Algorithmes de $\mathcal{L}(G, \bowtie)$

Si A est un algorithme de $\mathcal{L}(G, \bowtie)$, il s'écrit $A = ((v_1 \bowtie v_2) \bowtie \ldots) \bowtie v_n$. On remarque que si l'on supprime les étages $\bowtie v_i$ tels que le sommet v_i est égal à un sommet v_j avec $j < i$, l'algorithme donne le même résultat avec la même complexité. Donc, dans l'étude de $\mathcal{L}(G, \bowtie)$, on peut se ramener à des formules sans répétitions de sommet.

De telles formules sont entièrement définies par un ordre total sur les sommets de V. L'algorithme progresse en explorant successivement chaque sommet et en gérant l'ensemble des solutions partielles sur tous les sommet déjà visités. La complexité en temps de calcul est égale à la taille maximale de cet ensemble de solutions partielles au cours de la résolution.

On note qu'en utilisant le *back tracking*, on peut réellement implémenter de tels algorithmes A avec la complexité en temps $\Omega(k^{pC(A)})$ et $O(k^{pC(A)}n)$ et une complexité en espace $\Theta(n)$ où n est la taille du graphe. Si V est totalement ordonné par $v_1 < \ldots < v_n$, la procédure de la figure 4.1 appelée par $\textsc{Solution}_1(1, \emptyset)$ (où $\emptyset$ est le tuple de schéma vide) affiche la liste des solutions. C'est la notion de recherche exhaustive dans une résolution complète.

```
SOLUTION_1(i, t)
    Si i = n + 1 affiche t
    Sinon
        Pour tout u ∈ R(v_i)
            Si t et u sont compatibles, SOLUTION_1(i + 1, t ⋈ u)
        Fin pour
    Fin si
Fin SOLUTION_1.
```

Figure 4.1 : Algorithme de résolution linéaire.

Complexité dans $\mathcal{L}(G, \bowtie)$

Le théorème 4 conduit à la proposition suivante :

Proposition 7. *Pour un graphe d'équation muet $G = (V, E, r)$, une résolution dans $\mathcal{L}(G, \bowtie)$ est caractérisée par un ordre total $\leq$ sur V. Pour toute distribution localement uniforme des interprétations, sa complexité logarithmique est :*

$$\begin{aligned} pC(\leq) &= \max_{u \in V} G\left(\sum_{v \leq u} v\right) \\ &= \max_{u \in V}\left(\sum_{v \leq u} r(v) - \sum_{\substack{\{v,w\} \in E \\ v,w \leq u}} r\left(\{v, w\}\right)\right). \end{aligned}$$

Exemple

La figure 4.2 illustre un exemple de graphe d'équation. On suppose que les arêtes sont de rang 1. Le rang des sommets est indiqué en légende. On repère les sommets $v_{i,j}$ par leurs coordonnées comme dans un tableau (par exemple, le sommet situé en haut à droite est $v_{0,7}$). La figure 4.3 représente une résolution. Le graphique trace l'évolution de la taille de l'ensemble des solutions partielles : le temps est en abscisse, avec l'indication du sommet $v_{i,j}$ qui est visité, et la complexité est en ordonnée. Quelques instants particuliers de l'algorithme (de A à F) sont dessinés. Les sommets coloriés en gris sont des sommets déjà visités.

En imaginant que c'est le réseau de calcul d'une fonction f admettant des opérations d'entrée sur la couche $i = 0$ et des opérations de sortie sur la couche

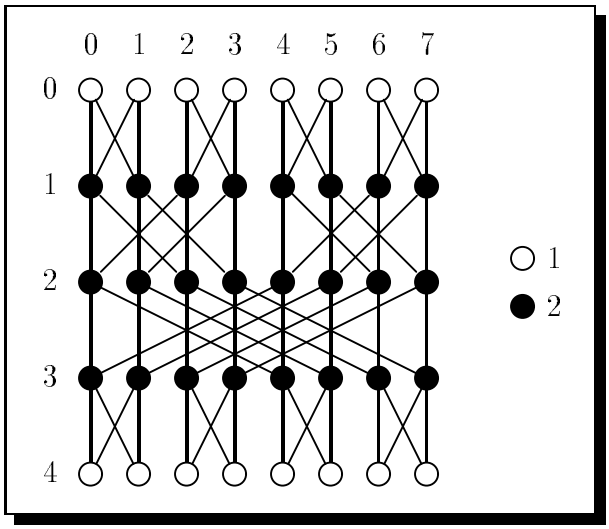

Figure 4.2 : Exemple de graphe d'équation.

$i = 4$, ce graphe d'équation est celui de l'inversion de f : c'est le graphe de l'équation $f(x) = a$.

Jusqu'à l'instant A, la complexité augmente, c'est la notion de recherche exhaustive qui consiste à "deviner" l'état des sommets de $v_{0,1}$ à $v_{0,3}$ (donc, une partie de x). Ensuite, on peut visiter le graphe avec complexité constante jusqu'à l'instant B. En effet, on connaît 2 arêtes du sommet $v_{1,0}$ qui est exactement de rang 2 (dans [73], on dit que le sommet est *déterminé*), donc on peut le visiter avec une complexité nulle.

A l'instant B, il n'y a plus de sommet déterminé. Seule une arête de $v_{3,0}$, qui est de rang 2 est résolue (on dit que le sommet $v_{3,0}$ est *sous-déterminé*). La visite de $v_{3,0}$ augmente donc la complexité à 5 (instant C).

A l'instant C, $v_{4,0}$ et $v_{4,1}$ sont déterminés. Un fois visité, on connaît 3 arêtes de $v_{3,1}$ qui est de rang 2 (on dit qu'il est *sur-déterminé*). Des solutions partielles sont donc rejetées à ce niveau, et le *back tracking* redescend à une complexité 4.

Le traitement du reste des couches $i = 3$ et $i = 4$ est semblable au traitement B-D. A l'instant E, les sommets restant des couches de $i = 4$ à $i = 1$ se déterminent successivement jusqu'à l'instant F. Enfin, les derniers sommets surdéterminés permettent d'éliminer presque toutes les solutions pour n'en retenir qu'une : celle de l'équation $f(x) = a$.

On peut comparer ce procédé de résolution à un système mécanique décrit par l'ensemble S des sommets résolus. On souhaite passer de l'état $S = \emptyset$ à l'état $S = V$, mais chaque état est caractérisé par un *potentiel* $G(S)$ qu'il faut atteindre. Ainsi, les états $S = \emptyset$ et $S = V$ sont séparés par une *barrière de potentiel* dont le

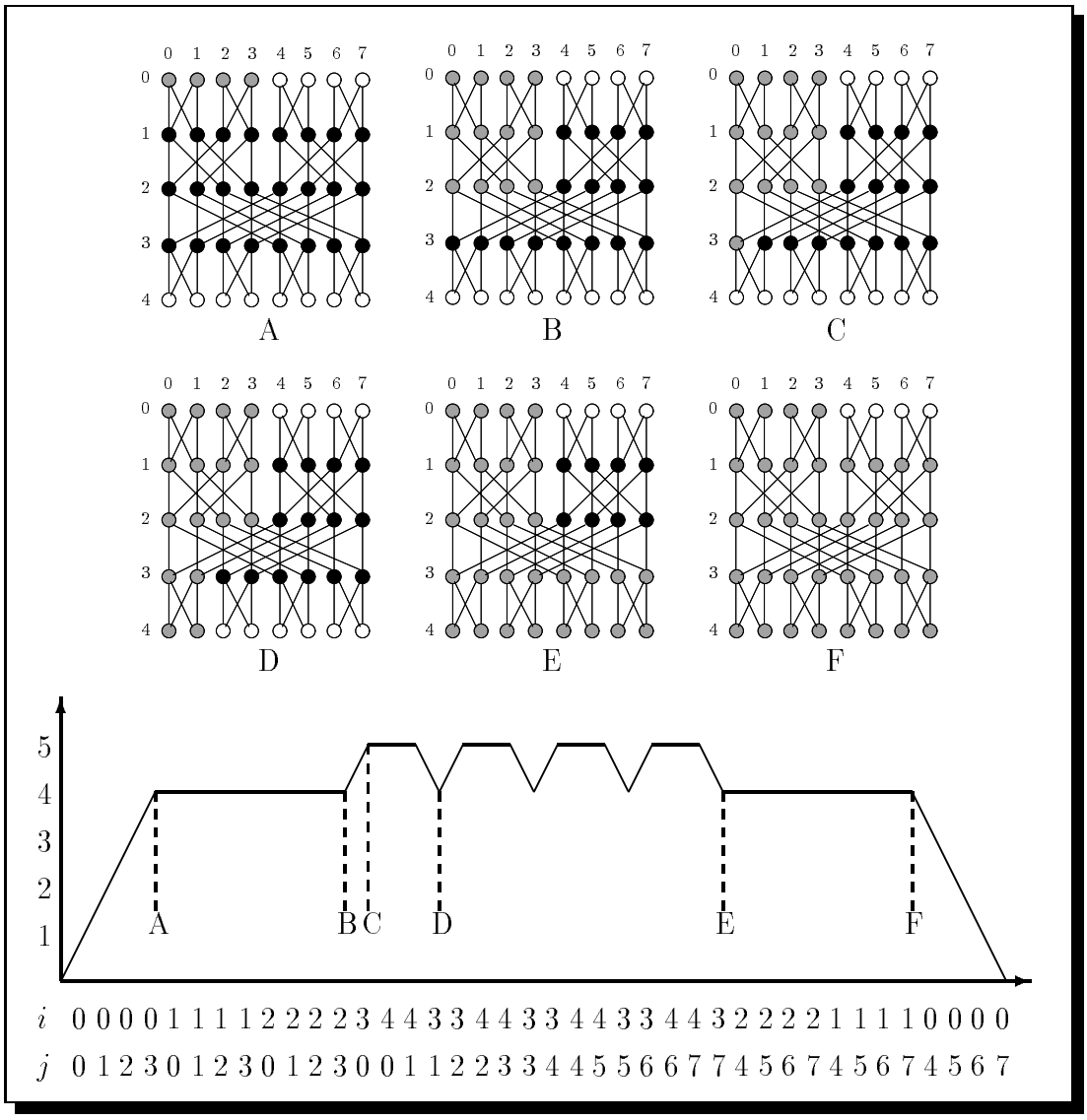

Figure 4.3 : Exemple de résolution.

plus petit col est de hauteur $pC(G)$. Le problème que l'on se pose est de trouver le chemin qui passe par ce col.

Un théorème permet d'estimer simplement la complexité linéaire d'un graphe :

Proposition 8. *Pour tout graphe d'équation muet $G = (V, E, r)$, on a :*

$$pC[\mathcal{L}(G, \bowtie)] \geq \max_{c \leq Card(V)} \min_{\substack{A \subseteq V \\ Card(A) = c}} G(A).$$

Preuve. Soit un ordre $\leq$ sur les sommets qui définit un algorithme optimal : $pC(\leq) = G\left(\sum_{w \leq v} w\right)$. Soit v le sommet qui vient d'être résolu à un instant où $pC(\leq) = G(\{w; w \leq v\})$. Si c est le nombre de w inférieurs à v, $pC(\leq)$ s'exprime bien comme étant $G(A)$ pour un A de cardinal c. □

Cette proposition et le théorème 5 entraînent :

Théorème 9. *Pour un graphe d'équation muet $G = (V, E, r)$ connexe localement réversible, si λ_2 est la seconde valeur propre de la forme quadratique G, on a :*

$$pC[\mathcal{L}(G, \bowtie)] \geq \frac{\lambda_2}{4} Card(V).$$

Preuve. Le maximum de $c\left(1 - \frac{c}{n}\right)$ lorsque c varie entre 0 et n est $\frac{n}{4}$. □

4.2.2 Résolution dans $\mathcal{L}(G, \bowtie, \gamma)$

La complexité dans $\mathcal{L}(G, \bowtie)$ est adaptée au problème de la résolution complète : la recherche de *toutes* les solutions. Le théorème 4 estime le rang moyen de l'ensemble des solutions à $G(V)$. Si l'on ne cherche qu'une solution, en se limitant artificiellement à des relations plus petites, on peut diminuer la complexité. C'est le rôle des opérations γ.

Réduction d'algorithmes dans $\mathcal{A}(G, \bowtie, \gamma)$

Le lemme suivant permet de comprendre le rôle du contrôle dans $\mathcal{A}(G, \bowtie, \gamma)$:

Lemme 10. *Etant donné un graphe d'équation G, si A est une formule relationnelle de $\mathcal{A}(G, \bowtie, \gamma)$ (respectivement $\mathcal{L}(G, \bowtie, \gamma)$), il existe une nouvelle définition du rang r' telle que $r'(v) \leq r(v)$ pour tout sommet v et $r'(e) = r(e)$ pour toute arête e et une formule A' dans $\mathcal{A}(G', \bowtie)$ (respectivement $\mathcal{L}(G', \bowtie)$) où $G' = (V, E, r', k, D)$ tels que $rg(A') = rg(A)$ et $pC(A') \leq pC(A)$.*

Ceci revient à dire que l'on peut ramener le contrôle effectué au cours de la résolution à un contrôle sur les relations de départ, en redéfinissant le rang des sommets. On dit que A' est une formule *réduite* de A.

Preuve. En parcourant la formule A de la racine vers les feuilles, on peut faire descendre les opérations γ_a. En effet, si $B = \gamma_a(\gamma_b(C))$ est une sous-formule de A, on peut la remplacer par $\gamma_{a+b}(C)$. Si $B = \gamma_a(C \bowtie D)$ est une sous-formule de A, le théorème 4 montre que l'on a $rg(\gamma_b(C) \bowtie \gamma_{a-b}(B)) = rg(\gamma_a(C \bowtie D))$, donc, on peut remplacer B par $\gamma_b(C) \bowtie \gamma_{a-b}(B)$ pour tout b. On note que ces deux procédés de réécriture ne modifient pas la complexité. On obtient donc une formule A_0 avec des opérations γ uniquement à la racine, de même rang et même complexité.

Cette nouvelle formule effectue la jointure de plusieurs relations $\gamma_a(v)$. Si un sommet a de multiples occurrences, on constate que $\bowtie_i \gamma_{a_i}(v)$ a la même distribution que $\gamma_a(v)$ où $a = \sum_i a_i$, ce qui revient à réduire le rang du sommet v avant de l'utiliser dans une formule sans opérations γ. On peut donc remplacer la formule A_0 sur G par une formule A' sur G' où A' est obtenue de A en supprimant les opérations γ, et G' est obtenu de G en diminuant le rang des sommets de tous les contrôles indiqués par A_0. Cette dernière opération peut éventuellement diminuer la complexité, car le rang de certaines sous-formules de A_0 est diminué dans A', mais le rang de A' est bien celui de A. □

Réduction d'algorithmes dans $\mathcal{L}(G, \bowtie, \gamma)$

Dans le cas de la classe $\mathcal{L}(G, \bowtie, \gamma)$, on obtient une formule du type

$$((\gamma_{c_1}(v_1) \bowtie \gamma_{c_2}(v_2)) \bowtie \ldots) \bowtie \gamma_{c_n}(v_n).$$

Un tel algorithme peut s'implémenter par la procédure illustrée sur la figure 4.4.

On peut poursuivre la réduction :

Lemme 11. *Etant donné un graphe d'équation G, si A est une formule relationnelle de $\mathcal{L}(G, \bowtie, \gamma)$, il existe une formule A' dans $\mathcal{L}(G, \bowtie, \gamma)$ telle que $rg(A') = rg(A)$ et $pC(A') \leq pC(A)$ et telle que A' s'écrit*

$$\begin{aligned}(\ldots((\quad & ((\ldots(\gamma_{r(v_1)}(v_1) \bowtie \gamma_{r(v_2)}(v_2)) \bowtie \ldots) \bowtie \gamma_{r(v_{p-1})}(v_{p-1})) \\ & \bowtie \gamma_{r(v_p)-a}(v_p)) \bowtie v_{p+1}) \bowtie \ldots) v_n\end{aligned}$$

avec $0 \leq a \leq r(v_p)$.

Une telle formule A' revient à choisir arbitrairement un seul tuple des premiers sommets $v_1, \ldots, v_{p-1}$ dans l'ordre de résolution, à prendre k^a tuples de v_p, et poursuivre la résolution normalement. On dit que A' est une formule *réduite* de $\mathcal{L}(G, \bowtie, \gamma)$. Ce lemme permet, dans l'étude de $\mathcal{L}(G, \bowtie, \gamma)$, de se limiter à l'étude des formules réduites. Il exprime que tout algorithme contrôlé se ramène à un algorithme réduit.

Preuve. D'après le lemme 10, on peut se ramener au cas d'une formule (qui reste linéaire) A' avec des opérations $\bowtie$ sur un graphe avec un nouveau contrôle r'.

```
SOLUTION_2(i, t)
    Si i = n + 1 affiche t
    Sinon
        Pour tout u ∈ R(v_i)
            Avec proba 1/k^{c_i}, fait
                Si t et u sont compatibles, SOLUTION_2(i + 1, t ⋈ u)
            Fin avec
        Fin pour
    Fin si
Fin SOLUTION_2.
```

Figure 4.4 : Algorithme de résolution linéaire contrôlé.

Dans cette formule, la répétition de tout sommet v est inutile, car une fois que la relation du sommet v est jointe, la joindre à nouveau est sans effet. Supprimer les occurrences multiples ne change donc pas la complexité ni le rang de la formule. Une telle formule revient à une formule du type

$$((\ldots(\gamma_{r(v_1)-r'(v_1)}(v_1) \bowtie \gamma_{r(v_2)-r'(v_2)}(v_2)) \bowtie \ldots) \bowtie \gamma_{r(v_n)-r'(v_n)}(v_n))$$

sur le graphe G.

Ainsi, on a un ordre total $v_1 < \ldots < v_n$ sur les sommets, et une fonction r' de V dans $\mathbb{N}$ qui définit le contrôle sur les sommets. On a $r'(v) \leq r(v)$ pour tout v. La complexité de la formule s'écrit

$$pC(A) = \max_{u \in V} \left(\sum_{v \leq u} r'(v) - \sum_{\substack{\{v,w\} \in E \\ v,w \leq u}} r\left(\{v,w\}\right) \right).$$

Pour montrer le lemme, il suffit de montrer que l'on peut remplacer la fonction r' par une fonction r'' telle que $r''(v_i) = 0$ pour $i < p$ et $r''(v_i) = r(v_i)$ pour $i > p$ et qui donne le même rang final, c'est-à-dire telle que $\sum_{i=1}^{n} r''(v_i) = \sum_{i=1}^{n} r'(v_i)$, sans augmenter la complexité.

Soit v_p le plus grand sommet tel que $\sum_{i=p}^{n} r(v_i) \geq \sum_{i=1}^{n} r'(v_i)$. On définit $r''(v_i) = 0$ si $i < p$, $r''(v_i) = r(v_i)$ si $i > p$ et

$$r''(v_p) = \sum_{i=1}^{n} r'(v_i) - \sum_{i=p+1}^{n} r(v_i).$$

On a donc une fonction r'' positive, plus petite en tout point que r, de même somme que r'. Pour $j < p$, on a $\sum_{i=1}^{j} r''(v_i) = 0 \leq \sum_{i=1}^{j} r'(v_i)$. Pour $j \geq p$, on a

$$\sum_{i=1}^{j} r''(v_i) = \sum_{i=1}^{n} r'(v_i) - \sum_{i=j+1}^{n} r(v_i) \leq \sum_{i=1}^{j} r'(v_i)$$

donc la complexité en remplaçant r' par r'' est plus faible. □

Un algorithme réduit de $\mathcal{L}(G, \bowtie, \gamma)$ consiste simplement à considérer un algorithme de $\mathcal{L}(G', \bowtie)$ où $G' = (V, E, r', k, D)$, où r' est obtenu à partir de r en diminuant la valeur du rang de certains sommets par l'opération de contrôle, les premiers contrôles étant nuls.

Complexité dans $\mathcal{L}(G, \bowtie, \gamma)$

On a :

Théorème 12. *Pour tout graphe d'équation muet $G = (V, E, r)$, quelle que soit l'interprétation de distribution localement uniforme considérée, on a :*

$$pC[\mathcal{L}^{\rho}(G, \bowtie, \gamma)] - \rho = pC[\mathcal{L}(G, \bowtie)] - G(V)$$

pour tout ρ.

On note que pour un algorithme A donnant un résultat de rang ρ, $pC(A) - \rho$ est sa *complexité relative*, c'est-à-dire le coût relatif pour obtenir une solution. En terme de complexité relative, le théorème signifie que l'utilisation des opérations γ permet de fractionner l'effort d'un algorithme de $\mathcal{L}(G, \bowtie)$ pour obtenir un algorithme de même complexité relative, avec un résultat de taille voulue.

Preuve. D'après le lemme 11, la complexité optimale dans la classe $\mathcal{L}^{\rho}(G, \bowtie, \gamma)$ est atteinte par des formules réduites.

Une formule réduite de $\mathcal{L}(G, \bowtie, \gamma)$ est équivalente à une formule de $\mathcal{L}(G, \bowtie)$ dans laquelle la fonction r est remplacée par une fonction r' plus petite (sur les sommets). On note que la complexité relative dans $\mathcal{L}(\bowtie)$ s'écrit :

$$pC(\leq) - G(V) = \max_{u \in V} \left(\sum_{\substack{\{v,w\} \in E \\ v > u \text{ ou } w > u}} r(\{v, w\}) - \sum_{v > u} r(v) \right),$$

donc le contrôle augmente la complexité relative. On a donc :

$$pC[\mathcal{L}^{\rho}(G, \bowtie, \gamma)] - \rho \geq pC[\mathcal{L}(G, \bowtie)] - G(V).$$

Si $(((v_1 \bowtie v_2) \bowtie v_3) \bowtie \ldots) \bowtie v_n$ est une formule de complexité optimale dans $\mathcal{L}(G, \bowtie)$, la construction utilisée dans la preuve du lemme 11 permet de contrôler cette formule pour obtenir le rang ρ avec la complexité $pC[\mathcal{L}(G, \bowtie)] - G(V) + \rho$. □

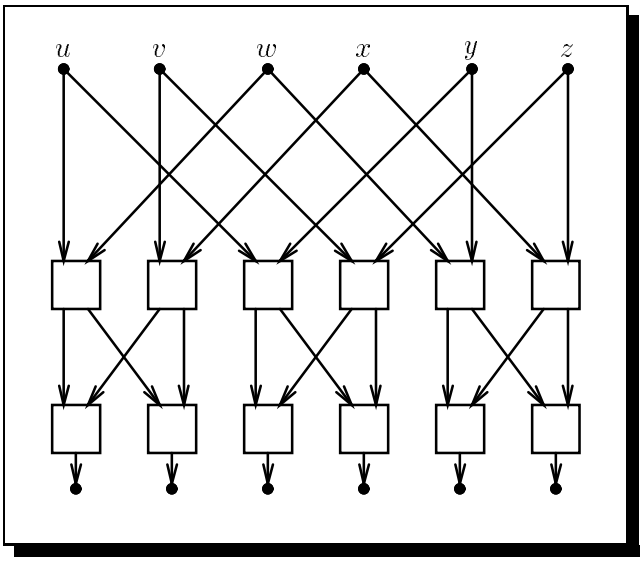

Figure 4.5 : Exemple de primitive.

Des théorèmes 9 et 12, on en déduit :

Théorème 13. *Pour tout graphe d'équation muet $G = (V, E, r)$ connexe et localement réversible, si λ_2 est la seconde valeur propre de la forme quadratique G, on a :*

$$pC[\mathcal{L}^{\rho}(G, \bowtie, \gamma)] \geq \frac{\lambda_2}{4} Card(V) - G(V) + \rho.$$

4.3 Résolution arborescente

4.3.1 Intérêt de l'arborescence

La notion d'algorithme arborescent est nécessaire dés que l'on souhaite modéliser l'emploi du paradoxe des anniversaires dans la recherche de collisions ou de pinces. Toutefois, dans le problème de l'inversion d'une fonction avec autant d'entrées que de sorties, on peut penser *a priori* que l'arborescence n'apporte rien de plus que la linéarité du point de vue de la complexité. Ceci est faux comme le montre l'exemple de la fonction illustrée sur la figure 4.5.

On cherche à inverser cette fonction pour une valeur donnée des sorties. Pour tout choix du couple (u, w), on peut résoudre le tier gauche du graphe et obtenir un ensemble de quadruplets possibles (u, v, w, x) de rang 2. Parallèlement, pour tout couple (w, y), on résout le tier droite et on obtient un ensemble de quadruplets (w, x, y, z) de rang 2. La jointure de ces deux ensembles (en hachant sur les valeurs (w, x)) est un ensemble de valeurs (u, v, w, x, y, z) solutions des

tiers gauches et droits. On peut alors isoler l'unique solution qui satisfait le tier central.

L'utilisation de l'arborescence dans la classe $\mathcal{A}(G,\bowtie)$ permet donc d'obtenir un algorithme de complexité logarithmique 2. On vérifie qu'aucun algorithme linéaire de complexité inférieure à 3 n'est possible.

4.3.2 Résolution dans $\mathcal{A}(G,\bowtie)$

Un théorème analogue au théorème 13 est le suivant :

Proposition 14. *Pour tout graphe d'équation muet $G=(V,E,r)$, on a :*

$$pC[\mathcal{A}(G,\bowtie)] \geq \max_{c<Card(V)} \min_{c\leq c'\leq 2c} \min_{\substack{A\subseteq V \\ Card(A)=c'}} G(A).$$

Preuve. Pour une formule F dans $\mathcal{A}(G,\bowtie)$ et pour tout c, l'ensemble des sommets de F qui correspondent à la jointure d'au moins $c+1$ relations admet un élément minimum g. Si g est une opération $\bowtie$, elle correspond à la jointure de deux parties de V de cardinal inférieur à c, donc à un ensemble de relations de cardinal compris entre $c+1$ et $2c$. □

Cette proposition et le théorème 5 entraînent :

Théorème 15. *Pour un graphe d'équation muet $G=(V,E,r)$ connexe localement réversible, si λ_2 est la seconde valeur propre de la forme quadratique G, on a :*

$$pC[\mathcal{A}(G,\bowtie)] \geq \frac{2\lambda_2}{9} Card(V).$$

On constate un rapport de $\frac{8}{9}$ avec le théorème analogue 9 dans la classe $\mathcal{L}(G,\bowtie)$.

Preuve. Le maximum de $\min_{c\leq c'\leq 2c} c'\left(1-\frac{c'}{n}\right)$ lorsque c varie de 0 à n est $\frac{2n}{9}$. □

4.3.3 Résolution dans $\mathcal{A}(G,\bowtie,\gamma)$

Si l'on cherche une pince aux fonctions f et g, c'est-à-dire si l'on cherche à résoudre l'équation $f(x)=g(y)$, on commence par restreindre l'espace des solutions par des opérations γ pour n'avoir qu'une solution. Ensuite, on résout séparément $f(x)=z$ et $g(y)=z$, et l'on joint les deux ensembles de solution par une opération $\bowtie$. C'est l'utilisation du paradoxe des anniversaires.

Le principal avantage d'une résolution arborescente est la possibilité de réduire la complexité de la recherche d'une solution par l'emploi du paradoxe des anniversaires. Du point de vue de l'implémentation réelle, on constate la difficulté d'obtenir une complexité en espace réduite car l'emploi du paradoxe des anniversaire nécessite l'utilisation de tables de hachage volumineuses. L'implémentation de résolutions atteignant la complexité théorique n'est donc pas simple.

La méthode de réduction des algorithmes contrôlés ne permet ici d'obtenir qu'une inégalité :

Théorème 16. *Pour tout graphe d'équation muet $G = (V, E, r)$, quelle que soit l'interprétation de distribution localement uniforme considérée, on a :*

$$pC[\mathcal{A}^{\rho}(G, \bowtie, \gamma)] - \rho \geq pC[\mathcal{A}(G, \bowtie)] - G(V)$$

pour tout ρ.

Preuve. Soit A une formule optimale de $\mathcal{A}^{\rho}(G, \bowtie, \gamma)$. Le lemme 10 montre qu'il existe une formule A' de $\mathcal{A}(G', \bowtie)$ telle que $pC(A) \geq pC(A')$, et où $G' = (V, E, r')$ pour un rang contrôlé r'. On utilise la formule A' sur le graphe de départ G. Pour éviter toute confusion, on notera en indice de la formule le graphe de référence. En notant $\rho = G'(V)$, on cherche donc à montrer $pC(A'_{G'}) - G'(V) \geq pC(A'_G) - G(V)$. Pour toute sous-formule B de A', on a :

$$\begin{aligned} rg(B_{G'}) - G'(V) &= \sum_{\substack{\{u,v\} \in E \\ u \notin B \text{ ou } v \notin B}} r(\{u, v\}) - \sum_{v \notin B} r'(v) \\ rg(B_G) - G(V) &= \sum_{\substack{\{u,v\} \in E \\ u \notin B \text{ ou } v \notin B}} r(\{u, v\}) - \sum_{v \notin B} r(v) \end{aligned}$$

et $r'(v) \leq r(v)$ pour tout v, d'où $pC(A'_{G'}) - \rho \geq pC(A'_G) - G(V)$. □

Les théorèmes 16 et 15 entraînent :

Théorème 17. *Pour tout graphe d'équation muet $G = (V, E, r)$ connexe et localement réversible, si λ_2 est la seconde valeur propre de la forme quadratique G, on a :*

$$pC[\mathcal{A}^{\rho}(G, \bowtie, \gamma)] \geq \frac{2\lambda_2}{9} Card(V) - G(V) + \rho.$$

muet.

Conclusion

Etant donné la forme quadratique G définie par un graphe d'équation muet (V, E, r), on peut définir le *profil* de G comme étant une fonction pG de $\mathbb{N}$ dans $\mathbb{N}$ définie par

$$pG(c) = \min_{Card(A)=c} G(A).$$

C'est donc le minimum d'une forme quadratique sur des vecteurs à coordonnées 0 ou 1 de poids donné.

Dans le cas d'un graphe localement réversible, on peut minorer le profil en utilisant la seconde valeur propre de G par

$$pG(c) \geq \lambda_2 c \left(1 - \frac{c}{Card(V)}\right).$$

Cela donne une borne inférieure précise facilement calculable.

La complexité du graphe vis-à-vis des classes $\mathcal{L}^0(G, \bowtie, \gamma)$ et $\mathcal{A}^0(G, \bowtie, \gamma)$ ne dépend pas de la base, et est liée au profil par

$$\begin{aligned} pC[\mathcal{L}^0(G, \bowtie, \gamma)] &\geq \max_c pG(c) - G(V) \\ pC[\mathcal{A}^0(G, \bowtie, \gamma)] &\geq \max_c \min_{c \leq c' \leq 2c} pG(c') - G(V). \end{aligned}$$

Dans le cas d'un graphe localement réversible, on obtient

$$\begin{aligned} pC[\mathcal{L}^0(G, \bowtie, \gamma)] &\geq \frac{\lambda_2}{4} Card(V) - G(V) \\ pC[\mathcal{A}^0(G, \bowtie, \gamma)] &\geq \frac{2\lambda_2}{9} Card(V) - G(V). \end{aligned}$$

La classe $\mathcal{L}^0(G, \bowtie, \gamma)$ généralise la notion de recherche exhaustive pour obtenir une seule solution. La classe $\mathcal{A}^0(G, \bowtie, \gamma)$ généralise la notion d'attaque de type anniversaire.

Chapitre 5

Contraction de graphe d'équation défini

Lorsque les boîtes d'un circuit ne sont pas spécifiées, la résolution de graphes d'équation (détaillée dans le chapitre 3) permet d'étudier sa complexité. Dans un graphe d'équation défini, certaines relations peuvent présenter des propriétés mutuelles qui ouvrent la voie à de nouvelles méthodes.

Dans ce chapitre, on met en valeur la notion de *contraction* de graphe. L'idée de la contraction consiste à fusionner plusieurs sommets du graphe, ce qui revient à expliciter le comportement mutuel des relations correspondantes. Formellement, on définit ainsi la contraction :

Définition 14. Soient un graphe d'équation muet $G = (V, E, r)$ et une relation d'équivalence $\sim$ sur V. On définit le *graphe contracté* $G/\sim = (V/\sim, E', r')$ par

$$E' = \{\{\bar{u}, \bar{v}\}; \{u, v\} \in E\}$$

où $\bar{u}$ désigne la classe d'équivalence du sommet u, et

$$\begin{aligned} r'(\bar{u}) &= G(\bar{u}) \\ r'(\{\bar{u}, \bar{v}\}) &= \sum_{\substack{u' \sim u \quad v' \sim v \\ \{u', v'\} \in E}} r(\{u', v'\}). \end{aligned}$$

Pour que cette contraction se justifie du point de vue de la résolution des graphes d'équations, il faut être en mesure d'assurer que la complexité de la détermination d'une relation contractée est négligeable. Ainsi, cette notion est bien adaptée, par exemple, pour exprimer que certaines relations adjacentes sont linéaires.

Cette méthode d'analyse est illustrée dans ce chapitre par un exemple : la cryptanalyse de la fonction de hachage FFT Hash II proposée par Claus Schnorr [25]. Dans [42], l'auteur de ce mémoire montre que l'on peut construire des collisions pour FFT Hash II. On étudie dans ce chapitre cette attaque en mettant en évidence la notion de contraction de graphe d'équation.

5.1 Description de FFT Hash II

5.1.1 Schéma général

La fonction FFT Hash II est une fonction de hachage itérative définie par une fonction de compression f qui à h et m retourne $f(h,m)$ et une valeur initiale

$$h_0 = 0123\ 4567\ 89ab\ cdef\ fedc\ ba98\ 7654\ 3210.$$

Les blocs h et m sont de 128 bits.

La fonction de compression est définie par deux bijections Rec et $FT2$ sur l'ensemble $(\mathbb{Z}/p\mathbb{Z})^{16}$ où $p = 2^{16}+1$. Si h et m sont des chaînes de 128 bits, la concaténation hm représente 16 nombres de 16 bits, qui peuvent être vus comme les éléments de $\mathbb{Z}/p\mathbb{Z}$ de 0 à $p-2$. Ainsi, hm peut être vu comme un élément de $(\mathbb{Z}/p\mathbb{Z})^{16}$.

Inversement, un élément de $\mathbb{Z}/p\mathbb{Z}$ considéré comme un entier compris entre 0 et $p-1$ peut être pris modulo 2^{16} et représenter une chaîne de 16 bits. On définit ainsi $f(h,m)$ comme étant $(Rec \circ FT2 \circ Rec)(hm)[8,15]$ pris modulo 2^{16} où l'on définit $x[i,j]$ comme étant la sous-suite de x du i-ième élément jusqu'au j-ième. L'irréversibilité est obtenue par l'oubli de $(Rec \circ FT2 \circ Rec)(hm)[0,7]$.

On définit :

$$\begin{aligned} A(m) &= h_0 m \\ B(m) &= Rec(A(m)) \\ C(m) &= FT2(B(m)) \\ D(m) &= Rec(C(m)) \end{aligned}$$

Donc, $f(h_0,m)$ est la chaîne des 8 derniers nombres de $D(m)$ pris modulo 2^{16}.

5.1.2 Fonctions internes

Fonction Rec

Si les x_i sont dans $\mathbb{Z}/p\mathbb{Z}$ pour $i = 0,\ldots,15$, on définit $y_{-3} = x_{13}$, $y_{-2} = x_{14}$, et $y_{-1} = x_{15}$, puis :

$$y_i = x_i + y_{i-1}^* y_{i-2}^* + y_{i-3} + 2^i \tag{5.1}$$

où l'opération $*$ est définie ainsi : si $y = 0$, on pose $y^* = 1$, et sinon, $y^* = y$. On pose alors :

$$Rec(x_0,\ldots,x_{15}) = y_0,\ldots,y_{15}$$

Ceci définit entièrement Rec.

Fonction $FT2$

Si les x_i sont dans $\mathbb{Z}/p\mathbb{Z}$ pour $i = 0,\ldots,7$, on définit :

$$y_j = \sum_{i=0}^{7} \omega^{ij} x_i$$

où $\omega = 2^4$. On pose ensuite $FT(x_0, \ldots, x_7) = y_0, \ldots, y_7$. Cela définit une bijection linéaire FT.

Si les x_i sont dans $\mathbb{Z}/p\mathbb{Z}$ pour $i = 0, \ldots, 15$, on définit :

$$\begin{aligned} y_0, y_2, \ldots, y_{14} &= FT(x_0, x_2, \ldots, x_{14}) \\ y_1, y_3, \ldots, y_{15} &= FT(x_1, x_3, \ldots, x_{15}) \end{aligned}$$

On définit ensuite $FT2(x_0, \ldots, x_{15}) = y_0, \ldots, y_{15}$. Cela définit entièrement $FT2$. On note que c'est une bijection linéaire.

5.1.3 Graphe de la fonction de compression

Le graphe de la fonction de compression de FFT Hash II peut donc se représenter comme le graphe G de la figure 5.1. Le rang des arêtes est 1 et la base est p. Les entrées de la constante h_0 correspondent à des sommets de rang 0. Les fonctions définies par l'équation (5.1) dans la fonction Rec correspondent à des boîtes de rang 4. Les deux fonctions linéaires FT se contractent en deux sommets de rang 8. Les sommets de rang 1 sont des sommets d'entrée/sortie ou des sommets de répétition. On a donc une *fonction* de $x \cup y$ dans $z \cup t$.

5.2 Principe de l'attaque

5.2.1 Remarques de base

On constate tout d'abord que l'on peut considérer le graphe de la figure 5.1 comme une fonction de $x \cup y$ dans $z \cup r$. Ces deux fonctions admettent simultanément des collisions car il y a correspondance bijective entre $z \cup t$ et $z \cup r$. On considère par la suite le graphe d'équation G' obtenu en fixant les valeurs de r. Des solutions de G' donnant le même z fournissent donc des collisions pour la fonction de compression.

On verra que l'on peut effectuer le groupement illustré sur la figure 5.1. La forme quadratique sur ce groupement donne le rang 16. Ce groupement se contracte donc en un sommet de rang 16.

A ce niveau d'observations, le graphe G' se contracte en le graphe illustré sur la figure 5.2. La dernière observation consiste à remarquer que l'on obtient une fonction de y dans z : il existe un algorithme linéaire qui résout le graphe où y est fixé avec complexité logarithmique 0. Une collision pour cette fonction fournissant une collision pour la fonction de compression, il suffit d'appliquer l'algorithme du paradoxe des anniversaires (de complexité 2^{24} ici).

Tout ce qu'il y a à démontrer est la validité du groupement effectué.

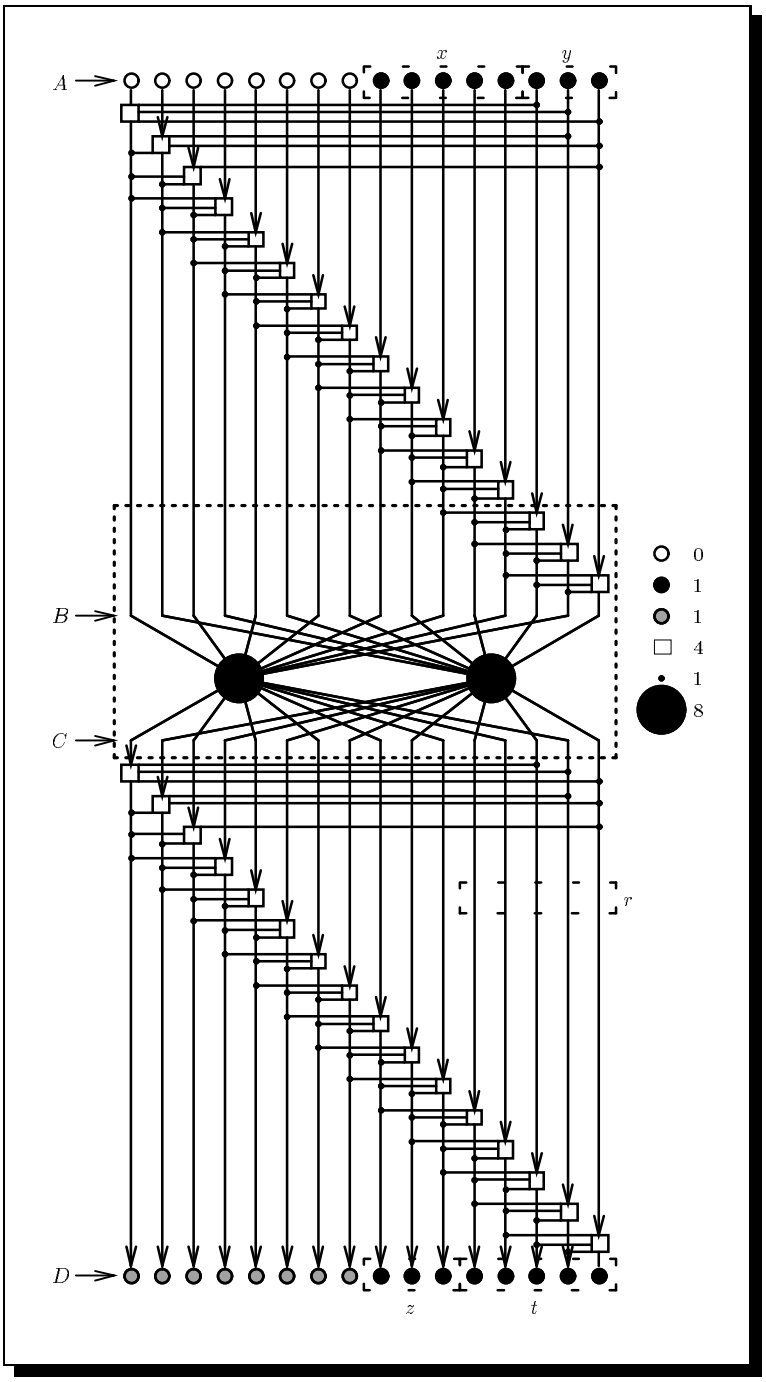

Figure 5.1 : Fonction de compression de FFT Hash II.

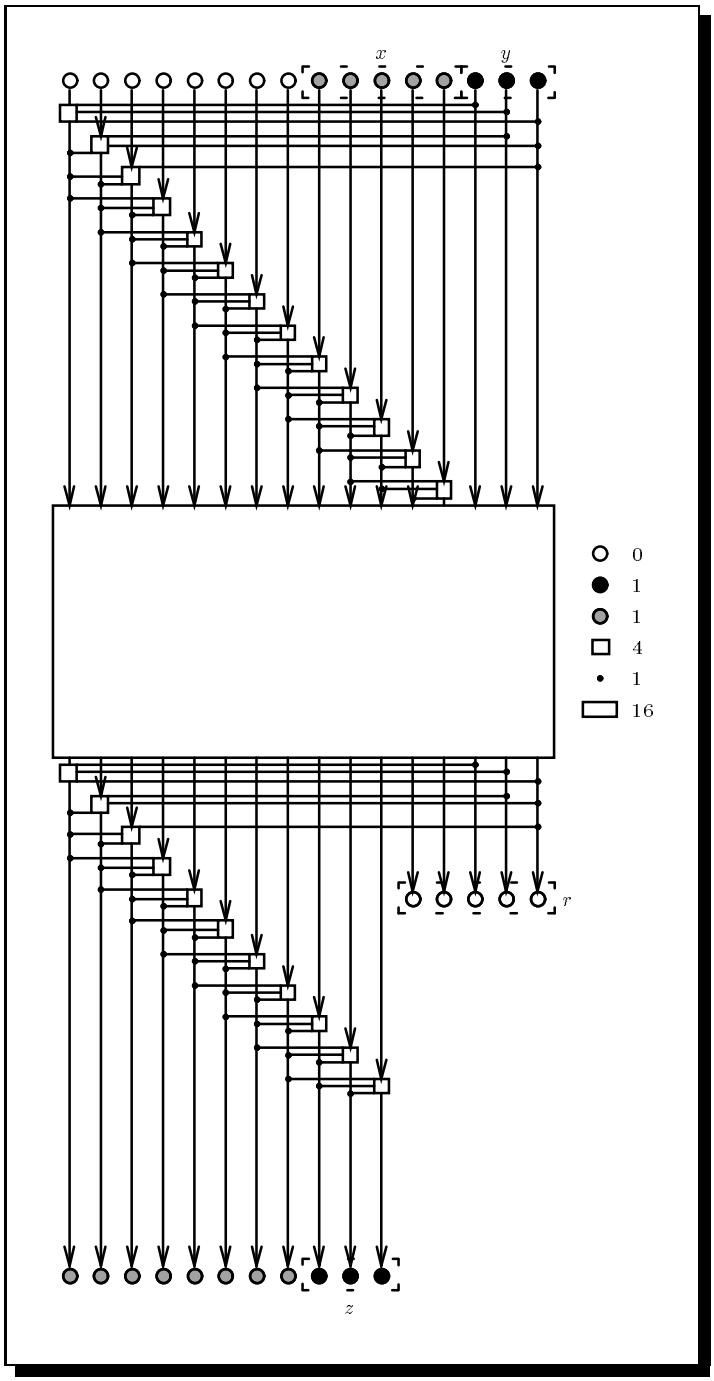

Figure 5.2 : Fonction de compression contractée.

5.2.2 Justification du groupement

On suppose connus $A(m)[0,7]$ (la constante h_0), $A(m)[13,15]$ (ensemble y de la figure 5.2) et $C(m)[11,15]$ (ensemble r de la figure). On cherche à résoudre le circuit.

On peut commencer le calcul normal en obtenant $B(m)[0,7]$. A ce point, on connaît 16 entrées du groupement. On note $A(m)[13,15] = abc$ et $B(m)[0,7] = g$.

Si l'on note $y = B(m)[8,15]$, l'équation $C(m)[11,15] = r$ équivaut à :

$$FT2(gy)[11,15] = r \tag{A}$$

Cette équation est linéaire en y. Dans le paragraphe 5.3.1, on montre que l'on a un vecteur v calculable à partir de abc, g et r et trois vecteurs u_p, u_i, et u'_p tels que l'on a :

$$(A) \Longleftrightarrow \exists \lambda, \lambda', \mu \quad y = v + \lambda u_p + \lambda' u'_p + \mu u_i$$

Donc, le système du groupement :

$$\left\{ \begin{array}{rcl} m[5,7] & = & abc \\ B(m)[0,7] & = & g \\ C(m)[11,15] & = & r \end{array} \right.$$

entraîne :

$$\left\{ \begin{array}{rcl} y & = & v + \lambda u_p + \lambda' u'_p + \mu u_i \\ y_{13} & = & a + y^*_{12} y^*_{11} + y_{10} + 2^{13} \\ y_{14} & = & b + y^*_{13} y^*_{12} + y_{11} + 2^{14} \\ y_{15} & = & c + y^*_{14} y^*_{13} + y_{12} + 2^{15} \end{array} \right. . \tag{B}$$

En substituant la valeur de y donnée par la première équation dans les autres, on obtient 3 équations à trois inconnues λ, λ' et μ. Dans le paragraphe 5.3.2, on montre que ceci peut être réduit à une équation du second degré. On obtient ainsi une, 0 ou 2 solutions y en négligeant les cas dégénérés, ce qui fait en moyenne une unique solution.

La complexité de la résolution obtenue justifiera le groupement dans le graphe de la figure 5.2.

5.3 Détail de l'attaque

5.3.1 Résolution de (A)

La fonction $x \longmapsto FT2(x)[11,15]$ est linéaire de rang 5, donc a un noyau de dimension 3. On pose :

$$\begin{array}{rcl} u & = & (\ 0\ \ 0\ \ 0\ \ 0\ \ 4081\ \ 256\ \ 1\ \ 61681\ ) \\ u' & = & (\ 0\ \ 0\ \ 0\ \ 0\ \ 65521\ \ 4352\ \ 1\ \ 0\ ) \end{array}$$

On remarque que :

$$\begin{array}{rcl} FT(u) & = & (\ 482\ \ 56863\ \ 8160\ \ 57887\ \ 7682\ \ 0\ \ 0\ \ 0\ ) \\ FT(u') & = & (\ 4337\ \ 61202\ \ 65503\ \ 544\ \ 61170\ \ 3855\ \ 0\ \ 0\ ) \end{array}$$

On définit l'opération $\times$ qui entrelace les vecteurs par :

$$(x_0, \ldots, x_7) \times (y_0, \ldots, y_7) = (x_0, y_0, \ldots, x_7, y_7)$$

On a $FT2(x \times y) = FT(x) \times FT(y)$. Donc, on peut définir :

$$\begin{array}{rcl} u_p & = & u \times 0 \\ u_i & = & 0 \times u \\ u'_p & = & u' \times 0 \end{array}$$

soit :

$$\begin{array}{rl} u_p = (0, & 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\ \ 4081,\ \ 0,\ \ 256,\ \ 0,\ 1,\ 0,\ 61681,\ \ 0) \\ u_i = (0, & 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\ \ 0,\ \ 4081,\ \ 0,\ \ 256,\ 0,\ 1,\ \ 0,\ 61681) \\ u'_p = (0, & 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\ 65521,\ \ 0,\ \ 4352,\ \ 0,\ 1,\ 0,\ \ 0,\ \ 0) \end{array}$$

de manière à avoir une base du noyau de $x \longmapsto FT2(x)[11, 15]$.

Si M représente la matrice de FT, on la décompose en blocs de 4 lignes et 4 colonnes :

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}$$

Si x et y sont deux vecteurs de $(\mathbb{Z}/p\mathbb{Z})^4$, on a :

$$FT(xy)[4, 7] = 0 \Longleftrightarrow y = -M_{22}^{-1} M_{21} x$$

Soit donc :

$$N = -M_{22}^{-1} M_{21} = \begin{pmatrix} 65281 & 4335 & 289 & 61170 \\ 3823 & 8992 & 53012 & 65248 \\ 8447 & 61748 & 56545 & 4335 \\ 4369 & 57090 & 3823 & 256 \end{pmatrix}$$

Si x et y sont maintenant des vecteurs de $(\mathbb{Z}/p\mathbb{Z})^8$, on a :

$$FT2(xy)[8, 15] = 0 \Longleftrightarrow y = Nx_p \times Nx_i$$

où l'on note $x = x_p \times x_i$. On pose $g = x_p \times x_i$. On choisit un vecteur r_0 de 8 scalaires tel que $r = FT2(0r_0)[11, 15]$. On définit alors le vecteur v par :

$$v = g(Nx_p \times Nx_i + r)$$

Ainsi, v est un vecteur qui commence par g, et qui est tel que $FT2(v)$ finisse par le vecteur r.

On a donc :

$$(A) \Longleftrightarrow \exists \lambda, \lambda', \mu \quad y = v + \lambda u_p + \lambda' u'_p + \mu u_i$$

5.3.2 Résolution de (B)

Si aucun y_i ($i = 11, 12, 13, 14$) n'est égal à 0 (ce qui se produit le plus souvent, on pourra le tester à la fin de la résolution pour rejeter les y qui ne vérifient pas cette condition), les y_i^* valent y_i. Donc, le système :

$$\left\{\begin{array}{rcl} y & = & v + \lambda u_p + \lambda' u'_p + \mu u_i \\ y_{13} & = & a + y_{12}^* y_{11}^* + y_{10} + 2^{13} \\ y_{14} & = & b + y_{13}^* y_{12}^* + y_{11} + 2^{14} \\ y_{15} & = & c + y_{14}^* y_{13}^* + y_{12} + 2^{15} \end{array}\right.$$

est équivalent à :

$$\begin{array}{rcl} v_{13} + \mu & = & a + (v_{12} + \lambda + \lambda')(v_{11} + 256\mu) + v_{10} + 256\lambda + 4352\lambda' + 2^{13} \\ v_{14} + 61681\lambda & = & b + (v_{13} + \mu)(v_{12} + \lambda + \lambda') + (v_{11} + 256\mu) + 2^{14} \\ v_{15} + 61681\mu & = & c + (v_{14} + 61681\lambda)(v_{13} + \mu) + (v_{12} + \lambda + \lambda') + 2^{15} \end{array} .$$

On définit :

$$\begin{array}{rcl} a' & = & a + v_{12}v_{11} + v_{10} + 2^{13} - v_{13} \\ b' & = & b + v_{13}v_{12} + v_{11} + 2^{14} - v_{14} \\ c' & = & c + v_{14}v_{13} + v_{12} + 2^{15} - v_{15} \end{array}$$

Le système (B) est équivalent à :

$$\begin{pmatrix} v_{11} + 256\mu + 256 & v_{11} + 256\mu + 4352 & a' - (1 - 256v_{12})\mu \\ v_{13} + \mu - 61681 & v_{13} + \mu & b' + (256 + v_{12})\mu \\ 61681\,(v_{13} + \mu) + 1 & 1 & c' - (61681 - v_{14})\mu \end{pmatrix} \begin{pmatrix} \lambda \\ \lambda' \\ 1 \end{pmatrix} = 0$$

Ceci est un système linéaire aux inconnues λ et λ'. Il ne peut admettre de solutions que lorsque le déterminant est nul. Cette condition est suffisante sauf pour un nombre négligeable de cas.

Le déterminant est un polynôme en μ de degré *a priori* 3. Cependant, le coefficient de μ^3 est le déterminant de la matrice suivante :

$$\begin{pmatrix} 256 & 256 & (1 - 256v_{12}) \\ 1 & 1 & -(256 + v_{12}) \\ 61681 & 0 & (61681 - v_{14}) \end{pmatrix}$$

qui est nul, car la première ligne est multiple de la seconde.

Le coefficient de μ^2 est non nul sauf dans un cas sur p. Donc, sauf dans un nombre négligeable de cas, μ doit être solution d'une équation du second degré. On remarque que le discriminant de cette équation est un carré environ une fois sur deux. On a donc 0 ou 2 valeurs de μ solutions en général.

Pour chaque μ, on a une unique solution (λ, λ') au système linéaire (sauf un nombre négligeable de cas). Pour chaque (λ, λ', μ), on peut calculer y solution du système (B), puis le message m. On a donc 0 ou 2 solutions m, avec une moyenne de 1.

5.3.3 Réduction de la fonction FFT Hash II

En utilisant les résultats des paragraphes 5.3.1 et 5.3.2, on a une *pseudo-fonction* f_r qui à abc associe un ensemble de $D(m)[8,10]$ correspondant à des m tels que $m[5,7] = abc$ et $C(m)[11,15] = r$. Le temps de calcul de cette fonction est comparable à celui de FFT Hash II. De plus, la moyenne du cardinal de l'ensemble $f_r(abc)$ est 1, d'où le qualificatif de "pseudo-fonction".

La fonction f_r est une réduction de FFT Hash II dans le sens qu'une collision pour f_r donne une collision pour FFT Hash II. En utilisant le paradoxe des anniversaires pour f_r, on peut donc trouver une collision en un temps de l'ordre de 2^{24} opérations.

On peut également utiliser f_r pour inverser FFT Hash II. Si l'on cherche m tel que $D(m)[8,15] = h$, on peut calculer $r = Rec^{-1}(h)[11,15]$ et chercher abc tel que $h[0,2] \in f_r(abc)$. Ceci fournit une solution m. Le temps de calcul est de l'ordre de 2^{48} opérations.

Avec cette attaque, on a trouvé deux collisions en 24 heures de travail sur une station de travail SUN4. Avec le choix :

$$r = 5726\ 17fc\ b115\ c5c0\ a631$$

on a trouvé :

$$\begin{aligned} f(h_0, 17b3\ 2755\ 4e52\ b915\ 2218\ 1948\ 00a8\ 0002) &= \\ f(h_0, 9c70\ 504e\ 834c\ b15c\ f404\ 94e2\ 02a7\ 0002) &= \\ 0851\ 393d\ 37c9\ 66e3\ d809\ d806\ 5e8c\ 05b8 & \end{aligned}$$

et :

$$\begin{aligned} f(h_0, 8ccc\ 23a4\ 086d\ fbb9\ 85f4\ 70b2\ 029e\ 0002) &= \\ f(h_0, 9d53\ 45ae\ 3286\ ada7\ 8c77\ 9877\ 02b4\ 0002) &= \\ 10e5\ 49f5\ 9df0\ d91b\ 0450\ afcc\ fba4\ 2063 & \end{aligned}$$

Conclusion

On a mis en évidence des faiblesses de FFT Hash II qui ont permis de trouver des collisions : d'une part, le début du calcul de $f(h,m)$ dépend de trop peu d'informations sur m, puisque $B(m)[0,7]$ est une fonction des trois derniers coefficients de m seulement. D'autre part, le résultat de $f(h,m)$ permet d'obtenir trop d'informations dans les calculs intermédiaires, puisque $D(m)[8,15]$ permet

de calculer $C(m)[11, 15]$. Comme le lien entre $B(m)$ et $C(m)$ est linéaire, on a une attaque possible.

On note que ces défauts étaient déjà présents dans FFT Hashing, mais conduisaient à une attaque moins facile. La faiblesse fondamentale de FFT Hash II est donc de reposer sur un mauvais graphe de calcul.

L'attaque présentée illustre la technique de contraction de graphe dans l'analyse des graphes interprétés. Elle montre que de bons graphes doivent posséder des boîtes de calcul sans structure commune, interdisant ainsi la contraction.

Chapitre 6

Etude d'une fonction de hachage

Dans ce chapitre, on étudie la sécurité d'une nouvelle famille de fonctions de compression $g_{k,s}$ dépendant de deux paramètres k et s décrites uniquement par leur graphe de calcul. Ceci a été proposé dans [26, 73], résultat d'un travail en commun avec Claus Schnorr qui faisait suite aux propositions des fonctions FFT Hashing (voir le chapitre 5) [24, 25]. Ces nouvelles fonctions reprennent des idées des précédentes, comme le graphe de la transformée de Fourier rapide et l'irréversibilité par l'oubli, et présentent la nouvelle notion de *multipermutation* qui sera étudiée au chapitre 8 et dont la nécessité sera justifiée dans le chapitre 7.

Une fonction de compression $g_{k,s}$ admet deux entrées h et m dont une (h) est un résultat déjà compressé (ou une valeur initiale) et une sortie h'. Elle est supposée satisfaire les critères suivants :

- (problème de l'inversion) étant donnés h et h', il est difficile de trouver m tel que $g(h, m) = h'$;
- (*problème de collisions ciblé*) étant donnés h et h', il est difficile de trouver m et m' tels que $g(h, m) = g(h', m')$.

6.1 Description de $g_{k,s}$

On décrit la fonction de compression $g_{k,s}$ associée aux paramètres k et s. Elle se décrit par un graphe de calcul $G_{k,s}$. Toutes les arêtes ont le même domaine D et tous les sommets ont pour rang 2. Ils représentent une classe de boîtes particulière : les *multipermutations* (voir chapitre 8). La fonction de compression admet 2^k entrées et 2^{k-1} sorties dans D.

De manière informelle, $G_{k,s}$ est le graphe de la transformée de Fourier rapide de 2^{k-1} sommets généralisée à $s+1$ couches au lieu de se limiter à k couches. Les sommets de la première couche admettent deux entrées, et les sommets de la dernière couche admettent une seule sortie (l'"autre" est oubliée).

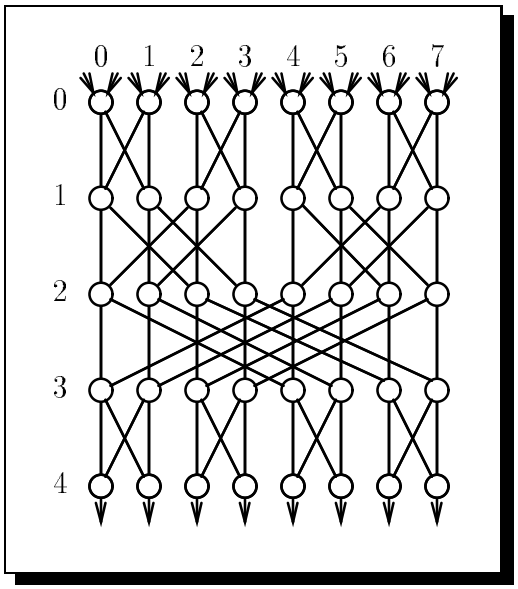

Figure 6.1 : La fonction de compression $g_{4,4}$.

Les sommets du graphe sont regroupés en couches de la couche $i = 0$ jusqu'à la couche $i = s$. Chaque couche i contient 2^{k-1} sommets $v_{i,0}, \ldots, v_{i,2^{k-1}-1}$. Chaque sommet $v_{i,j}$ est lié à deux sommets de la couche $i-1$ (sauf pour $i = 0$) et deux sommets de la couche $i+1$ (sauf pour $i = s$) qui sont $v_{i+1,j}$ et $v_{i+1,j\oplus 2^{i \bmod k-1}}$, où $\oplus$ représente le *ou*-exclusif bit-à-bit.

Les sommets de la couche $i = 0$ ont deux entrées et les sommets de la couche $i = s$ ont une sortie. Les valeurs h et m sont scindées en 2^{k-1} valeurs mises en vis-à-vis pour entrer dans le graphe : chaque entrée prend une valeur de h et une valeur de m.

La figure 6.1 représente la fonction $g_{4,4}$. Le problème d'inversion de $g_{4,4}$ correspond exactement au graphe d'équation de la figure 4.2.

La famille de fonction de compression proposée pour un domaine D est l'ensemble de toutes les fonctions $g_{4,4}$ pour chaque jeu de *multipermutations* possible.

La complexité logarithmique du problème de l'inversion par recherche exhaustive est 2^{k-1}. Celle du problème des collisions ciblé par le paradoxe des anniversaires est 2^{k-2}.

Initialement, cette famille a été proposée pour un domaine de 2^{16} éléments. Pour avoir des paramètres comparables à ceux des autres fonctions, soit une complexité de l'ordre de 2^{64}, le choix $k = 4$ s'impose. Le problème est la recherche de s pour avoir une sécurité suffisante.

La résolution illustrée sur la figure 4.3, de complexité 5, montre que l'on peut

trouver des collisions avec une complexité théorique de $(2^{16})^5 = 2^{80}$. Une attaque de type anniversaire a une complexité sensiblement égale à la racine carrée de ce nombre[1], le choix $s = 4$ est donc insuffisant.

On cherche, dans ce chapitre, à estimer la complexité du problème d'inversion de $g_{k,s}$ vis-à-vis des classes d'algorithmes $\mathcal{L}^0(\bowtie, \gamma)$ et $\mathcal{A}^0(\bowtie, \gamma)$ définies au chapitre 3. On utilise la méthode de l'étude spectrale de la forme quadratique du réseau présentée au chapitre 4 et une méthode adaptée aux graphes symétriques. On obtiendra des bornes inférieures pour la complexité. Une méthode directe, adaptée aux graphes utilisés, fournira des bornes supérieures.

6.2 Recherche de points fixes

La fonction de compression $g_{k,s}$ repose sur une permutation $f_{k,s}$: si l'on note $f_{k,s}(h, m) = (h', h'')$, la partie h'' est "oubliée". $g_{k,s}$ est donc la composition d'une permutation supposée réaliser de bonnes diffusions et confusions avec une fonction de troncature.

Dans cette partie, on étudie le problème de recherche de points fixes pour la permutation $f_{k,s}$. La complexité logarithmique, par une recherche exhaustive, est 2^k. Ce problème présente l'avantage de posséder un graphe symétrique. Il n'est pas lié directement à la sécurité de la fonction de compression, mais une manipulation permettra d'exploiter les résultats de cette étude.

6.2.1 Etude spectrale

Dans la suite, on utilise une numérotation des sommets plus agréable dans les calculs (mais moins dans les représentations graphiques). Pour éviter les confusions, on notera les sommets $w_{i,j}$ avec la nouvelle numérotation. On suppose que le sommet $w_{i,j}$ est lié aux sommets $w_{i+1,2j \bmod 2^{k-1}}$ et $w_{i+1,2j+1 \bmod 2^{k-1}}$. Il est facile de voir le lien entre les deux numérotations :

$$v_{i,j} = w_{i,R^i(rev(j))}$$

où R est la rotation circulaire d'un bit vers la gauche dans une numérotation binaire de j ($R(b_{k-2}.2^{k-2} + \ldots + b_0) = b_{k-3}.2^{k-2} + \ldots + b_{k-2}$) et $rev(j)$ est la chaîne renversée des bits de j ($rev(b_{k-2}.2^{k-2} + \ldots + b_0) = b_0.2^{k-2} + \ldots + b_{k-2}$). Ceci provient du fait que $\{2j \bmod 2^{k-1}, 2j + 1 \bmod 2^{k-1}\} = \{R(j), R(j) \oplus 1\}$ et que $2^{i \bmod k-1} = R^i(1)$.

[1] en fait, la même attaque adaptée au problème de recherche de collisions ciblé a une complexité logarithmique de 3, soit 2^{48}.

On note

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \\ 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \end{pmatrix}$$

la matrice de type $2^{k-1} \times 2^{k-1}$ qui représente la matrice de transition d'une couche à l'autre dans la numérotation des $w_{i,j}$. On note

$$H_{k,s} = \begin{pmatrix} 0 & A & 0 & \cdots & 0 \\ {}^tA & 0 & A & \cdots & 0 \\ 0 & {}^tA & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

la matrice de type $(s+1).2^{k-1} \times (s+1).2^{k-1}$ d'adjacence du graphe de $g_{k,s}$[2].

Dans la suite, on étudie une matrice légèrement différente :

$$H'_{k,s} = \begin{pmatrix} 0 & A & 0 & \cdots & {}^tA \\ {}^tA & 0 & A & \cdots & 0 \\ 0 & {}^tA & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A & 0 & 0 & \cdots & 0 \end{pmatrix}$$

de type $(s+1).2^{k-1} \times (s+1).2^{k-1}$. Si I représente la matrice identité, $2I - \frac{1}{2}H'_{k,s}$ représente le graphe d'équation $G'_{k,s}$ du problème de recherche de point fixe.

Dans la suite, on note f_α le vecteur de taille 2^{k-1} dont la $(i+1)$-ième coordonnée est $(-1)^{\alpha \cdot i}$, où $\alpha \cdot i$ représente le produit scalaire de α et i vus comme des vecteurs de bits correspondant à leur numération binaire. $\alpha \cdot i$ est aussi le poids de Hamming du *et* bit-à-bit de α et i.

Lemme 18. *Pour tout m, M, n et j entiers, on pose $q = e^{\frac{2i\pi n}{s+1}}$ et on note*

$$V_{n,m,M,j} = \sqrt{\frac{2}{(s+1).2^{k-1}.(m+1)}} \begin{pmatrix} U \\ q.U \\ \vdots \\ q^s.U \end{pmatrix}$$

[2]la notation ${}^tA$ représente la matrice transposée de A.

où

$$U = \sum_{\ell=1}^{m} q^{-\ell} \sin \frac{j\pi\ell}{m+1} . f_{M.2^{\ell-1}}.$$

Une base orthonormale de vecteurs propres de $H'_{k,s}$ est constituée de deux types de vecteurs :

1. *pour chaque $n = 0, \ldots, s$, le vecteur $V_{n,1,0,1}$ correspondant à la valeur propre $4\cos\frac{2\pi n}{s+1}$ (dans le vecteur $V_{n,1,0,1}$, U est le vecteur dont toutes les coordonnées sont identiques);*

2. *pour chaque $n = 0, \ldots, s$, chaque $m = 1, \ldots, k-1$, chaque $j = 1, \ldots, m$ et chaque entier impair M compris entre 2^{k-m-1} et 2^{k-m}, le vecteur $V_{n,m,M,j}$ correspondant à la valeur propre $4\cos\frac{j\pi}{m+1}$, avec pour multiplicité l'entier $(s+1)\lceil 2^{k-m-2}\rceil$.*

Preuve. Pour chaque $n = 0, \ldots, s$, posons $q = e^{\frac{2i\pi n}{s+1}}$. On remarque que pour tout vecteur X de taille 2^{k-1}, si l'on note

$$X^q = \begin{pmatrix} X \\ q.X \\ \vdots \\ q^s.X \end{pmatrix}$$

on a

$$(qA + \bar{q}^t A).X = \lambda X \Longleftrightarrow H'_{k,s}.X^q = \lambda X^q.$$

Ceci réduit donc l'étude de $H_{k,s}$ à celle des matrices $qA + \bar{q}^t A$.

On remarque les égalités suivantes :

$$A.f_\alpha = \begin{cases} 0 & \text{si } \alpha \text{ impair} \\ 2f_{\frac{\alpha}{2}} & \text{si } \alpha \text{ pair} \end{cases}$$

$$^tA.f_\alpha = \begin{cases} 0 & \text{si } \alpha \geq 2^{k-2} \\ 2f_{2\alpha} & \text{si } \alpha < 2^{k-2}. \end{cases}$$

Donc, il est utile d'ordonner la base des f_α suivant le "motif" de α : tout entier α non nul s'écrit $2^i M$ où M est un entier impair que l'on appelle motif de α. Si M est un motif de $k-m-2$ bits, sur les vecteurs $f_M, \ldots, f_{2^{m-1}M}$, l'effet de $qA + \bar{q}^t A$ est celui de la matrice

$$\begin{pmatrix} 0 & 2q & 0 & \cdots & 0 & 0 \\ 2\bar{q} & 0 & 2q & \cdots & 0 & 0 \\ 0 & 2\bar{q} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 2q \\ 0 & 0 & 0 & \cdots & 2\bar{q} & 0 \end{pmatrix}.$$

Il est bien connu que les vecteurs propres de cette matrice sont, pour $j = 1, \ldots, m$, les vecteurs dont le ℓ-ième élément est $\bar{q}^\ell \sin \frac{j\pi\ell}{m+1}$. On obtient ainsi les vecteurs de deuxième type.

Pour $\alpha = 0$, f_0 est déjà vecteur propre. On obtient les vecteurs de premier type. □

Ceci permet d'énoncer le lemme suivant :

Lemme 19. *On a*

$$\lambda_2(G'_{k,s}) = \begin{cases} 4\sin^2 \frac{\pi}{2k} & \text{si } s+1 < 2k \\ 4\sin^2 \frac{\pi}{s+1} & \text{si } s+1 \geq 2k. \end{cases}$$

On en déduit, grâce aux théorèmes 9 et 15 les bornes inférieures suivantes pour $k = 4$:

s	3	4	5	6	7
$pC[\mathcal{L}^0(G'_{4,s}, \bowtie, \gamma)]$	5	6	8	9	10
$pC[\mathcal{A}^0(G'_{4,s}, \bowtie, \gamma)]$	5	6	7	8	9

Théorème 20. *Pour $s = 2k-3$, on a*

$$pC[\mathcal{L}^0(G'_{k,2k-3}, \bowtie, \gamma)] \geq \left((k-1)\sin^2 \frac{\pi}{2k}\right) 2^k$$

et

$$pC[\mathcal{A}^0(G'_{k,2k-3}, \bowtie, \gamma)] \geq \left(\frac{8(k-1)}{9}\sin^2 \frac{\pi}{2k}\right) 2^k.$$

L'étude spectrale n'est donc pas assez fine pour pouvoir conclure une complexité égale à 2^k. On a les bornes inférieures suivantes :

k	2	3	4	5	6
$pC[\mathcal{L}^0(G'_{k,2k-3}, \bowtie, \gamma)]$	2	4	8	13	22
$pC[\mathcal{A}^0(G'_{k,2k-3}, \bowtie, \gamma)]$	2	4	7	11	20

6.2.2 Utilisation des symétries

Le problème des graphes d'expansion est traité de manière spéciale dans le cas des graphes fortement symétriques. De tels graphes, comme les *graphes de Cayley*, sont en général issus de la théorie des groupes. Les problèmes algorithmiques posés sur les groupes débouchent rapidement sur des problèmes d'expansion. On utilise le théorème suivant (sous une nouvelle formulation), dû à László Babai et Mario Szegedy [77] :

Théorème 21. *Si G est un graphe d'équation localement réversible, transitif pour les arêtes, si d est la moyenne harmonique du degré de ses n sommets et si δ est la moyenne de la distance entre deux sommets, pour tout ensemble de sommets A de cardinal c, on a*

$$G(A) \geq \frac{d}{2\delta} c \left(1 - \frac{c}{n}\right).$$

On rappelle que la moyenne harmonique de a et b est $\frac{2}{1/a+1/b}$. La transitivité au sens des arêtes signifie que pour deux arêtes quelconques, il existe un automorphisme du graphe qui transforme l'une en l'autre.

Un corollaire analogue aux théorèmes 9 et 15 est :

Corollaire 22. *Avec les notations du théorème 21, on a*

$$pC[\mathcal{L}^0(G, \bowtie, \gamma)] \geq \frac{d}{8\delta}\mathrm{Card}(V) \quad \text{et} \quad pC[\mathcal{A}^0(G, \bowtie, \gamma)] \geq \frac{d}{9\delta}\mathrm{Card}(V).$$

Ce résultat s'applique directement au graphe $G'_{k,s}$:

Lemme 23. *Pour $s \geq k-1$, $G'_{k,s}$ est transitif au sens des arêtes.*

Preuve. On montre tout d'abord que $G'_{k,s}$ est transitif au sens des sommets : pour envoyer le sommet $w_{0,0}$ sur $w_{a,b}$, on utilise

$$\phi \ : \ w_{i,j} \mapsto w_{i+a \bmod s+1, j \oplus R^i(b)}.$$

On montre facilement que ϕ est un automorphisme.

Il ne reste plus qu'à montrer qu'il est possible d'envoyer l'arête $\{w_{0,0}, w_{1,0}\}$ sur l'arête $\{w_{0,0}, w_{1,1}\}$ par un automorphisme. Pour cela, on montre que

$$\phi(w_{i,j}) = \begin{cases} w_{i,j\oplus 2^{i-1}} & \text{si } 0 < i \leq k-1 \\ w_{i,j} & \text{sinon} \end{cases}$$

définit un tel automorphisme. □

On note que dans le cas $s = k-2$, le graphe n'est pas transitif au sens des arêtes en général. Dans le cas où k est pair par exemple, le nombre de couches du graphe est impair, et un cycle de longueur $s+1$ passe nécessairement par chaque couche. On remarque que $\{w_{0,0}, w_{1,0}\}$ est bien dans un tel cycle, mais pas $\{w_{0,0}, w_{1,1}\}$.

On peut majorer la distance moyenne du graphe par son diamètre, plus facile à calculer.

Lemme 24. *Dans le graphe $G'_{k,s}$, le diamètre est :*

$$\begin{array}{rl} 2k-2-\lceil\frac{s+1}{2}\rceil & \text{si } k-2\leq s\leq\frac{4k-7}{3} \\ s+1 & \text{si } \frac{4k-7}{3}\leq s\leq 2k-3 \\ 2k-2 & \text{si } 2k-3\leq s\leq 4k-5 \\ \lfloor\frac{s+1}{2}\rfloor & \text{si } 4k-5\leq s \end{array}$$

Preuve. Le graphe étant transitif au sens des sommets (dans tous les cas), il suffit d'étudier le plus court chemin de $w_{0,0}$ aux autres sommets.

On note que $w_{i,j}$ est lié à $w_{i+1,R(j)}$ et à $w_{i+1,R(j)\oplus 1}$. Donc, pour se rendre de $w_{0,0}$ à $w_{i,j}$, il faut modifier un certain nombre de bits pour obtenir la "forme" de j, puis se rendre sur la couche i. Ainsi, le sommet le plus éloigné de $w_{0,0}$ est de la forme $w_{i,2^{k-1}-1}$: Tout chemin allant de l'un à l'autre doit traverser $k-1$ couches consécutives et se rendre à la couche i.

Dans le cas $\frac{3(s+1)}{4}\leq k-1\leq s+1$, le sommet le plus éloigné est sur la couche $\lceil\frac{s+1}{2}\rceil$. La distance est $k-1+\left(k-1-\lceil\frac{s+1}{2}\rceil\right)$.

Dans le cas $\frac{s+1}{2}\leq k-1\leq\frac{3(s+1)}{4}$, le sommet le plus éloigné est sur la couche 0 et la distance est $s+1$.

Dans le cas $\frac{s+1}{4}\leq k-1\leq\frac{s+1}{2}$, le sommet le plus éloigné est sur la couche 0 et la distance est $2(k-1)$.

Dans le cas $k-1\leq\frac{s+1}{4}$, le sommet le plus éloigné est sur la couche $\lfloor\frac{s+1}{2}\rfloor$ et la distance est $\lfloor\frac{s+1}{2}\rfloor$. □

Dans le cas $s=2k-3$ par exemple, on en déduit $pC[\mathcal{L}^0(G'_{k,2k-3},\bowtie,\gamma)]\geq\frac{1}{4}2^k$ au lieu du 2^k espéré, mais on peut chercher une estimation plus fine de δ. Cela permet d'obtenir un coefficient de $\frac{1}{3}$ au lieu de $\frac{1}{4}$.

On note $C(j)$ la longueur de la plus grande suite de bits nuls de j, c'est-à-dire le maximum de ℓ tel qu'il existe des entiers a, b et c tels que $j=a+2^{b+\ell}c$ avec $a<2^b$ et $b+\ell\leq k-1$.

Lemme 25. *Si $v_{i,j}$ est un sommet de $G'_{k,2k-3}$, la distance de $v_{0,0}$ à $v_{i,j}$ est*

$$d(v_{0,0},v_{i,j})=\begin{cases} 2(k-1)-i-2C\left(\lfloor\frac{j}{2^i}\rfloor\right) & \text{si } i\leq k-1 \\ i-2C\left(j \bmod 2^{i-k+1}\right) & \text{si } i>k-1. \end{cases}$$

Preuve. Les deux cas étant symétriques, on se contente d'étudier le cas $i\leq k-1$.

Le plus court chemin entre $v_{0,0}$ et $v_{i,j}$ passe par les couches $0,\ldots,i-1$. Donc, pour changer les bits à 1 de j correspondant à ces positions, il n'est pas nécessaire de passer par des couches supplémentaires. Par contre, la partie $\lfloor\frac{j}{2^i}\rfloor$ impose

certains détours dans le chemin. Le détour peut se faire en traversant les couches $s, s-1, \ldots, s-k+i+1$ ou les couches $i, i+1, \ldots, k-1$.

Il est facile de voir que le plus court détour correspond à celui qui évite de parcourir une suite de couches qui correspond à la plus grande liste de bits nuls de $\lfloor \frac{j}{2^i} \rfloor$. La distance est ainsi

$$i + 2\left(k - 1 - i - C\left(\lfloor \frac{j}{2^i} \rfloor\right)\right).$$

□

Ce lemme permet d'obtenir la formule

$$\delta = \frac{3}{2}(k-1) + \frac{C_{k-1}}{k-1} - \frac{2}{k-1}\sum_{i=1}^{k-1} C_i$$

où C_i est la moyenne de $C(j)$ pour des entiers j de i bits (entre 0 et $2^i - 1$). Les premières valeurs de C_i sont :

$$\begin{aligned} C_0 &= 0 \\ C_1 &= 1/2 \\ C_2 &= 4/4 \\ C_3 &= 11/8 \\ C_4 &= 27/16 \\ C_5 &= 62/32 \end{aligned}$$

Ceci permet de calculer la distance moyenne dans $G'_{k,2k-3}$ pour les petites valeurs de k et d'en déduire les bornes inférieures de complexité :

k	2	3	4	5	6
$\delta(G'_{k,2k-3})$	1	2	73/24	265/64	423/80
$pC[\mathcal{L}^0(G'_{k,2k-3}, \bowtie, \gamma)]$	2	4	8	16	31
$pC[\mathcal{A}^0(G'_{k,2k-3}, \bowtie, \gamma)]$	2	4	8	14	27

On constate que δ n'est pas beaucoup plus grand que $\frac{3}{2}(k-1)$ car C_j est petit. En effet, on a

$$C_i = \sum_{j=1}^{i} Prob[C(x) \geq j]$$

mais on a $Prob[C(x) \geq j] \leq \frac{j}{2^i}$, donc

$$\begin{aligned} C_i &\leq j_0 + \sum_{j=j_0+1}^{i} \frac{j}{2^i} \\ &\leq j_0 + \frac{i}{2^{j_0}} - \frac{i}{2^i} \end{aligned}$$

donc, en prenant $j_0 = \log_2 i$, on a $C_i \leq \log_2 i + 1$. Ceci montre que la distance moyenne est équivalente à $\frac{3}{2}(k-1)$.

On a donc

Théorème 26.

$$pC[\mathcal{L}^0(G'_{k,2k-3}, \bowtie, \gamma)] \geq \frac{1}{3}2^k$$

et

$$pC[\mathcal{A}^0(G'_{k,2k-3}, \bowtie, \gamma)] \geq \frac{8}{27}2^k.$$

6.2.3 Bilan

Le tableau suivant récapitule les bornes inférieures obtenues pour la complexité $pC[\mathcal{L}^0(G'_{k,2k-3}, \bowtie, \gamma)]$ et leurs équivalents asymptotiques suivant les différentes méthodes :

k	2	3	4	5	6	asymptote
Analyse spectrale	2	4	8	13	22	$(\pi^2/4k)2^k(1+o(1))$
Utilisation des symétries	2	4	8	14	27	$(1/3)2^k(1+o(1))$

En rappelant que 2^k est une borne supérieure, ceci prouve

$$pC[\mathcal{L}^0(G'_{k,2k-3}, \bowtie, \gamma)] = \Theta(2^k)$$

quand k tend vers $+\infty$.

6.3 Inversion de $g_{k,s}$

Dans cette partie, on étudie la complexité du graphe $G_{k,s}$ qui modélise l'inversion de $g_{k,s}$. La complexité logarithmique, par recherche exhaustive, est 2^{k-1}. Une borne supérieure plus précise, qui sera développée plus loin, est $2^{k-2} + 2^{s-k}$ pour $k-1 \leq s \leq 2(k-1)$.

6.3.1 Etude spectrale

On note la matrice de taille $(s+1).2^{k-1} \times (s+1).2^{k-1}$

$$G_{k,s} = \begin{pmatrix} I & -\frac{1}{2}A & 0 & \cdots & 0 & 0 \\ -\frac{1}{2}{}^tA & 2I & -\frac{1}{2}A & \cdots & 0 & 0 \\ 0 & -\frac{1}{2}{}^tA & 2I & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2I & -\frac{1}{2}A \\ 0 & 0 & 0 & \cdots & -\frac{1}{2}{}^tA & I \end{pmatrix}$$

où A est la même matrice que dans l'étude des points fixes.

Lemme 27. *Pour $s \geq k$, les valeurs propres de $G_{k,s}$ sont de la forme*

$$\begin{array}{ll} 4\sin^2 \dfrac{j\pi}{2(s+1)} & \\ 4\sin^2 \dfrac{\pi + 2j\pi}{2(2i+1)} & \text{pour} \quad i = 1, \ldots, k-1 \\ 4\sin^2 \dfrac{j\pi}{2(i+1)} & \text{pour} \quad i = 1, \ldots, k-1 \end{array}$$

et tout entier j.

Preuve. Dans la base des vecteurs de la forme

$$\begin{pmatrix} 0 \\ \vdots \\ 0 \\ f_\alpha \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

ordonnée suivant les motifs de α, la matrice $G_{k,s}$ s'écrit de la même manière en remplaçant la matrice $\frac{1}{2}A$ par

$$A' = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & J(M_1) & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & J(M_n) \end{pmatrix}$$

où $M_1, \ldots, M_n$ est la liste des motifs (les entiers impairs) et $J(M)$ est, pour un motif M de $k-m-2$ bits, la matrice J_m de taille $m \times m$

$$J_m = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix}.$$

Donc, pour étudier les valeurs propres de $G_{k,s}$, il suffit d'étudier celles des matrices de taille $(s+1).m \times (s+1).m$

$$G_s(J) = \begin{pmatrix} I & -J & 0 & \cdots & 0 & 0 \\ -{}^tJ & 2I & -J & \cdots & 0 & 0 \\ 0 & -{}^tJ & 2I & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2I & -J \\ 0 & 0 & 0 & \cdots & -{}^tJ & I \end{pmatrix}$$

où J est une matrice de taille $m \times m$ qui est J_m ou la valeur 1.

Soit λ une valeur propre de $G_s(J)$ et X un vecteur propre associé. On note

$$X = \begin{pmatrix} X_0 \\ \vdots \\ X_s \end{pmatrix}.$$

Dans le cas $J = 1$, en convenant $X_{-1} = X_0$ et $X_s = X_{s+1}$, la suite des X_i vérifie la relation de récurrence

$$X_{i+1} = (2 - \lambda).X_i - X_{i-1}.$$

Les valeurs propres de $G_{k,s}$ étant comprises entre 0 et 4, on peut poser $\lambda = 4\sin^2\frac{\theta}{2}$. Dans la suite, on note $z = e^{i\theta}$ une racine complexe de $z^2 - (2-\lambda)z + 1$ (l'autre racine est $\bar{z}$). Il existe α et β non nuls tels que

$$X_i = \alpha z^{i+1} + \beta \bar{z}^{i+1}.$$

Des relations $X_{-1} = X_0$ et $X_s = X_{s+1}$ on déduit

$$\begin{vmatrix} (1-z)z^{s+1} & (1-\bar{z})\bar{z}^{s+1} \\ 1-z & 1-\bar{z} \end{vmatrix} = 0$$

donc $z^{2(s+1)} = 1$. On en déduit que la liste des valeurs propres de $G_s(1)$ est la liste des $4\sin^2\frac{j\pi}{2(s+1)}$.

Dans le cas $J = J_m$, avec les mêmes notations, on montre facilement que la suite X_i (solution formelle de $G_s(J_m)X = \lambda X$) vérifie une relation de récurrence $X_i = D_i.J_m.X_{i+1}$ où D_i est une matrice diagonale. Notons $d_i^1, \ldots, d_i^m$ les coefficients diagonaux de D_i. Les d_i sont des fractions rationnelles en λ. La relation $X_0 - J_m.X_1 = \lambda.X_0$ montre que les d_0^j sont tous égaux à $\frac{1}{1-\lambda}$. La relation

$$-{}^tJ_m.X_{i-1} + 2X_i - J_m.X_{i+1} = \lambda X_i$$

montre la relation

$$D_i = ((2-\lambda)I_m - {}^tJ_m.D_{i-1}.J_m)^{-1}.$$

Donc, on a

$$d_i^j = \frac{1}{2 - \lambda - d_{i-1}^{j+1}}$$

pour $j < m$ et $d_i^m = \frac{1}{2-\lambda}$. La dernière relation $-{}^tJ_m.X_{s-1} + X_s = \lambda X_s$ montre $X_s = D_s.J_m.0$ où

$$D_s = ((1-\lambda)I_m - {}^tJ_m.D_{s-1}.J_m)^{-1}$$

donc

$$d_s^j = \frac{1}{1 - \lambda - d_{s-1}^{j+1}}$$

pour $j < m$ et $d_s^m = \frac{1}{1-\lambda}$.

Ceci montre que les valeurs propres de $G_s(J_m)$ sont les pôles des fractions rationnelles d_i^j. Les relations de récurrences sur les d_i^j montrent qu'il y a trois types de fractions rationnelles que l'on notera Q_i, R_i et S_i pour $i = 1, \ldots, m$. On a

$$d_i^j = \begin{cases} Q_{i+1} & \text{si } i+j \leq m \\ R_{m-j+1} & \text{si } i+j > m \text{ et } i < s \\ S_{m-j+1} & \text{si } i = s \end{cases}$$

avec

$$\begin{aligned} Q_1 &= \frac{1}{1-\lambda} \quad \text{et} \quad Q_{i+1} = \frac{1}{2-\lambda-Q_i} \\ R_1 &= \frac{1}{2-\lambda} \quad \text{et} \quad R_{i+1} = \frac{1}{2-\lambda-R_i} \\ S_1 &= \frac{1}{1-\lambda} \quad \text{et} \quad S_{i+1} = \frac{1}{1-\lambda-R_i}. \end{aligned}$$

On visualise le calcul des d_i^j sur une matrice, i étant l'indice de ligne entre 0 et s et j étant l'indice de colonne entre 1 et m :

$$\begin{pmatrix} Q_1 & Q_1 & \cdots & Q_1 & Q_1 \\ Q_2 & Q_2 & \cdots & Q_2 & R_1 \\ Q_3 & Q_3 & \cdots & R_2 & R_1 \\ \vdots & \vdots & \iddots & \vdots & \vdots \\ Q_m & R_{m-1} & \cdots & R_2 & R_1 \\ R_m & R_{m-1} & \cdots & R_2 & R_1 \\ \vdots & \vdots & & \vdots & \vdots \\ R_m & R_{m-1} & \cdots & R_2 & R_1 \\ S_m & S_{m-1} & \cdots & S_2 & S_1 \end{pmatrix}.$$

Si l'on note Q_i^d, R_i^d et S_i^d les dénominateurs de ces fractions rationnelles, elles vérifient une relation de récurrence. Par exemple, on a $Q_0^d = 1$, $Q_1^d = 1-\lambda$ et

$$Q_{i+1}^d = (2-\lambda)Q_i^d - Q_{i-1}^d.$$

Donc, on obtient

$$Q_i^d = \frac{z^{i+1}}{z+1} + \frac{\bar{z}^{i+1}}{\bar{z}+1}.$$

De même, on obtient

$$\begin{aligned} R_i^d &= \frac{z^{i+2}}{z^2-1} + \frac{\bar{z}^{i+2}}{\bar{z}^2-1} \\ S_i^d &= Q_i^d. \end{aligned}$$

On a donc

$$\begin{aligned} Q_i^d &= \frac{\cos\left(i+\frac{1}{2}\right)\theta}{\cos\frac{\theta}{2}} \\ R_i^d &= \frac{\sin(i+1)\,\theta}{\sin\theta}. \end{aligned}$$

Les racines de ces fonctions sont bien les valeurs annoncées dans le lemme. □

Ceci permet d'énoncer le lemme suivant :

Lemme 28. *On a*

$$\lambda_2(G_{k,s}) = \begin{cases} 4\sin^2\frac{\pi}{2(2k-1)} & \text{si } k \leq s < 2k-1 \\ 4\sin^2\frac{\pi}{2(s+1)} & \text{si } s \geq 2k-1. \end{cases}$$

On en déduit, grâce au théorèmes 9 et 15 :

Théorème 29. *Pour $s = k$, on a*

$$pC[\mathcal{L}^0(G_{k,k}, \bowtie, \gamma)] \geq \left((k+1)\sin^2\frac{\pi}{4k-2}\right)2^{k-1}$$

et

$$pC[\mathcal{A}^0(G_{k,k}, \bowtie, \gamma)] \geq \left(\frac{8(k+1)}{9}\sin^2\frac{\pi}{4k-2}\right)2^{k-1}.$$

L'étude spectrale n'est donc pas assez fine pour pouvoir conclure une complexité au moins égale à $2^{k-2}+1$ dans le cas général. On note cependant les bornes inférieures suivantes :

k	2	3	4	5	6
$pC[\mathcal{L}^0(G_{k,k}, \bowtie, \gamma)]$	2	2	2	3	5
$pC[\mathcal{A}^0(G_{k,k}, \bowtie, \gamma)]$	2	2	2	3	5

6.3.2 Liens avec $G'_{k,s}$

Il n'est hélas pas possible d'utiliser le théorème 21 pour le graphe $G_{k,s}$, car il n'est pas transitif pour les arêtes. On peut cependant utiliser l'analyse faite sur le graphe $G'_{k,s}$ en revêtant $G_{k,s}$ sur celui-ci. Le revêtement doit être tel qu'un ensemble de sommets se projette en un ensemble de cardinal voisin, et de complexité voisine.

On propose ici un revêtement dans le cas particulier où $s = k$. On propose de "voir" $G_{k,s}$ comme deux parties de $s-1$ étages de 2^{k-2} sommets joints de

manière cyclique par la couche d'entrée et la couche de sortie. Ainsi, étant donné un sommet $v_{i,j}$ interne (c'est-à-dire tel que $0 < i < s$) de $G_{k,s}$, on note

$$\phi(v_{i,j}) = \begin{cases} v_{i-1,j/2} & \text{si } j \text{ est pair} \\ v_{2s-i-3 \bmod 2s-4, rev((j-1)/2)} & \text{si } j \text{ est impair} \end{cases}$$

dans le graphe $G'_{k-1,2s-5}$. Les sommets de la couche 0 et s sont ignorés. On montre que pour les sommets internes, ϕ est un homomorphisme de graphe.

Cette réduction permet d'énoncer le théorème suivant :

Théorème 30. *Pour $k \geq 3$, on a*

$$pC[\mathcal{L}^0(G_{k,k}, \bowtie, \gamma)] \geq \frac{2^{k-1}}{3}\left(1 - \left(\frac{1}{k-2}\right)^2\left(3 + \frac{2}{2^{k-2}}\right)^2\right).$$

Preuve. Soit $s = k$. On note

$$B_j = \{v_{0,j}, v_{0,j\oplus 1}, v_{1,j}, v_{1,j\oplus 1}\}$$

et

$$C_j = \{v_{s,j}, v_{s,j\oplus 1}, v_{s-1,j}, v_{s-1,j\oplus 1}\}.$$

Si A est un ensemble de sommets de $G_{k,s}$, on dit que A vérifie la propriété P si pour tout j, l'intersection $B_j \cap A$ est $\emptyset$ ou B_j, et l'intersection $C_j \cap A$ est $\emptyset$ ou C_j.

Si A vérifie la propriété P, les sommets non internes ne jouent aucun rôle sur le périmètre de A (l'ensemble des arêtes liant un sommet de A à un sommet hors de A). Cela entraîne que $G'(\phi(A)) = G(A)$.

Il est facile d'encadrer le cardinal de $\phi(A)$:

$$Card(A) - 3.2^{k-1} \leq Card\,(\phi(A)) \leq Card(A).$$

En utilisant le théorème 21 et le calcul de la distance moyenne du théorème 26, on a

$$G(A) \geq \frac{4}{3(k-2)}\,Card\,(\phi(A))\left(1 - \frac{Card\,(\phi(A))}{(2s-4)2^{k-2}}\right).$$

Dans une résolution linéaire, si A est l'ensemble des sommets résolus, on peut transformer l'algorithme de résolution sans augmenter sa complexité. Si l'on joint un sommet interne appartenant à un nouveau B_j ou C_j, on peut faire descendre les trois autres sommets pour les joindre immédiatement après. Si l'on joint un sommet non interne appartenant à un nouveau B_j ou C_j, on peut le faire remonter jusqu'à ce qu'un autre sommet du même ensemble soit joint juste après, et faire redescendre les autres sommets du même ensemble à ce niveau.

Un algorithme optimal dans $\mathcal{L}^0(G_{k,s}, \bowtie, \gamma)$ est donc tel qu'au moins une fois sur quatre jointures successives, A vérifie la propriété P. Donc, on a au moins un

A vérifiant la propriété P de cardinal proche de $\frac{(2s-4)2^{k-2}}{2}$ d'au moins $2+3.2^{k-2}$. Ainsi, on a

$$pC[\mathcal{L}^0(G_{k,k},\bowtie,\gamma)] \geq \frac{4}{3(k-2)}c\left(1-\frac{c}{(2s-4)2^{k-2}}\right)$$

avec $c=\frac{(2s-4)2^{k-2}}{2}\pm(2+3.2^{k-2})$ donc

$$pC[\mathcal{L}^0(G_{k,k},\bowtie,\gamma)] \geq \frac{2^{k-1}}{3}\left(1-\left(\frac{1}{s-2}\right)^2\left(3+\frac{2}{2^{k-2}}\right)^2\right).$$

On retrouve donc le coefficient $\frac{1}{3}$. □

Ce théorème ne commence à donner des bornes inférieures non nulles qu'à partir de $k \geq 6$. On obtient notamment $pC[\mathcal{L}^0(G_{6,6},\bowtie,\gamma)] \geq 5$.

6.3.3 Bilan

Le tableau suivant récapitule les bornes inférieures obtenues pour la complexité $pC[\mathcal{L}^0(G_{k,k},\bowtie,\gamma)]$ et leurs équivalents asymptotiques suivant les différentes méthodes :

k	2	3	4	5	6	asymptote
Analyse spectrale	2	2	2	3	5	$(\pi^2/16k)2^{k-1}(1+o(1))$
Projection sur $G'_{k-1,2k-5}$	0	0	0	0	5	$(1/3)2^{k-1}(1+o(1))$

En rappelant que $2^{k-2}+1$ est une borne supérieure, ceci prouve

$$pC[\mathcal{L}^0(G_{k,k},\bowtie,\gamma)] = \Theta(2^k)$$

quand k tend vers $+\infty$.

6.4 Méthode directe

Dans cette partie, on étudie le problème de l'inversion de $g_{k,s}$ par une méthode directe, adaptée au graphe. Cette méthode, due à Claus Schnorr, permet d'évaluer précisément la complexité linéaire.

6.4.1 Bornes supérieures

On a le théorème suivant :

Théorème 31. *Pour $k-1 \leq s \leq 2k-2$, on a*

$$pC[\mathcal{L}^0(G_{k,s},\bowtie,\gamma)] \leq 2^{k-2}+2^{s-k}.$$

Preuve. On montre dans un premier temps

$$pC[\mathcal{L}^0(G_{k,k-1}, \bowtie, \gamma)] \leq 2^{k-2}.$$

On utilise l'ordre sur les sommets définit par

$$\begin{array}{ccccccccccc}
v_{0,0} & < & v_{0,1} & < & v_{0,2} & < & \cdots & < & v_{0,2^{k-2}-1} & < \\
v_{1,0} & < & v_{1,1} & < & v_{1,2} & < & \cdots & < & v_{1,2^{k-2}-1} & < \\
\vdots & & \vdots & & \vdots & & & & \vdots & \\
v_{k-1,0} & < & v_{k-1,1} & < & v_{k-1,2} & < & \cdots & < & v_{k-1,2^{k-2}-1} & < \\
v_{k-1,2^{k-2}} & < & v_{k-1,2^{k-2}+1} & < & v_{k-1,2^{k-2}+2} & < & \cdots & < & v_{k-1,2^{k-1}-1} & < \\
\vdots & & \vdots & & \vdots & & & & \vdots & \\
v_{1,2^{k-2}} & < & v_{1,2^{k-2}+1} & < & v_{1,2^{k-2}+2} & < & \cdots & < & v_{1,2^{k-1}-1} & < \\
v_{0,2^{k-2}} & < & v_{0,2^{k-2}+1} & < & v_{0,2^{k-2}+2} & < & \cdots & < & v_{0,2^{k-1}-1}. &
\end{array}$$

A la fin de la première ligne, la complexité est à 2^{k-2}. Ensuite, elle n'évolue pas jusqu'à la dernière ligne où elle diminue. On a donc un algorithme linéaire de complexité 2^{k-2}.

Cet algorithme peut s'étendre à $G_{k,s}$ en visualisant ce graphe comme $G_{k,k-1}$ sur les premiers étages suivi d'un empilement de 2^{2k-2-s} graphes $G_{s-k+2,s-k+1}$ sur les derniers étages. Au milieu de l'algorithme précédent, chacun de ces graphes peut être résolu successivement par la même méthode. Soit C_i la chaîne

$$\begin{array}{ccccccccccc}
v_{k-1,i2^{s-k+1}} & < & v_{k-1,i2^{s-k+1}+1} & < & \cdots & < & v_{k-1,i2^{s-k+1}+2^{s-k}-1} & < \\
v_{k,i2^{s-k+1}} & < & v_{k,i2^{s-k+1}+1} & < & \cdots & < & v_{k,i2^{s-k+1}+2^{s-k}-1} & < \\
\vdots & & \vdots & & & & \vdots & \\
v_{s,i2^{s-k+1}} & < & v_{s,i2^{s-k+1}+1} & < & \cdots & < & v_{s,i2^{s-k+1}+2^{s-k}-1} & < \\
v_{s,i2^{s-k+1}+2^{s-k}} & < & v_{s,i2^{s-k+1}+2^{s-k+1}+1} & < & \cdots & < & v_{s,i2^{s-k+1}+2^{s-k+1}-1} & < \\
\vdots & & \vdots & & & & \vdots & \\
v_{k,i2^{s-k+1}+2^{s-k}} & < & v_{k,i2^{s-k+1}+2^{s-k+1}+1} & < & \cdots & < & v_{k,i2^{s-k+1}+2^{s-k+1}-1} & < \\
v_{k-1,i2^{s-k+1}+2^{s-k}} & < & v_{k-1,i2^{s-k+1}+2^{s-k+1}+1} & < & \cdots & < & v_{k-1,i2^{s-k+1}+2^{s-k+1}-1} &
\end{array}$$

pour $i = 0, \ldots, 2^{2k-2-s} - 1$. On utilise l'ordre suivant :

$$\begin{array}{ccccccccc}
v_{0,0} & < & v_{0,1} & < & \cdots & < & v_{0,2^{k-2}-1} & < \\
v_{1,0} & < & v_{1,1} & < & \cdots & < & v_{1,2^{k-2}-1} & < \\
\vdots & & \vdots & & & & \vdots & \\
v_{k-2,0} & < & v_{k-2,1} & < & \cdots & < & v_{k-2,2^{k-2}-1} & < \\
C_0 & < & C_1 & < & \cdots & < & C_{2^{2k-2-s}-1} & < \\
v_{k-2,2^{k-2}} & < & v_{k-2,2^{k-2}+1} & < & \cdots & < & v_{k-2,2^{k-1}-1} & < \\
\vdots & & \vdots & & & & \vdots & \\
v_{1,2^{k-2}} & < & v_{1,2^{k-2}+1} & < & \cdots & < & v_{1,2^{k-1}-1} & < \\
v_{0,2^{k-2}} & < & v_{0,2^{k-2}+1} & < & \cdots & < & v_{0,2^{k-1}-1}. &
\end{array}$$

Cet algorithme de résolution consiste à insérer les résolutions successives des graphes $G_{s-k+2,s-k+1}$ dans la résolution de $G_{k,k-1}$. Un exemple est illustré sur la figure 4.3 dans le cas $k = s = 4$. □

6.4.2 Bornes inférieures

Le problème de la complexité linéaire dans $G_{k,k-1}$ peut être traité comme un problème de flots. On a :

Lemme 32. *Dans $G_{k,k-1}$, soit V_0 l'ensemble des sommets $v_{0,j}$. Pour toute fonction s de V_0 dans $\mathbb{Z}$ de valeur absolue inférieure à 2 et de somme nulle, il existe un flot de source s, c'est-à-dire une fonction f de l'ensemble $\bar{E}$ des arêtes orientées vers $\mathbb{Z}$ telle que :*

$$\begin{aligned} f(v,w) &= -f(w,v) \\ |f(v,w)| &\leq 1 \\ \sum_w f(v,w) &= \begin{cases} s(v) & \text{si } v \in V_0 \\ 0 & \text{sinon.} \end{cases} \end{aligned}$$

Preuve. On définit le flot par récurrence sur chaque couche de telle sorte que le flot provenant de la couche $i-1$ sur un sommet de la couche i soit équitablement réparti sur les deux arêtes allant vers la couche $i+1$. On obtient ainsi, pour $j = j_0.2^i + j_1$ et $j' = j$ ou $j' = j \oplus 2^i$

$$f(v_{i,j}, v_{i+1,j'}) = \frac{1}{2}.\frac{1}{2^i} \sum_{j_2=0}^{2^i-1} s(v_{0,j_0.2^i+j_2}).$$

f étant complétée par symétrie ($f(v,w) = -f(w,v)$), les deux premières conditions sont vérifiées. La troisième est vérifiée par construction sur toutes les couches sauf la dernière. Sur la dernière, elle est vérifiée grâce à la condition d'équilibre sur s. □

Voici une conséquence de ce lemme :

Lemme 33. *Dans le graphe $G_{k,k-1}$, soit A un ensemble de sommets et V_0 l'ensemble des sommets de la première couche. Si $c = Card(A \cap V_0)$, on a*

$$G(A) \geq 2^{k-2} - |2^{k-2} - c|.$$

Preuve. En remplaçant éventuellement A par $V \backslash A$, comme $G(V \backslash A) = G(A)$, on peut supposer $c \leq 2^{k-2}$. Soit $A_0 = A \cap V_0$ et B_0 une partie de V_0 disjointe de A_0 et de même cardinal. On définit

$$s(v_{0,j}) = \begin{cases} 2 & \text{si } v_{0,j} \in A_0 \\ -2 & \text{si } v_{0,j} \in B_0 \\ 0 & \text{sinon.} \end{cases}$$

On relie donc les sommets de A_0 à une source et les sommets de B_0 à un puits.

Le lemme précédent affirme l'existence d'un flot de source s, donc de capacité $2c$. Le théorème de la coupure minimale et du flot maximum montre donc que toute coupure du réseau est de capacité au moins $2c$. Or, A est une coupure de capacité égale à son périmètre, soit $2.G(A)$. On a donc $G(A) \geq c$. □

Théorème 34. *On a :*

$$\begin{aligned} pC[\mathcal{L}^0(G_{k,k-1}, \bowtie, \gamma)] &= 2^{k-2} \\ pC[\mathcal{A}^0(G_{k,k-1}, \bowtie, \gamma)] &\geq \frac{2}{3}2^{k-2} \end{aligned}$$

Preuve. Dans un terme représentant un algorithme de résolution, tout sous-terme définit une valeur c comme dans le lemme précédent.

Dans un algorithme linéaire, l'ensemble de ces valeurs c est l'intervalle de tous les entiers de 0 à 2^{k-1}. Le maximum de $2^{k-2} - |2^{k-2} - c|$ est atteint pour $c = 2^{k-2}$ et vaut 2^{k-2}. Donc, la complexité est au moins 2^{k-2}. Comme ceci est aussi une borne supérieure, on a égalité.

Dans un algorithme arborescent, de même que dans la proposition 14, deux valeurs de c consécutives c_1 et c_2 sont telles que $c_1 \leq c_2 \leq 2c_1$. Donc, on a

$$pC[\mathcal{A}^0(G_{k,k-1}, \bowtie, \gamma)] \geq \max_c \min_{c \leq c' \leq 2c} 2^{k-2} - |2^{k-2} - c'|$$

d'où le résultat. □

S'il est difficile d'adapter la méthode des flots dans le cas général, on peut toutefois le faire dans le cas du graphe $G_{k,2k-2}$.

Lemme 35. *Dans $G_{k,2k-2}$, soit V_0 l'ensemble des sommets $v_{0,j}$ et V_s l'ensemble des sommets $v_{2k-2,j}$. Pour toute fonction s de $V_0 \cup V_s$ dans $\mathbb{Z}$ de valeur absolue inférieure à 2 et de somme nulle, il existe un flot de source s, c'est-à-dire une fonction f de l'ensemble $\bar{E}$ des arêtes orientées vers $\mathbb{Z}$ telle que :*

$$\begin{aligned} f(v,w) &= -f(w,v) \\ |f(v,w)| &\leq 1 \\ \sum_w f(v,w) &= \begin{cases} s(v) & \text{si } v \in V_0 \cup V_s \\ 0 & \text{sinon.} \end{cases} \end{aligned}$$

Preuve. Les couches de 0 à $k-1$ définissent un graphe isomorphe à $G_{k,k-1}$ sur lequel on définit une fonction f comme dans le lemme 32 à partir de la couche V_0. De même, les couches de $2k-2$ à $k-1$ définissent un autre graphe isomorphe à $G_{k,k-1}$ sur lequel on prolonge f comme dans le lemme 32 à partir de la couche V_s.

Pour montrer que l'on a un flot, il suffit de montrer que la condition de conservation est vérifiée à la couche $k-1$. Le flot entrant dans un sommet quelconque de la couche $k-1$ en provenant des couches inférieures est

$$\frac{1}{2}.\frac{1}{2^{k-1}}\sum_{j=0}^{2^{k-1}} s(v_{0,j}).$$

Le flot entrant dans un sommet quelconque de la couche $k-1$ en provenant des couches supérieures est

$$\frac{1}{2}.\frac{1}{2^{k-1}}\sum_{j=0}^{2^{k-1}} s(v_{2k-2,j}).$$

Comme s est de somme nulle, la condition est vérifiée. □

On peut alors facilement adapter le lemme 33 :

Lemme 36. *Dans le graphe $G_{k,2k-2}$, soit A un ensemble de sommets et $V_0 \cup V_s$ l'ensemble des sommets de la première et de la dernière couche. On a*

$$G(A) \geq 2^{k-1} - |2^{k-1} - c|$$

en notant $c = Card(A \cap V_0)$.

Ceci entraîne le théorème suivant :

Théorème 37. *On a :*

$$\begin{aligned} pC[\mathcal{L}^0(G_{k,2k-2}, \bowtie, \gamma)] &= 2^{k-1} \\ pC[\mathcal{A}^0(G_{k,2k-2}, \bowtie, \gamma)] &\geq \frac{2}{3}2^{k-1} \end{aligned}$$

Conclusion

La famille de fonction $g_{k,s}$, de par sa symétrie, développe une sécurité pouvant être étudiée. La méthode d'analyse directe montre qu'il faut $s \geq 2k-2$ pour espérer rendre le problème d'inversion de complexité égale à une recherche exhaustive, soit 2^{k-1} (en comptant son logarithme). Pour un tel paramètre, la seconde valeur propre du réseau de calcul est équivalente à $\frac{\pi^2}{s^2}$, ce qui donne l'encadrement

$$pC[\mathcal{A}^0(G_{k,s}, \bowtie, \gamma)] \in [\sim \frac{2\pi^2}{9s}2^{k-1}, 2^{k-1}].$$

Pour $s = k$, en utilisant les symétries du graphe, on obtient l'encadrement

$$pC[\mathcal{L}^0(G_{k,k}, \bowtie, \gamma)] \in [\sim \frac{2}{3}2^{k-2}, 2^{k-2}].$$

De plus, dans les cas $s = k-1$ et $s = 2k-2$, une méthode directement adaptée au graphe utilisant la théorie des flots montre des bornes très fines qui sont exactes dans le cas linéaire :

$$pC[\mathcal{L}^0(G_{k,k-1}, \bowtie, \gamma)] = 2^{k-2} \quad \text{et} \quad pC[\mathcal{A}^0(G_{k,k-1}, \bowtie, \gamma)] \geq \frac{2}{3}2^{k-2}$$
$$pC[\mathcal{L}^0(G_{k,2k-2}, \bowtie, \gamma)] = 2^{k-1} \quad \text{et} \quad pC[\mathcal{A}^0(G_{k,2k-2}, \bowtie, \gamma)] \geq \frac{2}{3}2^{k-1}.$$

Ces résultats sont plutôt encourageants pour faire une étude plus complète de la sécurité de $g_{k,s}$.

Partie II

Les boîtes des primitives cryptographiques

Chapitre 7

Contrôle des diffusions

Cryptanalysis is fun, especially in the morning... at breakfast.
Lars Knudsen

Dans l'attaque d'une primitive cryptographique, on peut tenter d'utiliser des valeurs particulières sur certaines boîtes qui soient telles que l'on arrive à contrôler la diffusion des autres informations. Dans ce chapitre, on étudie le cas de la fonction de hachage MD4, proposée par Ronald Rivest [27].

Cette fonction est construite à partir d'une fonction de chiffrement C par le procédé de Donald Davies et Carl Meyer (voir le paragraphe 2.2.2). La fonction C utilise trois tours. On montrera que l'on peut construire des collisions sur les deux premiers tours : si C' est la fonction C restreinte aux deux premiers tours, pour toute valeur initiale v, on peut trouver deux blocs de message x et x' tels que $C'_x(v) = C'_{x'}(v)$. Pour la fonction C, on obtient des valeurs $C_x(v)$ et $C_{x'}(v)$ proches au sens de la distance de Hamming. On dit alors que la fonction C n'est pas *correlation-free* au sens de Ross Anderson [29].

Cette attaque de MD4 fait partie d'une étude commandée par le CELAR publiée avec son autorisation [43].

7.1 Description de MD4

7.1.1 Schéma général

La fonction de chiffrement C utilise une valeur v de 128 bits codés sur 4 registres de 32 bits a, b, c et d. Ces registres sont modifiés par l'action d'un bloc x de 512 bits codés sur 16 valeurs $x_1, \ldots, x_{16}$ de 32 bits en 3 tours suivant le schéma de la figure 7.1.

Dans le tour i, les x_k sont mélangés suivant une permutation σ_i pour alimenter des boîtes B_i^j pour $i = 1, 2, 3$. C'est $x_{\sigma_i(j)}$ qui alimente B_i^j. σ_1 est la fonction

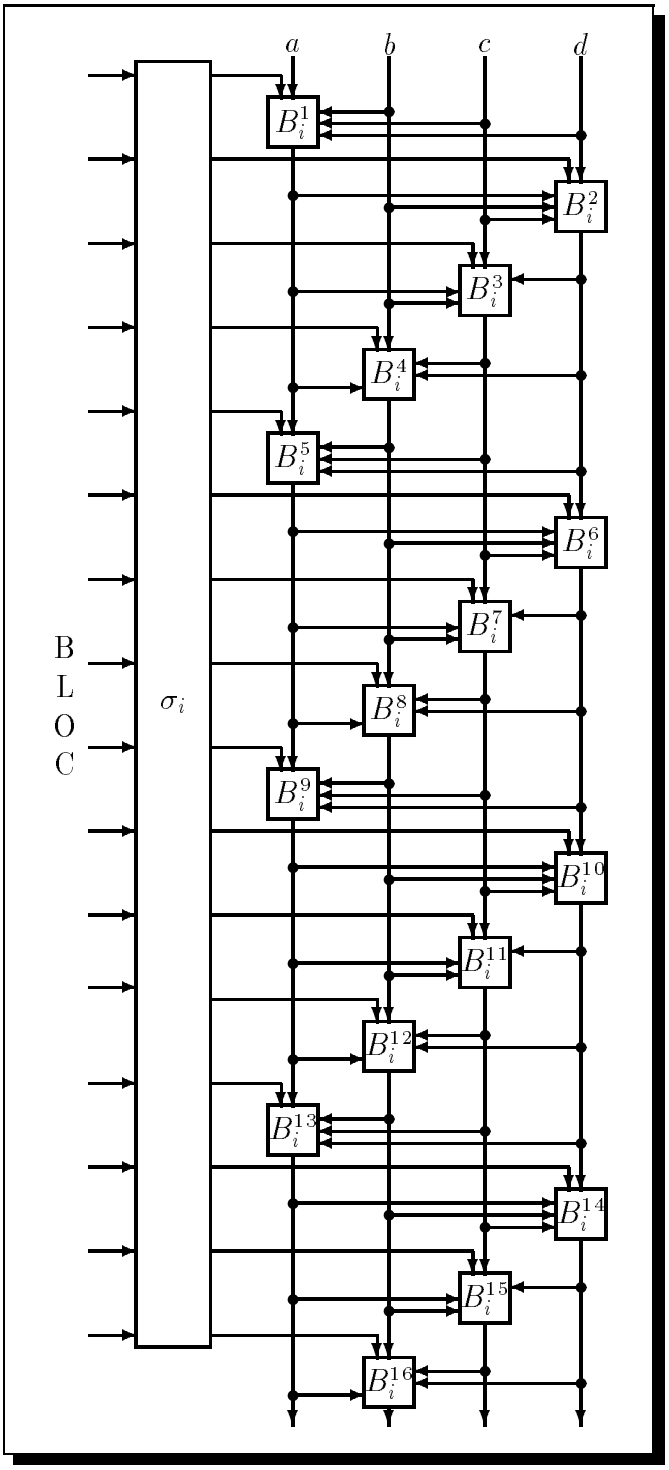

Figure 7.1 : Tour i de la fonction de chiffrement.

$C_x(r_3, r_2, r_1, r_0)$
    Pour $i = 1$ jusqu'à 3
        Pour $j = 1$ jusqu'à 16
            $r_{j+2 \bmod 4} \leftarrow B_i^j(r_{j+2 \bmod 4}; x_{\sigma_i(j)}; r_{j+1 \bmod 4}, r_{j \bmod 4}, r_{j-1 \bmod 4})$
        Fin pour j
    Fin pour i
    Retourne (r_3, r_2, r_1, r_0)
Fin C.

Figure 7.2 : Fonction de chiffrement de MD4.

identité. On a

$$\sigma_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 1 & 5 & 9 & 13 & 2 & 6 & 10 & 14 & 3 & 7 & 11 & 15 & 4 & 8 & 12 & 16 \end{pmatrix}$$

et

$$\sigma_3 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 1 & 9 & 5 & 13 & 3 & 11 & 7 & 15 & 2 & 10 & 6 & 14 & 4 & 12 & 8 & 16 \end{pmatrix}.$$

Les boîtes B_i^j admettent une alimentation (un x_k), une entrée principale (le registre à modifier), et trois entrées latérales (les trois autres registres dans l'ordre des dernières modifications effectuées). En notant $(a, b, c, d) = (r_3, r_2, r_1, r_0)$, la fonction C est décrite par le programme suivant le schéma de la figure 7.2.

7.1.2 Les boîtes B_i^j

On définit les boîtes par :

$$B_i^j(a; x; b, c, d) = R^{\alpha_{i,j}}(a + F_i(b, c, d) + x + k_i)$$

où $+$ est l'addition modulo 2^{32}, R^α est la rotation circulaire de α bits vers la gauche (par exemple, $R^4(3) = 48$), $\alpha_{i,j}$ et k_i sont des constantes. Les fonctions F_i sont des fonctions booléennes bit-à-bit : le bit en position ℓ du résultat d'une fonction booléenne f_i sur les bits de position ℓ de b, c et d.

On dit que x est une alimentation, a est l'entrée principale, et b, c et d sont les entrées latérales. Le rôle de ces boîtes est de perturber un registre (l'entrée principale) en diffusant l'alimentation et les entrées latérales.

On constate que si l'on modifie uniquement l'entrée principale ou l'alimentation, la sortie est toujours modifiée, d'après la régularité de la loi $+$. La diffusion de l'entrée principale et de l'alimentation est donc parfaite.

Fonction multiplexeur

La fonction f_1 est la fonction multiplexeur :

$$f_1(u,v,w) = \begin{cases} v & \text{si } u = 1 \\ w & \text{si } u = 0. \end{cases}$$

Il est très facile de contrôler des variations autour des boîtes B_1^j : comme on a $f_1(u,v,v) = v$, $f_1(0,u,v) = v$ et $f_1(1,v,u) = v$ indépendamment de u, ces propriété permettent de faire varier une entrée latérale de B_1^j sans que la sortie soit modifiée. On a

$$\begin{aligned} B_1^j(a;x;y,b,b) &= R^{\alpha_{1,j}}(a+b+x+k_1) \\ B_1^j(a;x;0,y,b) &= R^{\alpha_{1,j}}(a+b+x+k_1) \\ B_1^j(a;x;2^{32}-1,b,y) &= R^{\alpha_{1,j}}(a+b+x+k_1) \end{aligned}$$

quel que soit y. La diffusion des entrée latérales dans le premier tour n'est donc pas parfaite.

Fonction majorité

La fonction f_2 est la fonction majorité :

$$f_2(u,v,w) = uv \vee vw \vee wu$$

où $\vee$ est la porte *ou*. On a donc, par exemple, $f_2(u,v,v) = v$ pour tout u. Ainsi, on a

$$\begin{aligned} B_2^j(a;x;y,b,b) &= R^{\alpha_{2,j}}(a+b+x+k_2) \\ B_2^j(a;x;b,y,b) &= R^{\alpha_{2,j}}(a+b+x+k_2) \\ B_2^j(a;x;b,b,y) &= R^{\alpha_{2,j}}(a+b+x+k_2). \end{aligned}$$

La diffusion des entrée latérales dans le second tour n'est donc pas parfaite.

Ou exclusif

La fonction f_3 est le *ou* exclusif :

$$f_3(u,v,w) = u \oplus v \oplus w.$$

La diffusion des entrées latérales dans le troisième tour est donc parfaite.

7.2 Attaque de MD4 sur deux tours

On s'intéresse à la fonction $C'_x(a,b,c,d)$ qui donne la valeur des registres à la fin du deuxième tour dans le calcul de $C_x(a,b,c,d)$. On cherche, pour les valeurs initiales (a_0,b_0,c_0,d_0) des registres choisies dans la construction de la fonction de hachage, deux blocs x et x' tels que $C'_x(a_0,b_0,c_0,d_0) = C'_{x'}(a_0,b_0,c_0,d_0)$.

On cherche x et x' tels que seule la seizième valeur soit changée : on cherche $x'_k = x_k$ pour $k = 1,\ldots,15$ et $x_{16} \neq x'_{16}$. Les propriétés de diffusion sont telles qu'à la fin du premier tour, seul b contient une valeur diffusant x_{16}. La stratégie de l'attaque consiste donc à contrôler la diffusion du registre b dans le second tour.

7.2.1 Contrôle du second tour

On remarque que l'on a

$$\begin{aligned} B_2^j(0;-k_2 \bmod 2^{32};b,0,0) &= 0 \\ B_2^j(0;-k_2 \bmod 2^{32};0,b,0) &= 0 \\ B_2^j(0;-k_2 \bmod 2^{32};0,0,b) &= 0 \end{aligned}$$

pour tout b. Une remarque semblable s'applique à la valeur $2^{32}-1$ lorsque l'alimentation vaut $1-k_2$.

En supposant qu'à l'entrée du second tour, les registres a, c et d contiennent 0 et que les boîtes B_2^j soient alimentées par $-k_2$ pour les indices

$$j = 1,2,3,5,6,7,9,10,11,13,14,$$

le registre b ne se diffuse pas. A la sortie, on a $a=0$, $d=0$ et $c = R^{\alpha_{2,15}}(x_{12}+k_2)$. Ces trois valeurs sont donc indépendantes du registre b entré.

On note que la valeur x_{16} alimente une boîte qui touche au registre b. On peut donc espérer que cette deuxième action de x_{16} annule la précédente pour donner une collision.

L'analyse du second tour impose donc $x_k = -k_2$ pour $k = 1,\ldots,11$.

7.2.2 Analyse du premier tour

Les valeurs de x_k étant fixées pour $k = 1,\ldots,11$, on choisit x_{12} aléatoirement. Si a_1, b_1, c_1 et d_1 sont les valeurs des registres correspondants après le calcul des boîtes B_1^j pour $j = 1,\ldots,12$, on pose

$$\begin{aligned} x_{13} &= -a_1 - F_1(b_1,c_1,d_1) - k_1 \bmod 2^{32} \\ x_{14} &= -d_1 - c_1 - k_1 \bmod 2^{32} \\ x_{15} &= -c_1 - b_1 - k_1 \bmod 2^{32} \end{aligned}$$

et on obtient, en sortie du tour $a = c = d = 0$.

7.2.3 Déroulement de l'attaque

Les valeurs $x_1, \ldots, x_{15}$ étant fixées, en sortie du deuxième tour, seul le contenu de b dépend de x_{16}. Soit $g(x_{16})$ ce contenu. On peut utiliser la méthode ρ (voir le paragraphe 1.2.2) pour trouver une collision sur g.

Si l'on a $g(x_{16}) = g(x'_{16})$, en utilisant x_{16} et x'_{16}, on obtient une collision à la fin du second tour. L'ordre de grandeur du nombre de calculs à effectuer pour trouver une collision sur g est 2^{16}, car g opère sur des nombres de 32 bits. Sur une station de travail, cette attaque nécessite effectivement moins d'un dixième de seconde de calcul.

7.2.4 Le troisième tour

Si le troisième tour commence avec la collision trouvée, seule l'alimentation par x_{16} diffusera une nouvelle variation. Cette alimentation est effectuée en dernier, donc, dans le résultat donné par $C_x(a_0, b_0, c_0, d_0)$, seule la valeur de b est changée par l'utilisation de x'. On note qu'il n'y a pas d'espoir pour obtenir une collision ainsi, car la diffusion de x_{16} par B_3^{16} est parfaite.

Conclusion

Dans une primitive cryptographique, de mauvaises propriétés de diffusion peuvent donner lieu à une attaque par contrôle du flot d'information. Dans le cas de MD4, une attaque efficace est possible sur les deux premiers tours par ces simples remarques. Ceci montre la nécessité de développer le critère de "diffusion parfaite" pour les boîtes d'une primitive.

Chapitre 8

Les multipermutations

La notion de *multipermutation* a été proposée dans [73] dans un travail avec Claus Schnorr, dans le cadre de fonctions à deux entrées et deux sorties. Le but de ces fonctions est d'offrir au graphe de la transformée de Fourier rapide une boîte qui réalise une "bonne" diffusion. Si f est une fonction de Z^2 dans Z^2 bijective, on admet qu'elle diffuse bien si pour tout a, les fonctions $x \mapsto f_i(a,x)$ et $x \mapsto f_i(x,a)$ sont bijectives pour $i=1$ ou $i=2$ (en posant $f=(f_1,f_2)$). Cela signifie que si l'on ne modifie qu'une entrée de f, les deux sorties seront modifiées. Cette propriété rend difficile la recherche des collisions en tentant de contrôler des modifications locales des entrées.

On propose ici une définition plus générale. Une multipermutation apparaît comme une boîte qui diffuse parfaitement chaque entrée. On voit, dans le cas de MD4 (voir chapitre 7) qu'une telle propriété des boîtes est un critère de sécurité.

On présente de multiples propriétés des multipermutations en mettant en évidence leurs liens avec d'autres objets combinatoires existant.

8.1 Définition des multipermutations

8.1.1 Définitions

Une multipermutation sur un ensemble Z est une fonction de Z^r dans Z^n vérifiant certaines propriétés. On parle de (r,n)-multipermutation. On propose les définitions équivalentes suivantes :

Définition 15. Une (r,n)-*multipermutation* sur Z est une fonction f de Z^r vers Z^n telle que deux tuples différents de la forme $(x,f(x))$ ne peuvent pas coïncider sur r positions différentes.

Définition 16. Une (r,n)-*multipermutation* sur Z est une fonction f de Z^r vers Z^n telle que si l'on projette l'ensemble des tuples de la forme $(x,f(x))$ sur r coordonnées quelconques, les $Card(Z)^r$ tuples sont représentés.

Définition 17. Une (r, n)-*multipermutation* sur Z est une fonction f de Z^r vers Z^n telle que l'ensemble des tuples de la forme $(x, f(x))$ est un code de distance minimale $n + 1$.

Ici, la distance entre deux tuples est le nombre de positions où ils diffèrent.

Preuve. Pour une fonction f de Z^r dans Z^n, le nombre de tuples $(x, f(x))$ est $Card(Z)^r$, donc, si l'on observe tous les tuples sur r positions quelconques, il est équivalent de dire que deux tuples ne coïncident pas sur ces positions et que tous les sous-tuples à r coordonnées sont représentés. Ceci montre l'équivalence entre les définitions 15 et 16. Cela montre que l'on peut considérer f comme une fonction de r valeurs prises indifféremment dans les entrées et les sorties vers les autres positions.

On note que pour toute fonction f de Z^r dans Z^n, $n + 1$ est la plus grande distance minimale possible pour le code $\{(x, f(x)); x \in Z^r\}$. En effet, si x et x' ne diffèrent que sur une seule position, les tuples $(x, f(x))$ et $(x', f(x'))$ diffèrent sur au plus $n + 1$ positions. Si deux tuples différents sont à distance au plus n, ils coïncident sur r positions de manière équivalente, ce qui montre l'équivalence entre les définitions 15 et 17. □

Une multipermutation correspond à la notion intuitive de *diffusion parfaite*, dans le sens où modifier peu d'entrées de la fonction, soit t entrées, se répercute par une modification d'un maximum de sorties, soit $n+1-t$, et inversement. Les conséquences de cette propriété sont très utiles en cryptographie.

8.1.2 Valeurs remarquables des paramètres

Multipermutations à une entrée

Lorsqu'il n'y a qu'une entrée, les tuples diffèrent sur toutes les positions. Ainsi, chaque sortie dépend de manière bijective de l'entrée. Ceci montre qu'une $(1, n)$-multipermutation est équivalente à une famille de n permutations de l'entrée.

Multipermutations à deux entrées et une sortie

Lorsqu'il y a deux entrées, chaque sortie est une fonction des entrées qui possède des propriétés de régularité : si x et y sont les entrée, et si $g(x, y)$ représente une sortie, pour tout x_0, l'application $y \mapsto g(x_0, y)$ est une permutation, et pour tout y_0, l'application $x \mapsto g(x, y_0)$ est une permutation. En terme d'opération, on dit que g est *régulière* à droite et à gauche. Dans ce cas, Z munis de g est un *quasi-groupe* en ce sens qu'il manque la propriété d'associativité pour obtenir un *groupe*. En fait, une $(2, 1)$-multipermutation est équivalente à une loi de quasigroupe sur Z.

La table d'un quasigroupe (l'analogue de la table de Cayley pour un groupe) porte un nom : un *carré latin*. Formellement, un carré latin sur Z de cardinal q est une matrice $q \times q$ à coefficients dans Z telle que dans chaque ligne et chaque colonne, tous les éléments de Z sont représentés. Les carrés latins ont été étudiés par Euler au $XVII$-ième siècle. Ils sont largement décrits dans le livre [81].

Multipermutations à deux entrées et plusieurs sorties

Il ne suffit par aux sorties d'une $(2,n)$-multipermutation d'être des lois de quasigroupe pour former une multipermutation. Il faut de plus que toute paire d'entre elles forme une bijection du couple des entrées. Ceci correspond à la notion d'orthogonalité : deux lois de quasigroupe g et h sont orthogonales si $(x,y) \mapsto (g(x,y), h(x,y))$ est une bijection. Ainsi, une $(2,n)$-multipermutation est équivalente à une famille de n lois de quasigroupe deux-à-deux orthogonales. L'existence de tels objets n'est pas certaine. Par exemple, la conjecture d'Euler, démontrée en 1900 seulement, revient à dire qu'il n'existe pas de $(2,2)$-multipermutation sur un ensemble de cardinal 6.

8.1.3 Liens avec d'autres objets

Matrices orthogonales

Plus généralement, une (r,n)-multipermutation est équivalente à une *matrice orthogonale* de type $(q^r, r+n, q, r)$, où q est le cardinal de Z. Une $(M, r+n, q, r)$-matrice orthogonale est une matrice de taille $M \times (r+n)$ sur un ensemble à q éléments telle que toute sous-matrice de taille $M \times r$ contient chacune des q^r lignes possibles exactement $\frac{M}{q^r}$ fois. Dans cette équivalence, les lignes de la matrice orthogonale sont les tuples de la forme $(x, f(x))$.

On note que dans le cas où Z a une structure de corps, une (r,n)-multipermutation linéaire est équivalente à un code MDS de taille $r+n$ et de dimension r. Les codes MDS sont notamment étudiés dans [89].

Générateurs aléatoires locaux parfaits

Ueli Maurer et James Massey [78] ont proposé la notion de *générateur aléatoire local parfait*. Une fonction $f = (f_1, \ldots, f_n)$ de Z^r dans Z^n est un générateur aléatoire local parfait d'ordre t si pour tout ensemble d'indices $S = \{i_1, \ldots, i_t\}$ de cardinal t, la fonction $f_S = (f_{i_1}, \ldots, f_{i_t})$ est telle que $f_S(X)$ est une variable aléatoire uniformément distribuée si X l'est.

La valeur maximale de l'ordre d'un tel générateur est $t = r$. On constate que cette notion donne une dissymétrie entre les entrées et les sorties. On peut cependant voir un (r,n) générateur aléatoire local parfait d'ordre r comme une bijection entre Z^r et le graphe d'une $(r, n-r)$-multipermutation.

Objets géométriques

Dans [81], on trouve également des liens entre les multipermutations et d'autres objets combinatoires. Ainsi, on a des liens avec des notions géométriques comme les *k-réseaux* ou les plans projectifs finis. On a également des liens avec des notions utilisées en statistique : les *configurations* (en anglais :*designs*).

8.2 Propriétés élémentaires

8.2.1 Isotopismes

La notion de multipermutation est invariante par la notion suivante :

Définition 18. Deux (r,n)-fonctions f et $g = (g_1, \ldots, g_n)$ sur Z sont dites *isotopes* s'il existe des permutations $\sigma_1, \ldots, \sigma_r$ et $\pi_1, \ldots, \pi_n$ de Z telles que pour tout $x_1, \ldots, x_r$, on a

$$f(x_1, \ldots, x_r) = ((\pi_1 \circ g_1)(\sigma_1(x_1), \ldots, \sigma_r(x_r)), \ldots, (\pi_n \circ g_n)(\sigma_1(x_1), \ldots, \sigma_r(x_r))).$$

La notion d'isotopisme correspond à l'action du groupe $\mathcal{S}_Z^{r+n}$ des permutations sur le graphe des fonctions. La relation d'isotopisme définit une relation d'équivalence. Il est évident qu'une fonction isotope à une multipermutation est une multipermutation. Ceci montre, en particulier, que la distribution uniforme sur l'ensemble des multipermutations est une distribution localement uniforme sur l'ensemble des relations au sens du chapitre 3.

Ainsi, il est très facile de construire une multipermutation à partir d'une autre. Une méthode efficace, pour de telles construction, consiste à prendre une multipermutation équivalente à un code MDS.

8.2.2 Orthomorphismes

La construction de quasigroupes sans groupe est difficile. Dans le cas des fonctions à deux entrées, il sera donc intéressant de munir Z d'une structure de groupe et de chercher des lois de quasigroupe isotopes à la loi de groupe et orthogonales.

Si Z est un groupe, on peut par exemple chercher une permutation σ de Z telle que $(x,y) \mapsto x.y$ et $(x,y) \mapsto \sigma(x).y$ soient orthogonales. Ceci équivaut au fait que $y \mapsto \sigma(x).x^{-1}$ est une permutation. Des permutations σ possédant une telle propriété sont appelées des *orthomorphismes*. L'existence d'orthomorphismes dans des groupes n'est pas systématique, comme le montre le théorème suivant, extrait de [82] :

Théorème 38. *Si le sous-groupe de Sylow d'ordre 2 d'un groupe est cyclique, le groupe n'admet pas d'orthomorphisme. La réciproque est vraie dans un groupe résoluble.*

La validité de la réciproque dans le cas général reste une conjecture. Il est important de noter que le groupe trivial réduit à un élément n'est, par convention, pas cyclique. Ainsi, on trouve dans tout groupe d'ordre impair un orthomorphisme très simple : $x \mapsto x^2$.

La proposition suivante permet de conclure à l'inexistence de multipermutations isotopes à une loi de groupe dans beaucoup de cas :

Proposition 39. *L'existence de deux lois de quasigroupe orthogonales, chacune d'elles isotope à une loi de groupe, est équivalente à l'existence d'orthomorphisme dans l'un ou l'autre des groupes.*

Preuve. Si σ est un orthomorphisme, on a vu que $(x, y) \mapsto x.y$ et $(x, y) \mapsto \sigma(x).y$ sont bien orthogonales.

Inversement, si deux lois de quasigroupe orthogonales sont isotopes à $*$ et à T, on peut se ramener par isotopisme commun aux deux lois orthogonales $(x, y) \mapsto x * y$ et $(x, y) \mapsto \pi(x)T\sigma(y)$. La fonction $(x, y) \mapsto (x * y, \pi(x)T\sigma(y))$ est une permutation.

Etant donné z_0, si l'on fait dépendre x de y de telle manière que l'on ait la relation $\pi(x)T\sigma(y) = z_0$, la fonction $y \mapsto x * y$ doit être une permutation. Autrement dit,

$$y \mapsto \pi^{-1}\left[z_0 T(\sigma(y))^{-1}\right] * y$$

est une permutation (l'inverse de $\sigma(y)$ étant pris au sens de T). Donc, pour tout z_0, la fonction $y \mapsto \pi^{-1}\left[z_0 T(\sigma(y))^{-1}\right] * y$ est un orthomorphisme pour la loi $*$. □

8.3 Quelques constructions

8.3.1 Exemple de $(2, 2)$-multipermutation sur M^ℓ

Une construction simple d'orthomorphisme sur $GF(2)^\ell$ est donnée dans [73]. Elle permet ainsi d'obtenir une $(2, 2)$-multipermutation. On la généralise au cas où $Z = M^\ell$ où M est un module.

Théorème 40. *Si $Z = M^\ell$ où M est un module sur un anneau $\mathcal{A}$, si P est une permutation sur $\{1, \ldots, \ell\}$ et si $c = (c_1, \ldots, c_\ell)$ est un élément de $\mathcal{A}^\ell$, l'application qui à $(x_1, \ldots, x_\ell)$ de Z associe $(y_1, \ldots, y_\ell)$ défini par*

$$y_i = c_i.x_i + x_{P(i)}$$

est un orthomorphisme de Z lorsque l'itération de P sur c fait passer des 0 et des 1 sur chaque coordonnée.

Dans le cas $M = GF(2)$, cet application s'écrit vectoriellement de manière très simple :

$$x \mapsto (x \wedge c) \oplus x_P.$$

Ici, $\oplus$ désigne l'addition vectorielle dans $GF(2)^\ell$ (le *ou* exclusif bit-à-bit), et $\wedge$ désigne le produit bit-à-bit (le *et*).

Par exemple, si $\ell > 1$, on peut prendre un cycle de longueur ℓ pour P et des c_i tels qu'il y ait au moins un 0 et au moins un 1.

On note que le cas $Z = G^\ell$ où G est un groupe abélien est inclus dans ce théorème, puisque l'on peut voir G comme un module sur $\mathbb{Z}$. Plus précisément, il faudra voir G comme un module sur $\mathbb{Z}/_{\mu}\mathbb{Z}$ où μ est l'exposant du groupe.

Pour montrer ce théorème, on montre le lemme suivant :

Lemme 41. *Soient $Z = M^\ell$ le module produit d'un module M sur un anneau $\mathcal{A}$, $c = (c_1, \ldots, c_\ell)$ un élément de $\mathcal{A}^\ell$, et P une permutation sur $\{1, \ldots, \ell\}$. L'application qui à $(x_1, \ldots, x_\ell)$ de Z associe $(y_1, \ldots, y_\ell)$ défini par*

$$y_i = c_i.x_i + x_{P(i)}$$

est une permutation de Z lorsque l'itération de P sur c fait passer des 0 sur chaque coordonnée.

Pour montrer le théorème, on applique ce lemme aux c_i et aux $c_i - 1$ et l'on montre que l'on a un orthomorphisme de M^ℓ.

Preuve. L'application étant linéaire, on se contente d'étudier le système d'équations $c_i.x_i + x_{P(i)} = 0$. En itérant l'égalité ℓ fois, on a

$$x_i = (-1)^\ell x_i \prod_{k=0}^{\ell-1} c_{P^k(i)}.$$

Il y a au moins un coefficient $c_{P^k(i)}$ nul, donc $x_i = 0$ pour tout i. □

8.3.2 Etude des modules

On suppose ici que Z est munis d'une structure de module sur un anneau $\mathcal{A}$. Par exemple, si Z est un groupe abélien, l'anneau associé est $\mathbb{Z}/_{\mu}\mathbb{Z}$ où μ est l'exposant du groupe.

On recherche des (r, n)-multipermutations dont toutes les sorties sont "linéaires" : si $(x_1, \ldots, x_r)$ est la liste des entrées et $(y_1, \ldots, y_n)$ est la liste des sorties, on a

$$y_i = \sum_{j=1}^{r} a_{i,j}.x_j$$

pour des coefficients $a_{i,j}$ dans $\mathcal{A}$. Ainsi, une telle fonction est définie par la $n \times r$ matrice $A = (a_{i,j})$. On note f_A la fonction définie par la matrice A.

Théorème 42. *Si M est un module sur un anneau $\mathcal{A}$, pour toute matrice A, l'application f_A est une multipermutation sur Z si, et seulement si, tous les mineurs de A sont inversibles dans $\mathcal{A}$.*

Preuve. Si l'on regarde r positions, soit r_1 entrées et $r - r_1$ sorties. Si des x coïncident sur les r_1 entrées, on peut considérer les $r - r_1$ sorties comme une fonction affine des $r - r_1$ autres entrées, avec pour matrice la sous-matrice carrée de A de taille $r - r_1$ correspondante. Donc, f_A est une multipermutation si pour toute sous-matrice carrée, l'application qu'elle définit est une bijection. □

Une construction possible, étant donné un module Z, est celle des matrices de Cauchy : on se donne une famille $(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_r)$ d'éléments de $\mathcal{A}$ telle que la différence de deux quelconques d'entre eux soit inversible. Si l'on définit $a_{i,j} = \frac{1}{\alpha_i - \beta_j}$, la matrice définit une multipermutation. En effet, si l'on considère la sous-matrice correspondant aux indices de lignes $i_1 < \ldots < i_s$ et aux indices de colonnes $j_1 < \ldots < j_s$, son déterminant est

$$\det \begin{pmatrix} \frac{1}{\alpha_{i_1} - \beta_{j_1}} & \frac{1}{\alpha_{i_1} - \beta_{j_2}} & \cdots & \frac{1}{\alpha_{i_1} - \beta_{j_s}} \\ \frac{1}{\alpha_{i_2} - \beta_{j_1}} & \frac{1}{\alpha_{i_2} - \beta_{j_2}} & \cdots & \frac{1}{\alpha_{i_2} - \beta_{j_s}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\alpha_{i_s} - \beta_{j_1}} & \frac{1}{\alpha_{i_s} - \beta_{j_2}} & \cdots & \frac{1}{\alpha_{i_s} - \beta_{j_s}} \end{pmatrix} = \frac{\prod_{1 \leq k < \ell \leq s} (\alpha_{i_\ell} - \alpha_{i_k})(\beta_{j_k} - \beta_{j_\ell})}{\prod_{1 \leq k < \ell \leq s} (\alpha_{i_\ell} - \beta_{j_k})}.$$

Par exemple, si Z est un groupe abélien dont le plus petit facteur premier de l'ordre est supérieur ou égal à $r+n$, on peut poser $\alpha_i = i-1$ et $\beta_j = n-1+j$. Les inverses étant calculés modulo l'exposant de Z, on obtient une multipermutation. Cette construction exclut naturellement les groupes d'ordre pair.

Le groupe $GF(2)^\ell$ peut être vu comme un module sur le corps $GF(2^\ell)$. En fait, on a un espace vectoriel. On peut donc appliquer la construction de Cauchy en prenant une famille $(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_r)$ d'éléments de $GF(2^\ell)$ deux-à-deux distincts en prenant les inverses au sens du corps. Cela n'est bien sûr possible que si $2^\ell \geq r + n$.

On constate ainsi que la construction de multipermutations sur un groupe abélien munis d'une structure de module peut s'inspirer de la théorie des codes correcteurs d'erreurs.

8.4 Classification dans le cas linéaire

Les problème général de la classification des multipermutations est un problème trop complexe, puisque l'on ne sait même pas décider du problème de l'existence dans certains cas. On se contente ici de classifier les $(2,2)$-multipermutations isotopes à une multipermutation linéaire f_A définie au paragraphe 8.3.2. On suppose que Z a une structure d'espace vectoriel sur un corps fini K. Les entrées de A sont donc des éléments de $GL_K(Z)$.

On a le théorème suivant :

Théorème 43. *Une $(2,2)$-multipermutation sur un espace vectoriel Z sur K isotope à une multipermutation linéaire est isotope à une multipermutation de la*

forme $(x, y) \mapsto (x + y, x + Ay)$ *où* A *est une matrice qui n'admet ni* 0 *ni* 1 *comme valeur propre, unique à similitude près.*

La classe d'isotopie d'une telle multipermutation est donc caractérisée par les invariants de similitude de A.

Preuve. La notion de similitude n'étant pas dépendante du corps, on peut se ramener au corps $\mathbb{Z}/_{p\mathbb{Z}}$ où p est la caractéristique de K. Dans de tels corps, il suffit de vérifier $\varphi(x+y) = \varphi(x)+\varphi(y)$ pour montrer la linéarité d'une application φ.

La même démonstration que celle de la proposition 39 montre qu'une multipermutation linéaire est isotope à une multipermutation de la forme

$$(x, y) \mapsto (x + y, x + Ay)$$

où $y \mapsto Ay$ est un orthomorphisme, c'est-à-dire, dans ce cas, une matrice qui n'admet ni 0 ni 1 comme valeur propre.

On constate que l'action d'un isotopisme linéaire qui laisse la multipermutation sous cette forme réduite transforme A en une matrice semblable quelconque. Pour démontrer le théorème, il suffit donc de montrer que si deux multipermutations $(x, y) \mapsto (x + y, x + Ay)$ et $(x, y) \mapsto (x + y, x + By)$ sont isotopes, elles sont linéairement isotopes.

Supposons qu'il existe des permutations α, β, γ et δ de Z telles que

$$\begin{aligned} \alpha(x + y) &= \gamma(x) + \delta(y) \\ \beta(x + Ay) &= \gamma(x) + B\delta(y). \end{aligned}$$

D'après la commutativité de l'addition, la première équation permet d'écrire $\delta(y) = \gamma(y) - \gamma(0) + \delta(0)$. Ainsi, en posant

$$\begin{aligned} \gamma'(y) &= \gamma(y) - \gamma(0) \\ \alpha'(z) &= \alpha(z) + \gamma(0) + \delta(0) \\ \beta'(z) &= \beta(z) + \gamma(0) + B\delta(0) \end{aligned}$$

on a un isotopisme de la forme

$$\begin{aligned} \alpha'(x + y) &= \gamma'(x) + \gamma'(y) \\ \beta'(x + Ay) &= \gamma'(x) + B\gamma'(y). \end{aligned}$$

La première équation appliquée en remplaçant x par $x + y$ et y par 0 montre $\gamma'(x + y) = \gamma'(x) + \gamma'(y)$. Donc, γ' est linéaire, α' et β' aussi. Par la suite, α', β' et γ' sont égales et A et B sont semblables. □

On peut se demander si les multipermutations construites au paragraphe 8.3 sont isotopes ou non. Le théorème suivant permet leur classification :

Théorème 44. *Si $Z = M^\ell$ où M est un espace vectoriel sur le corps K, si $(c_1, \ldots, c_\ell)$ est un vecteur de K^ℓ et si P est une permutation, l'application qui à $(x_1, \ldots, x_\ell)$ associe $(y_1, \ldots, y_\ell)$ défini par $y_i = c_i.x_i + x_{P(i)}$ se décompose en espaces cycliques associés aux orbites $O_1, \ldots, O_s$ de P et avec les polynômes minimaux*

$$\prod_{j \in O_i} (X - c_j) - (-1)^{Card(O_i)}.$$

Donc, si P est un cycle, la classe d'isotopie de la multipermutation construite est uniquement définie par $\{\{c_1, \ldots, c_\ell\}\}$. Dans le cas $K = GF(2)$, elle est définie par le poids de Hamming de $(c_1, \ldots, c_\ell)$.

Preuve. En changeant l'ordre des vecteurs de base, on peut supposer que P est produit de cycles $(\ell_{i-1} + 1, \ell_{i-1} + 1, \ldots, \ell_i)$ pour $i = 1, \ldots, s$ et

$$1 < \ell_1 < \ldots < \ell_s = \ell.$$

Ainsi, l'espace engendré par les coordonnées de $\ell_{i-1} + 1$ à ℓ_i admet pour matrice

$$\begin{pmatrix} c_{\ell_{i-1}+1} & 0 & 0 & \cdots & 0 & 1 \\ 1 & c_{\ell_{i-1}+2} & 0 & \cdots & 0 & 0 \\ 0 & 1 & c_{\ell_{i-1}+3} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & c_{\ell_i - 1} & 0 \\ 0 & 0 & 0 & \cdots & 1 & c_{\ell_i} \end{pmatrix}$$

donc l'espace est monogène. Le polynôme minimal de cette matrice est son polynôme caractéristique. En se servant de la formule

$$\det A = \sum_\sigma \epsilon(\sigma) A_{1,\sigma(1)} \ldots A_{r,\sigma(r)},$$

on constate que seules deux permutations σ interviennent. Le polynôme minimal est donc

$$\prod_{j=\ell_{i-1}+1}^{\ell_i} (c_j - X) - 1.$$

□

Conclusion

La notion de multipermutation formalise l'idée de diffusion parfaite. Elle rend donc les attaques par contrôle de diffusion difficiles. Cet objet combinatoire est relié à une multitude d'autres, tels les codes MDS ou les familles de carrés latins orthogonaux, et il est possible d'en construire avec un faible coût. Comme le montre l'exemple de MD4 (voir le chapitre 7), de telles boîtes sont certainement incontournables dans la construction de primitives cryptographiques sûres.

Chapitre 9

Cryptanalyse linéaire

> *I have many times used the discrete exponential or the discrete logarithm as nonlinear cryptographic functions and they have never let me down.*
> James Massey

Dans ce chapitre, on illustre un exemple de cryptanalyse linéaire sur une généralisation de la fonction SAFER. Cette fonction de chiffrement, proposée par James Massey [11] utilise une permutation interne particulière (l'exponentiation discrète). On montre que pour la plupart des autres permutations possibles, la fonction SAFER peut être cassée.

9.1 SAFER

SAFER est une fonction de chiffrement dédiée aux microprocesseurs 8 bits. Elle permet de chiffrer un message de 64 bits avec une clef de 64 bits. La clef étant représentée par 8 entiers $k_1, \ldots, k_8$ sur 8 bits, un procédé de diversification fabrique des sous-clefs $k_1^i, \ldots, k_8^i$. Dans la suite, on a juste besoin de savoir que k_j^i est obtenu à partir de k_j uniquement par une fonction simple (de plus, on a $k_j^1 = k_j$).

Le chiffrement s'opère en 6 tours et demi. Le i-ième tour est illustré sur la figure 9.1. Il utilise les sous-clefs k_j^{2i-1} and k_j^{2i}. Le dernier demi-tour consiste simplement à utiliser les k_j^{13} comme on le ferait dans un 7-ième tour.

$\oplus$ représente le *ou* exclusif bit-à-bit, $+$ représente l'addition modulo 256, P est une permutation définie dans la description de SAFER et Q est sa réciproque. L est une opération linéaire au sens de l'anneau $\mathbb{Z}/256\mathbb{Z}$:

$$L(x, y) = (2.x + y, x + y) \pmod{256}.$$

Dans la description originale de SAFER, P est l'exponentiation en base 45 modulo 257 : les entiers du groupe des unités de $\mathbb{Z}/257\mathbb{Z}$ (de 1 à 256) sont codés

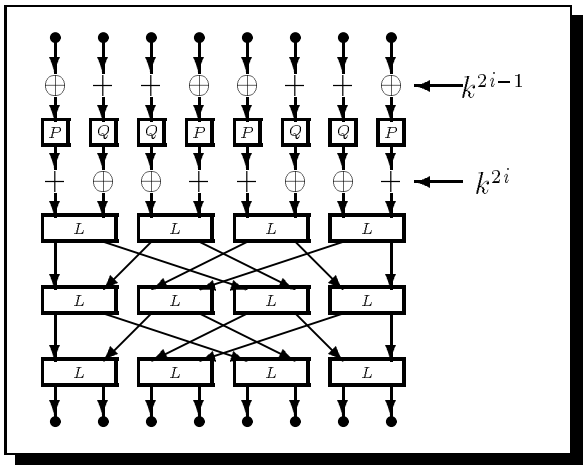

Figure 9.1 : Le i-ième tour de SAFER.

sur 8 bits (256 est codé par 0). 45 est un générateur de ce groupe d'ordre 256, donc on obtient bien une permutation.

Dans une implémentation de SAFER, il semble nécessaire de garder en mémoire les tables de P et Q. Donc, il n'y a pas de raison de considérer l'exponentiation en base 45 plus qu'une autre permutation. Dans la suite, on considère une généralisation de SAFER où l'on envisage n'importe quelle permutation P, et où Q reste sa réciproque. On montre qu'une cryptanalyse linéaire est possible pour au moins 6.1% des permutations P possibles, mais pas pour $\exp_{45}$.

9.2 Cryptanalyse linéaire de SAFER

La cryptanalyse de SAFER se déroule en plusieurs phases. Dans un premier temps, on collectionne une liste de N couples (x, z) de textes clairs x et de textes chiffrés z avec la même clef inconnue k. Dans un second temps, on envisage toute les valeurs possibles d'une certaine information sur la clef (dans la suite, cette information sera le couple (k_3, k_4)) en testant leur vraisemblance dans une analyse statistique. On obtient un ou plusieurs candidats. On peut alors refaire une analyse sur une autre partie de la clef ou faire une recherche exhaustive avec ces candidats.

9.2.1 Remarques générales

La permutation L n'est pas une multipermutation. En effet, on a

$$L_1(x + 128, y) = L_1(x, y)$$

pour tout x et y (L_1 représente la première composante de L). Donc, on a une paire de quadruplets $(x, y, L(x, y))$ à distance de Hamming 2. En fait, il n'y a pas de $(2, 2)$-multipermutations basées sur la structure du groupe $\mathbb{Z}/256\mathbb{Z}$ puisqu'il est cyclique (voir le théorème 38 et la proposition 39).

On utilise une propriété duale de $L_1(x + 128, y) = L_1(x, y)$ en observant

$$L_1(x, y) \equiv y \pmod 2.$$

De même, on a

$$L_1(x, y) + L_2(x, y) \equiv x \pmod 2.$$

Soit F la fonction définie par les trois couches de la figure 9.1 utilisant L. Si $x_1, \ldots, x_8$ est l'entrée du premier tour, sa sortie est $F(y_1, \ldots, y_8)$ où l'on note $(y_1, \ldots, y_8)$ les valeurs sortant des étages réalisant l'opération de confusion, c'est-à-dire $y_1 = P(x_1 \oplus k_1^1) + k_1^2$, ... On remarque que si $F(y_1, \ldots, y_8) = (z_1, \ldots, z_8)$, on a une 2-2 *caractéristique linéaire* :

$$z_3 + z_4 \equiv y_3 + y_4 \pmod 2.$$

Cela signifie que l'on a une dépendance linéaire entre seulement 2 entrées et 2 sorties de F. Il y a cinq autres 2-2 caractéristiques linéaires :

$$\begin{aligned}
z_2 + z_6 &\equiv y_2 + y_6 \pmod 2 \\
z_5 + z_7 &\equiv y_5 + y_7 \pmod 2 \\
z_3 + z_7 &\equiv y_5 + y_6 \pmod 2 \\
z_5 + z_6 &\equiv y_2 + y_4 \pmod 2 \\
z_2 + z_4 &\equiv y_3 + y_7 \pmod 2.
\end{aligned}$$

Si L était une multipermutation, la plus petite a-b caractéristique linéaire serait telle que $a + b = 6$. Cela signifierait qu'il faudrait plus d'information pour pouvoir récupérer une dépendance linéaire.

9.2.2 Fréquence de la caractéristique

Soit $q = \text{Prob}_x [x \equiv P(x) \pmod 2] - \frac{1}{2}$, le biais qui mesure la dépendance entre $x \bmod 2$ et $P(x) \bmod 2$. On obtient le même biais avec Q à la place de P. Si $(x_1, \ldots, x_8)$ est le texte clair, si l'on note

$$\begin{aligned}
y_1 &= P(x_1 \oplus k_1) \\
y_2 &= Q(x_2 + k_2) \\
&\vdots \\
y_8 &= P(x_8 \oplus k_8)
\end{aligned}$$

et si $(z_1, \ldots, z_8)$ est le texte chiffré, soit

$$b(x, z) = (y_3 + y_4 + z_3 + z_4) \bmod 2.$$

Sous des hypothèses heuristiques, on a le lemme probabiliste suivant :

Lemme 45. *Soit $k = (k_3, k_4)$. Si $\phi(k)$ est la parité de la somme de tous les k_3^i et k_4^i pour $i = 2, \ldots, 13$, si $y_3 = Q(x_3 + k_3)$, $y_4 = P(x_4 \oplus k_4)$ et*

$$b(x, z) = (y_3 + y_4 + z_3 + z_4) \bmod 2$$

où z est le chiffré de x suivant une clef inconnue. On a $b(x, z) = \phi(k)$ avec probabilité

$$\frac{1}{2}\left(1 \pm (2q)^{10+e}\right)$$

où e est le nombre d'entiers incorrects dans (k_3, k_4) vis-à-vis de la clef inconnue ($e = 0$ si les deux sont corrects, mais $e = 2$ le plus souvent). L'écart-type de $b(x, z)$ est

$$\frac{1}{2}\sqrt{1 - (2q)^{20+2e}}.$$

Preuve. Grâce à la caractéristique linéaire, si l'on note t_j^i la parité de la somme de l'entrée et de la sortie de la boîte P ou Q en position j dans le tour i, on constate

$$b(x, z) \equiv y_3 + y_4 + y_3' + y_4' + \sum_{i=2}^{6}\sum_{j=3}^{4} t_j^i + \phi(k') \pmod 2$$

où $\phi(k')$ est le vrai $\phi(k)$ et y_3' (respectivement y_4') est le y_3 (respectivement y_4) calculé à partir du vrai k_3 (respectivement k_4).

Le *piling-up lemma* mis en évidence par Mitsuru Matsui [52] exprime que si X et Y sont des variable aléatoires booléennes indépendantes, on a

$$2\,Prob[X \oplus Y = 0] - 1 = (2\,Prob[X = 0] - 1)\,(2\,Prob[Y = 0] - 1)\,.$$

En supposant les entrées des boîtes P et Q indépendantes et uniformément distribuées (cette hypothèse représente une approximation de la réalité vérifiée par l'expérience), ceci permet de montrer

$$\text{Prob}\left[\sum_{i=2}^{6}\sum_{i=3}^{4} t_j^i \equiv 0 \pmod 2\right] = \frac{1}{2}(1 + (2q)^{10})$$

Ceci permet déjà d'obtenir le cas où k_3 et k_4 sont bons.

Si l'un et/ou l'autre de k_3 et k_4 est faux, soit k_3 faux par exemple, il faut prendre en compte la probabilité $\text{Prob}\,[y_3 \equiv y_3'] = \frac{1}{2}(1 + 2q)$ supplémentaire. Cela

permet d'obtenir les cas $e = 1$ et $e = 2$. Le $\pm$ correspond au cas $\phi(k) = \phi(k')$ (cas $+$) ou $\phi(k) \neq \phi(k')$ (cas $-$).

Dans tous les cas, l'écart-type s'obtient par la formule

$$\sigma(b) = \sqrt{E(b)(1 - E(b))}$$

vraie pour les variables aléatoires b à valeurs 0 ou 1. □

Pour un couple (x, z) donné, on a seulement besoin de k_3 et k_4 pour calculer $b(x, z)$. D'après le lemme 45, si k_3 ou k_4 est mauvais, $b(x, z)$ est quasiment uniforme, devant le cas où k_3 et k_4 sont bons qui est biaisé. Ceci permet de distinguer le bon (k_3, k_4) des autres par une analyse statistique.

9.2.3 Analyse statistique

On rappelle le théorème central limite (voir [84] par exemple) :

Théorème 46. *Si B est la moyenne statistique de N variables aléatoires indépendantes de même loi d'espérance μ et d'écart-type σ, on a :*

$$Prob\left[(B - \mu)\frac{\sqrt{N}}{\sigma} \in [a, b]\right] \to \frac{1}{\sqrt{2\pi}} \int_a^b e^{-\frac{t^2}{2}} dt.$$

Soit $B(k_3, k_4)$ la moyenne des $b(x, z)$ sur un échantillon de N couples (x, z). Le lemme 45 montre que l'écart-type des $b(x, z)$ est sensiblement $\frac{1}{2}$. Posons :

$$\lambda_1 + \lambda_2 = \sqrt{N}(2q)^{10}.$$

Le théorème central limite montre que si (k_3, k_4) est mauvais, on a

$$\text{Prob}\left[\left|B(k_3, k_4) - \frac{1}{2}\right| < \frac{\lambda_1}{2\sqrt{N}}\right] \to \frac{1}{\sqrt{2\pi}} \int_{-\lambda_1}^{\lambda_1} e^{-\frac{t^2}{2}} dt;$$

et si (k_3, k_4) est bon, on a

$$\text{Prob}\left[\left|B(k_3, k_4) - \frac{1}{2}\right| < \frac{\lambda_1}{2\sqrt{N}}\right] \to \frac{1}{\sqrt{2\pi}} \int_{\lambda_2}^{\lambda_2 + 2\lambda_1} e^{-\frac{t^2}{2}} dt.$$

Pour tester si un couple (k_3, k_4) est bon, on teste :

$$\left|B(k_3, k_4) - \frac{1}{2}\right| > \frac{\lambda_1}{2\sqrt{N}}.$$

En prenant $\lambda_1 = \lambda_2 = 2$, le bon couple est accepté avec probabilité 98%, et les mauvais sont rejetés avec probabilité 95%.

Ainsi, le nombre d'échantillons nécessaire pour reconnaître le bon (k_3, k_4) avec ces paramètres est

$$N \sim \frac{16}{(2q)^{20}}.$$

Ainsi, pour $|q| \geq 2^{-4}$, c'est plus rapide qu'une recherche exhaustive.

Si l'on restreint le nombre de tours de SAFER à 4, on a $N \sim \frac{16}{(2q)^{12}}$. Donc, pour des permutations P biaisées ($q \neq 0$), cette attaque est plus rapide qu'une recherche exhaustive, et pour $|q| \geq 2^{-4}$, on peut l'implémenter.

9.2.4 Biais d'une permutation

Pour analyser la distribution de q, on utilise le lemme suivant :

Lemme 47. *Si $q = Prob[x \equiv P(x) \pmod 2] - \frac{1}{2}$ où P est une permutation sur l'ensemble $\{0, \ldots, n-1\}$ (on suppose que n est un multiple de 4), nq est toujours un entier pair, et pour tout entier k, on a :*

$$Prob\left[q = \frac{2k}{n}\right] = \frac{\left(\left(\frac{n}{2}\right)!\right)^4}{n!\left(\left(\frac{n}{4}-k\right)!\right)^2\left(\left(\frac{n}{4}+k\right)!\right)^2}$$

pour une permutation P aléatoire uniformément distribuée.

De telles distributions ont été étudiées dans leur généralité par Luke O'Connor [62]. Ici, on utilise ce lemme dans ce cadre particulier d'une permutation sur n valeurs.

Preuve. Si $k + \frac{n}{4}$ est le nombre d'entiers pairs x tels que $P(x)$ soit pair, on a $q = \frac{2k}{n}$. Donc, on a juste à énumérer les permutations pour un $k + \frac{n}{4}$ donné.

On doit choisir quatre ensembles de cardinal $k + \frac{n}{4}$ dans des ensembles de cardinal $\frac{n}{2}$: l'ensemble des entiers pairs qui doivent être envoyés sur des entiers pairs, l'ensemble de leurs images, l'ensemble des entiers impairs qui doivent être envoyés sur des entiers impairs et l'ensemble de leurs images. On a également à choisir deux permutations sur des ensembles de cardinal $k + \frac{n}{4}$ (comment envoyer les entiers pairs sur les entiers pairs et les entiers impairs sur les entiers impairs) et deux permutations sur des ensembles de cardinal $-k + \frac{n}{4}$ (comment envoyer les entiers pairs sur les entiers impairs et les entiers impairs sur les entiers pairs). Donc, le nombre des permutations P est

$$\binom{\frac{n}{2}}{\frac{n}{4}+k}^4 \times \left(\left(\frac{n}{4}+k\right)!\right)^2 \times \left(\left(\frac{n}{4}-k\right)!\right)^2.$$

□

Cela permet de calculer

$$\text{Prob}\left[|q| \geq 2^{-4}\right] \simeq 6.1\%$$

pour $n = 256$ et

$$\text{Prob}\left[q = 0\right] \simeq 9.9\%.$$

Malheureusement (ou heureusement), la permutation P choisie par James Massey n'est pas biaisée, donc cette attaque ne s'applique absolument pas. En fait, $q = 0$ est une propriété de toutes les exponentiations qui sont des permutations :

Lemme 48. *Pour tout générateur g du groupe $(\mathbb{Z}/257\mathbb{Z})^*$, la permutation*

$$x \mapsto g^x$$

n'est pas biaisée (Prob$[x \equiv P(x) \pmod 2] = 0$).

Preuve. On a $(g^{128})^2 \equiv g^{256} \equiv 1 \pmod{257}$ donc g^{128} est 1 ou -1. L'exponentiation en base g est une permutation et $g^0 = 1$, donc on a $g^{128} \equiv -1 \pmod{257}$.

On a $g^{x+128} \equiv -g^x \equiv 257 - g^x \pmod{257}$, donc, on peut partitionner les entiers en paires $\{x, x + 128\}$ (seul le bit de poids fort est modifié) d'entiers de même bit de poids faible telles que la paire de leurs images par P n'a pas le même bit de poids faible. On a donc Prob$[x \equiv P(x) \pmod 2] = 0$. □

9.2.5 Améliorations possibles

Les calculs effectués dans les paragraphes précédent ne sont pas fins : il est possible de gagner en rapidité dans l'attaque par les méthodes suivantes.

On note qu'il est plus important que la bonne clef soit reconnue par le test que de mauvaises clef soient rejetées : il n'est pas gênant de retenir une proportion (pas trop élevée) de mauvaises clefs que l'on pourra tester ultérieurement.

L'algorithme de diversification des clefs permet de calculer $\phi(k)$ à partie de k_3 et k_4. On peut donc faire un test centré sur la véritable moyenne de B. Posons

$$\mu = \frac{1}{2}\left(1 - (-1)^{\phi(k)}(2q)^{10}\right)$$

qui est l'espérance de $b(x, z)$ pour le bon (k_3, k_4). On teste un couple par

$$|B(k_3, k_4) - \mu| < \frac{\lambda_2}{2\sqrt{N}}.$$

On a

$$Prob\left[(k_3, k_4) \text{ accepté}\right] \to \frac{1}{\sqrt{2\pi}} \int_{-\lambda_2}^{\lambda_2} e^{-\frac{t^2}{2}} dt$$

si (k_3, k_4) est bon et

$$Prob\left[(k_3,k_4) \text{ accepté}\right] \to \frac{1}{\sqrt{2\pi}} \int_{\lambda_1}^{\lambda_1+2\lambda_2} e^{-\frac{t^2}{2}} dt$$

si (k_3, k_4) est mauvais.

En choisissant $\lambda_1 = \frac{1}{2}$ et $\lambda_2 = \frac{3}{2}$, la bonne clef est acceptée avec probabilité 95% et les mauvaises sont rejetées avec probabilité 69%. Cela améliore N d'un facteur 4.

On peut également utiliser les amélioration décrites par Burton Kaliski et Matthew Robshaw dans [51] en utilisant toutes les L 2-2 caractéristiques linéaires simultanément. Si la ℓ-ième caractéristique admet en entrée les y_j pour $j \in E_\ell$ et en sortie les z_j pour $j \in S_\ell$ en passant par les sous-clefs k_j^i pour $(i,j) \in C_\ell$, on note

$$b_\ell(x,z,k) = \left(\sum_{j \in E_\ell} y_j + \sum_{j \in S_\ell} z_j + \sum_{(i,j) \in C_\ell} k_j^i \right) \bmod 2$$

où les y sont obtenus à partir de x par action des k_j. On a

$$E(b_\ell(x,z,k)) = \mu = \frac{1}{2}\left(1 - (2q)^{10}\right)$$

si k est bon, et sensiblement $\frac{1}{2}$ sinon.

Si pour $k = (k_2, \ldots, k_7)$ on note $b(x,z,k)$ la moyenne des $b_\ell(x,z,k)$ pour les L caractéristiques et si $B(k)$ est la moyenne des $b(x,z,k)$, on a $\sigma\left(b(x,z,k)\right) \simeq \frac{L}{2}$. Donc, en posant

$$N = \frac{1}{L^2}\left(\frac{\lambda_1 + \lambda_2}{(2q)^{10}}\right)^2,$$

on obtient le même résultat en testant $|B(k) - \mu| < \frac{\lambda_2 L}{2\sqrt{N}}$. En utilisant les $L = 6$ 2-2 caractéristiques cela améliore N d'un facteur $L^2 = 36$.

Ces deux améliorations permettent de gagner un facteur 144 sur N. Ainsi, pour les 6.1% permutations de biais supérieur à 2^{-4}, l'attaque nécessite $2^{56.8}$ couples clairs/chiffrés. Elle est meilleure qu'une recherche exhaustive pour des biais supérieurs à $\frac{7}{128}$, soit pour 10.4% des permutations.

Conclusion

La fonction SAFER utilise des boîtes de diffusion qui ne sont pas des multipermutations. Cela entraîne l'existence de caractéristiques avec peu de boîtes. Cette faiblesse, compensée par une propriété inespérée de la fonction exponentielle, aurait pu mener à une attaque complète de la fonction. On a montré en outre qu'une cryptanalyse linéaire est possible pour 10.4% des permutations, sauf les exponentielles.

Chapitre 10

Critères de non linéarité

La cryptanalyse linéaire a été introduite par Mitsuru Matsui [52]. Un exemple d'une telle analyse est donné dans le chapitre 9. Cette cryptanalyse semblait fortement liée à la cryptanalyse différentielle introduite auparavant par Eli Biham et Adi Shamir dans [44, 47]. Malgré la similitude des deux attaques, les résultats expérimentaux montrent qu'une fonction peut s'attaquer d'une manière et résister à l'autre analyse. Ainsi, la fonction DES [6] a été construite pour résister à l'attaque différentielle. Eli Biham et Adi Shamir l'ont prouvé en montrant que toute légère modification des tables des S-boîtes rend DES vulnérable à leur attaque. De plus, Don Coppersmith, un des experts ayant participé à l'élaboration de DES, a montré dans un rapport rendu public récemment qu'une protection a été mise au moment de la création de DES pour que sa cryptanalyse différentielle nécessite au moins 2^{46} textes clairs connus, ce qui correspond aux meilleurs résultats de Eli Biham et Adi Shamir [49]. Il semble que DES s'attaque plus efficacement par l'approche de Mitsuru Matsui.

Pour une fonction cryptographique construite à partir d'un graphe interprété, les deux types de cryptanalyse reposent fortement sur une certaine mesure des relations utilisées. Les deux types d'attaque donnent une notion de *caractéristique* dans lesquelles certaines boîtes sont déclarées *actives*. La complexité de l'attaque est directement reliée au résultat de certaines mesures δ et λ définies plus loin sur les boîtes actives. Ainsi, dans le chapitre 9, l'attaque de la fonction SAFER repose sur la dépendance entre le bit de poids faible de l'entrée et celui de la sortie d'une relation. Dans ce chapitre, on étudie les relations qui rendent cette mesure optimale pour les deux type d'analyse. Ce travail résulte d'un travail en commun avec Florent Chabaud.

10.1 Définitions et notations

On considère une relation F de p bits d'entrée et q bits de sortie. Formellement, F est une fonction de K^p vers K^q où K est le corps à deux éléments.

On utilise les notations suivantes :

- θ_F est la "fonction caractéristique de F" définie par

$$\begin{aligned} \theta_F : K^p \times K^q &\rightarrow K \\ \theta_F(x,y) &\mapsto \begin{cases} 1 & \text{si } y = F(x) \\ 0 & \text{sinon.} \end{cases} \end{aligned}$$

- Si l'on a une fonction $f : K^p \rightarrow \mathbb{R}$, $\hat{f}$ est la transformée d'Hadamard-Walsh (transformée de Fourier discrète) :

$$\forall w \in K^p \quad \hat{f}(w) = \sum_{x \in K^p} f(x)(-1)^{x.w}$$

où $x.w$ est le produit scalaire sur K et où la somme est au sens du groupe $\mathbb{R}$.

- Si f et g sont deux fonctions sur K^p, $f \otimes g$ est le produit de convolution :

$$\forall a \in K^p \quad (f \otimes g)(a) = \sum_{x \in K^p} f(x)g(a \oplus x)$$

où $\oplus$ est la somme sur K^p, c'est-à-dire le *ou* exclusif bit-à-bit.

- Si l'on a une fonction $f : K^p \rightarrow K$, on note $\chi_f(x) = (-1)^{f(x)}$ sa représentation booléenne avec des ± 1.

Si $F : K^p \rightarrow K^q$ est la fonction à étudier, on définit ses mesures. Pour la cryptanalyse différentielle, on utilise les ensembles

$$D_F(a,b) = \{z \in K^p; F(z \oplus a) \oplus F(z) = b\}$$

où $a \in K^p - \{0\}$ et $b \in K^q$ de cardinal aussi grand que possible. L'efficacité de l'attaque différentielle est basée sur la distribution

$$\delta_F(a,b) = Card\, D_F(a,b).$$

De même, la cryptanalyse linéaire utilise les ensembles

$$L_F(a,b) = \{z \in K^p; a.z \oplus b.F(z) = 0\}$$

où $a \in K^p$ et $b \in K^q \backslash \{0\}$, de cardinal aussi loin de $\frac{1}{2}2^p$ que possible. Son efficacité est donnée par la distribution

$$\lambda_F(a,b) = Card\, L_F(a,b) - \frac{1}{2}2^p.$$

Ainsi, la faiblesse de F est mesurée par :

$$\begin{aligned}\Delta_F &= \sup_{a\neq 0,b} \delta_F(a,b)\\ \Lambda_F &= \sup_{b\neq 0,a} |\lambda_F(a,b)|\end{aligned}$$

Plus ces valeurs sont faibles, plus F résiste aux attaques correspondantes. Dans [60], on trouve d'ailleurs la notion de δ-*uniformité* pour les fonctions telles que $\Delta_F = \delta$.

On propose la définition suivante :

Définition 19. On dit qu'une fonction F de K^p vers K^q est Δ-*résistante* (respectivement Λ-*résistante*) si Δ_F (respectivement Λ_F) réalise le minimum pour de telles fonctions.

On rappelle les propriétés des fonctions *courbes*.

Définition 20. Si p est un entier pair, une fonction booléenne f sur K^p est dite *courbe* si

$$\forall s \in K^p \quad \widehat{\chi_f}(s) = \pm 2^{p/2}.$$

En fait, $2^{p/2}$ (en valeur absolue) est une borne inférieure pour $\sup_{s\in K^p} |\widehat{\chi_f}(s)|$. Donc, les fonctions courbes sont les fonctions qui réalisent cette borne. Cette définition a été généralisée pour des fonctions vectorielles dans [59] :

Définition 21. Une fonction $F : K^p \to K^q$ est *courbe* si pour tout $c \in K^q$, la fonction booléenne $x \mapsto c.F(x)$ est courbe.

Ceci est équivalent à

$$\forall c \neq 0 \quad \forall s \quad \hat{\theta}_F(s,c) = \pm 2^{p/2}$$

car on a $\widehat{\chi_{c.F}}(s) = \hat{\theta}_F(s,c)$. Donc, la valeur $2^{p/2}$ est une borne inférieure pour $\sup_{s\in K^p, c\neq 0} |\hat{\theta}_F(s,c)|$ et les fonctions courbes sont les fonctions qui réalisent cette borne.

10.2 Deux types de résistance

Dans toute la suite, on s'intéresse aux fonctions de K^p dans K^q.

10.2.1 Δ-résistance

La résistance à la cryptanalyse différentielle a largement été étudiée dans la littérature. On rappelle ici quelques résultats.

Lemme 49. *Pour tout couple (a,b) de $K^p \times K^q$, on a $\delta_F(a,b) = (\theta_F \otimes \theta_F)(a,b)$.*

Preuve. On a :

$$\begin{aligned}(\theta_F \otimes \theta_F)(a,b) &= \sum_{x\in K^p, y\in K^q} \theta_F(x,y)\theta_F(a\oplus x, b\oplus y)\\ &= \sum_{x\in K^p} \theta_F(a\oplus x, b\oplus f(x))\\ &= Card\{x\in K^p; b\oplus f(x) = f(a\oplus x)\}\\ &= \delta_F(a,b).\end{aligned}$$

□

Ceci conduit au théorème suivant :

Théorème 50. *Pour toute fonction F, on a $\Delta_F \geq 2^{p-q}$.*

Preuve. On a

$$\begin{aligned}\sum_b (\theta_F \otimes \theta_F)(a,b) &= \sum_{b,x,y} \theta_F(x,y)\theta_F(a\oplus x, b\oplus y)\\ &= \sum_{b,x} \theta_F(a\oplus x, b\oplus f(x))\\ &= \sum_x 1\\ &= 2^p\end{aligned}$$

D'après le lemme, on a donc

$$\sum_{b\in K^q} \delta_F(a,b) = 2^p.$$

Donc, la moyenne des $\delta_F(a,b)$ est 2^{p-q}. L'un d'entre eux est au moins égal à la moyenne. □

On qualifie cette borne d'*absolue*. On remarque qu'elle ne peut pas être atteinte si $p < q$, car elle n'est pas entière. On définit, dans le cas où elle existe, la notion suivante :

Définition 22. Une fonction F est dite *parfaitement non linéaire* si $\Delta_F = 2^{p-q}$.

Lorsqu'elles existent, ce sont donc les fonctions Δ-résistantes.

10.2.2 Λ-résistance

Parallèlement à l'étude de la Δ-résistance, on a :

Lemme 51. *Pour tout couple (a,b) de $K^p \times K^q$, on a $\lambda_F(a,b) = \frac{1}{2}\hat{\theta}_F(a,b)$.*

Preuve. On a :

$$\begin{aligned}\hat{\theta}_F(a,b) &= \sum_{x\in K^p, y\in K^q} \theta(x,y)(-1)^{a.x\oplus b.y} \\ &= \sum_{x\in K^p} (-1)^{a.x\oplus b.F(y)} \\ &= Card\, L_F(a,b) - (2^p - Card\, L_F(a,b)) \\ &= 2\lambda_F(a,b).\end{aligned}$$

□

La théorie des fonctions courbes montre que $2^{p/2}$ est une borne inférieure *absolue* de $\sup|\hat{\theta}_F(a,b)|$. Les fonctions qui atteignent cette borne sont précisément les fonction courbes vectorielles. Donc, lorsqu'elles existent, ce sont les fonctions Λ-résistantes.

10.2.3 Lien entre les bornes absolues

On a le théorème suivant, extrait de [59, 58] :

Théorème 52. *Une fonction est parfaitement non linéaire si, et seulement si, elle est courbe.*

Preuve. Soit $F : K^p \to K^q$ une fonction parfaitement non linéaire. On a, par définition, $\Delta_F = 2^{p-q}$, donc, pour tout $a \neq 0$, en étudiant les cas d'égalité dans la preuve du théorème 50, on a $\delta_F(a,b) = (\theta_F \otimes \theta_F)(a,b) = 2^{p-q}$. D'autre part, $\delta_F(0,0) = 2^p$, et, pour tout $a \neq 0$, on a $\delta_F(a,0) = 0$. On a donc

$$\begin{aligned}(\hat{\theta}_F)^2(a,b) &= (\theta_F \widehat{\otimes} \theta_F)(a,b) \\ &= \sum_{x,y}(\theta_F \otimes \theta_F)(x,y)(-1)^{a.x\oplus b.y} \\ &= 2^p + 2^{p-q}\sum_{x\neq 0,y}(-1)^{a.x\oplus b.y} \\ &= \begin{cases} 2^p & \text{si } b \neq 0 \\ 0 & \text{si } b = 0 \text{ et } a \neq 0 \\ 2^{2p} & \text{si } a = b = 0. \end{cases}\end{aligned}$$

Donc, comme $\hat{\theta}_F(a,b) = \pm 2^{p/2}$, pour tout $(a,b), b \neq 0$, F est courbe.

La réciproque se montre de manière analogue en utilisant les propriétés classiques de la transformation de Walsh :

$$(\theta_F \otimes \theta_F)(a,b) = \frac{1}{2^{p+q}}(\widehat{\theta_F \otimes \theta_F})(a,b) = \frac{1}{2^{p+q}}(\widehat{\hat{\theta}_F})^2.$$

□

Pour déterminer leur existence, on utilise le théorème suivant, extrait de [59]

Théorème 53. *Les fonctions courbes vectorielles existent si, et seulement si, $p \geq 2q$ et p est pair.*

Preuve. Si F est courbe, pour tout $b \neq 0$, $\hat{\theta}_F(a,b) = \pm 2^{\frac{p}{2}}$. Donc, p est pair.

Notons S la somme

$$S = 2^{-\frac{p}{2}} \sum_{b \neq 0} \hat{\theta}_F(0,b).$$

Si r_0 est le nombre de b non nuls tels que $\hat{\theta}_F(0,b) = +2^{p/2}$, on a

$$\begin{aligned} S &= r_0 - (2^q - 1 - r_0) \\ &= 2r_0 - 2^q + 1. \end{aligned}$$

Donc, S est impair.

D'autre part, on a

$$\begin{aligned} \sum_{b \neq 0} \hat{\theta}_F(0,b) &= \sum_b \hat{\theta}_F(0,b) - \hat{\theta}_F(0,0) \\ &= \sum_b \sum_x (-1)^{b.F(x)} - 2^p \\ &= \sum_x \sum_b (-1)^{b.F(x)} - 2^p \\ &= 2^q t - 2^p \end{aligned}$$

où t est le cardinal de $\{x; F(x) = 0\}$. Comme $S = 2^{-\frac{p}{2}}(2^q t - 2^p)$, on a

$$t = 2^{\frac{p}{2}-q}(S + 2^{\frac{p}{2}}).$$

Comme t est entier et S impair, $2^{\frac{p}{2}-q}$ doit être entier. Donc, $p \geq 2q$.

L'existence se montre en utilisant des constructions semblables à celles de la classe de Maiorana-McFarland : pour une permutation $\pi : K^p \to K^p$ et une fonction $f : K^p \to K^p$ quelconque, la fonction $F : K^p \times K^p \to K^p$ définie par

$$F(x,y) = x \times \pi(y) + f(y)$$

où $\times$ est le produit au sens de $GF(2^p)$ est courbe. En tronquant la sortie, on obtient une fonction courbe pour tout $q \leq p$. □

D'après ce théorème, lorsque p est pair et supérieur ou égal à $2q$, Δ-résistance et Λ-résistance sont équivalentes. Avec ces paramètres, de telles fonctions ont largement été étudiée. Les méthodes de construction des fonctions courbes, ainsi que leurs propriétés, sont détaillées dans la thèse de Claude Carlet [55].

Pour $p < 2q$, on doit chercher d'autres bornes.

10.3 Fonctions presque parfaites

10.3.1 Fonctions presque parfaitement non linéaires

La distribution $\delta_F(a,b)$ est toujours paire, car si z vérifie $F(z \oplus a) \oplus F(z) = b$, $z \oplus a$ aussi. Comme les $\delta_F(a,b)$ ne peuvent pas tous être nuls, on a $\Delta_F \geq 2$, ce qui constitue une nouvelle borne inférieure.

On a la notion suivante, extraite de [61] :

Définition 23. On appelle fonction *presque parfaitement non linéaire* une fonction telle que $\Delta_F = 2$.

A cause de la borne absolue $\Delta_F \geq 2^{p-q}$, de telles fonctions ne peuvent exister que si $q \geq p$ (le cas $(p,q) = (2,1)$ est trivial). Dans ce cas, les fonctions Δ-résistantes sont les fonctions presque parfaitement non linéaires.

10.3.2 Fonctions presque courbes

De manière analogue, on a une autre borne inférieure sur Λ_F.

On montre tout d'abord un lemme technique :

Lemme 54. *Pour toute fonction F, on a*

$$\sum_{b \neq 0, a} \hat{\theta}_F^4(a,b) \geq 2^{2p}(3 \times 2^{p+q} - 2^{q+1} - 2^{2p})$$

avec égalité si, et seulement si F est presque parfaitement non linéaire.

Preuve. Rappelons quelques propriétés classiques de la transformation de Walsh. Pour toute fonctions f sur K^n, on a :

$$\begin{aligned} (\hat{f})^2 &= \widehat{f \otimes f} \\ \widehat{(\hat{f})} &= 2^n f \\ \text{et } \sum_a f(a) &= \hat{f}(0). \end{aligned}$$

D'après la définition de λ_F, on a

$$\lambda_F(a,0) = \begin{cases} 2^{p-1} & \text{si } a = 0 \\ 0 & \text{sinon} \end{cases}$$

et d'après la définition de δ_F, on a aussi $\delta_F(0,0) = 2^p$. Donc, pour toute fonction F :

$$\sum_{b \neq 0, a} \hat{\theta}_F^4(a,b) = \sum_{b \neq 0, a} (\theta_F \widehat{\otimes} \theta_F)^2(a,b)$$

$$\begin{aligned} &= \sum_{a,b}(\theta_F \widehat{\otimes} \theta_F)^2(a,b) - \sum_a (\theta_F \widehat{\otimes} \theta_F)^2(a,0) \\ &= [(\widehat{\hat{\delta}_F})^2](0,0) - \sum_a (\hat{\delta}_F)^2(a,0) \\ &= 2^{p+q}[\delta_F \otimes \delta_F](0,0) - 2^4 \sum_a (\lambda_F)^4(a,0). \end{aligned}$$

D'après la définition du produit de convolution, on a

$$\begin{aligned} [\delta_F \otimes \delta_F](0,0) &= \sum_{a,b} \delta_F(a,b)\delta_F(a,b) \\ &= \sum_{a\neq 0,b} \delta_F^2(a,b) + \delta_F^2(0,0). \end{aligned}$$

En rassemblant ces résultats, on obtient

$$\sum_{b\neq 0,a} \hat{\theta}_F^4(a,b) = 2^{p+q} \sum_{a\neq 0,b} \delta_F^2(a,b) + 2^{3p+q} - 2^{4p}.$$

Pour tout entier naturel pair n, on a $n^2 \geq 2n$ avec égalité si, et seulement si $n = 2$ ou $n = 0$. Donc, pour tout $a \neq 0$ et tout b, on a $\delta_F^2(a,b) \geq 2\delta_F(a,b)$ avec égalité si, et seulement si F est presque parfaitement non linéaire. D'autre part, on a

$$\begin{aligned} \sum_{a\neq 0,b} \delta_F(a,b) &= \sum_{a\neq 0}\sum_b \delta_F(a,b) \\ &= \sum_{a\neq 0} 2^p \\ &= 2^p \times (2^p - 1). \end{aligned}$$

Donc

$$\begin{aligned} \sum_{b\neq 0,a} \hat{\theta}_F^4(a,b) &\geq 2^{p+q} \times 2 \times 2^p \times (2^p - 1) + 2^{3p+q} - 2^{4p} \\ &\geq 2^{2p}(3 \times 2^{p+q} - 2^{q+1} - 2^{2p}). \end{aligned}$$

avec égalité si, et seulement si F est presque parfaitement non linéaire. □

On obtient ainsi la borne suivante :

Théorème 55. *Pour toute fonction F, on a*

$$\Lambda_F \geq \frac{1}{2}\left(3 \times 2^p - 2 - 2\frac{(2^p - 1)(2^{p-1} - 1)}{2^q - 1}\right)^{1/2}.$$

Quand la borne est atteinte, on dit que la fonction est presque courbe. *De plus, toute fonction presque courbe est presque parfaitement non linéaire.*

Preuve. Tout d'abord, on remarque

$$\begin{aligned}\Lambda_F^2 &= \sup_{a,b\neq 0} \lambda_F^2(a,b)\\ &= \sup_{a,b\neq 0} \frac{1}{4}(\hat{\theta}_F)^2(a,b)\end{aligned}$$

et pour toute application $N(a,b)$ sur $\mathbb{Z}$, on a

$$M = \sup_{a,b\neq 0} N^2(a,b) \geq \frac{\sum_{a,b\neq 0} N^4(a,b)}{\sum_{a,b\neq 0} N^2(a,b)}.$$

avec égalité si, et seulement si

$$\forall a,b\neq 0 \begin{cases} N(a,b)=0 \\ \text{or } N(a,b)=-\sqrt{M} \\ \text{or } N(a,b)=+\sqrt{M}. \end{cases}$$

On évalue la somme $\sum_{b\neq 0,a} \hat{\theta}_F^2(a,b)$. Pour toute fonction F, on a

$$\begin{aligned}\sum_{b\neq 0,a} \hat{\theta}_F^2(a,b) &= \sum_{b\neq 0,a} \widehat{(\theta_F \otimes \theta_F)}(a,b)\\ &= \sum_{b\neq 0,a} \hat{\delta}_F(a,b)\\ &= \sum_{a,b} \hat{\delta}_F(a,b) - \sum_a \hat{\delta}_F(a,0)\\ &= \widehat{[\hat{\delta}_F]}(0,0) - 4\sum_a \lambda_F^2(a,0)\\ &= 2^{p+q}\delta_F(0,0) - 4\lambda_F^2(0,0)\\ &= 2^{2p}(2^q-1).\end{aligned}$$

Donc, en utilisant le lemme, on a

$$\begin{aligned}4\Lambda_F^2 = \sup_{a,b\neq 0}(\hat{\theta}_F)^2(a,b) &\geq \frac{2^{2p}(3\times 2^{p+q} - 2^{q+1} - 2^{2p})}{2^{2p}(2^q-1)}\\ &\geq \frac{3\times 2^{p+q} - 2^{q+1} - 2^{2p}}{2^q-1}\\ &\geq 3\times 2^p - 2 - 2\frac{(2^p-1)(2^{p-1}-1)}{2^q-1}\end{aligned}$$

avec égalité si, et seulement si F est presque parfaitement non linéaire et

$$\forall a,b\neq 0 \begin{cases} \lambda_F(a,b)=0 \\ \text{or } \lambda_F(a,b)=-\Lambda_F \\ \text{or } \lambda_F(a,b)=+\Lambda_F. \end{cases}$$

□

On note que pour des fonctions presque courbes, $\lambda_F(a,b)$ ne peut prendre que trois valeurs différentes pour $b \neq 0$ qui sont 0, $-\Lambda_F$ et Λ_F. Cette situation est semblable à celle des fonctions courbes où $\lambda_F(a,b)$ prend seulement deux valeurs $-\Lambda_F$ et Λ_F pour $b \neq 0$.

Pour étudier le problème de l'existence des fonctions presque courbes, on montre deux lemmes :

Lemme 56. *Si une fonction $F : K^p \to K^q$ est presque courbe et n'est pas courbe, alors $p \leq q$.*

Preuve. On a la borne absolue

$$\Lambda_F \geq \frac{1}{2} 2^{\frac{p}{2}}.$$

Donc, si F est presque courbe et non courbe, on a

$$\begin{aligned}
\frac{1}{2}\sqrt{\frac{3 \times 2^{p+q} - 2^{q+1} - 2^{2p}}{2^q - 1}} &> \frac{1}{2}\sqrt{2^p} \\
\frac{3 \times 2^{p+q} - 2^{q+1} - 2^{2p}}{2^q - 1} &> 2^p \\
3 \times 2^{p+q} - 2^{q+1} - 2^{2p} &> 2^{p+q} - 2^p \\
2^{p+q+1} - 2^{q+1} - 2^{2p} + 2^p &> 0 \\
2^{q+1}(2^p - 1) - 2^p(2^p - 1) &> 0 \\
q + 1 &> p.
\end{aligned}$$

□

On a le lemme suivant, dû à Julien Cassaigne :

Lemme 57. *Pour tout $q > p$, la quantité*

$$\frac{(2^p - 1)(2^{p-1} - 1)}{2^q - 1} \tag{10.1}$$

n'est pas entière.

Preuve. On a

$$\begin{aligned}
(2^p - 1)(2^{p-1} - 1) &= (2^q - 1)2^{2p-1-q} - (3 \times 2^{p-1} - 2^{2p-1-q} - 1) \\
&= A \times (2^q - 1) - B.
\end{aligned}$$

Comme $q > p$, on a $-2^{2p-1-q} > -2^{p-1}$, donc $3 \times 2^{p-1} - 2^{2p-1-q} > 2^p > 1$ et le reste B est strictement positif.

D'un autre coté, on a

$$\begin{aligned} B < 2^q - 1 &\iff 3 \times 2^{p-1} - 2^{2p-1-q} - 1 < 2^q - 1 \\ &\iff 2^{p-1}(3 - 2^{p-q}) < 2^q. \end{aligned}$$

Comme $q \geq p+1$, on a $2 < 3 - 2^{p-q} < 3$, donc $2^{p-1}(3-2^{p-q}) < 3 \times 2^{p-1}$. Comme $2^{p+\log_2(\frac{3}{2})} < 2^q$, on a bien $B < 2^q - 1$.

On a donc

$$(2^p - 1)(2^{p-1} - 1) = A \times (2^q - 1) - B$$

avec $0 < B < 2^q - 1$, et la quantité (10.1) ne peut être entière si $q > p$. □

On a donc le théorème suivant :

Théorème 58. *Si une fonction $F : K^p \to K^q$ est presque courbe et n'est pas courbe, alors $p = q$ et p est impair. Dans ce cas, on a*

$$\Lambda_F = \frac{1}{2} 2^{\frac{p+1}{2}}. \tag{10.2}$$

Preuve. D'après le lemme 56, on a $q \geq p$. Si $q > p$, d'après le lemme 57, Λ_F n'est pas entier, ce qui est impossible. On a donc $q = p$. Λ_F ne peut être entier que si p est impair. □

Pour construire un exemple de fonction presque courbe, soit le monôme

$$F(x) = x^{2^k+1}$$

sur $GF(2^n)$. Si n est impair, $1 < k < n$ et si n et k sont premiers entre eux, alors F est presque courbe d'après les résultats de [60].

Un exemple de fonction presque parfaitement non linéaire et non presque courbe, dû à Claude Carlet, est la fonction inverse $F(x) = x^{-1}$ de $GF(2^n)$ complétée par $F(0) = 0$. Si n est impair, d'après [60], F est une permutation presque parfaitement non linéaire. Cependant, ce n'est pas une fonction presque courbe, d'après [57].

Conclusion

On a donc étudié les fonction Δ-résistantes et Λ-résistantes dans certains cas :

- si $p \geq 2q$ et si p est pair, Δ-résistant est équivalent à Λ-résistant et à la notion de fonction courbe vectorielle;
- si $p = q$ et si p est impair, Δ-résistant est équivalent à la notion de fonction presque parfaitement non linéaire, Λ-résistant est équivalent à la notion de fonction presque courbe, et Λ-résistant entraîne Δ-résistant sans être équivalent.

Dans tous les cas, on a les bornes

$$\begin{aligned}\Delta_F &\geq \max\left(2^{p-q}, 2\right)\\ \Lambda_F &\geq \max\left(\frac{1}{2}2^{p/2}, \frac{1}{2}\left(3\times 2^p - 2 - 2\frac{(2^p-1)(2^{p-1}-1)}{2^q-1}\right)^{1/2}\right).\end{aligned}$$

L'étude de Δ-résistance et de Λ-résistance est un problème encore ouvert dans les autres cas. En particulier, si $p = q$ et si p est pair, on n'a pas de caractérisation simple de telles fonctions. De même, pour $q < p < 2q$, il existe des fonctions telles que $\Lambda_F = \frac{1}{2}2^{\frac{p+1}{2}}$, mais on ignore s'il existe des fonctions telles que $\frac{1}{2}2^{\frac{p}{2}} < \Lambda_F < \frac{1}{2}2^{\frac{p+1}{2}}$.

Chapitre 11

Cryptanalyse statistique

It is possible to solve many kinds of ciphers by statistical analysis.
Claude Shannon

Depuis la proposition du Standard de Chiffrement de Données (DES) par le gouvernement américain en 1977, de nombreux efforts ont été développés par les experts en cryptanalyse pour attaquer cette fonction. La longueur des clefs secrètes étant de 56 bits, la complexité d'une recherche exhaustive est de l'ordre de 2^{56}.

La première attaque proposée fut une amélioration de la recherche exhaustive d'un facteur 2 par la propriété de *complémentation* [41]. Des articles étudièrent le coût théorique d'une machine parallèle qui serait dédiée à l'attaque de DES [39, 40, 69]. Un tel recourt à la force brutale peut être interprété comme le constat de l'impuissance à lier une quelconque information utilisable entre le message clair, le message chiffré et la clef secrète.

Le principal progrès significatif dans l'attaque de DES fut la cryptanalyse différentielle proposée par Eli Biham et Adi Shamir [44, 47, 48]. L'idée de cette attaque est de constater qu'une corrélation particulière choisie entre une paire de messages clairs se répercute par une corrélation visible sur la paire de messages chiffrés avec une probabilité significativement biaisée. L'attaque consiste donc à saisir les messages chiffrés de paires de messages clairs choisis jusqu'à l'observation du biais. Un tel biais ouvre une brèche dans la fonction de chiffrement par où fuit des informations sur la clef. L'innovation apportée par les idées de Eli Biham et Adi Shamir réside principalement dans l'analyse heuristique permettant la découverte des corrélations efficaces.

La cryptanalyse différentielle, si elle a attaqué avec succès diverses autres fonctions, a plutôt mis en évidence la sécurité de DES [45, 46]. La complexité de l'attaque se mesure par rapport au nombre de messages clairs choisis. Ainsi, la complexité de la meilleur attaque plafonne à 2^{47}, ce qui est tout de même un progrès significatif par rapport à 2^{55}. Eli Biham et Adi Shamir ont surtout montré que d'infimes modifications aléatoires de la structure des S-boîtes de DES

entraînaient un écroulement de la complexité à 2^{38} et moins, ce qui montre que DES était construit pour résister à cette attaque.

Une approche duale de la cryptanalyse différentielle fut proposée par Mitsuru Matsui : la cryptanalyse linéaire [52, 53]. Son idée est de constater la corrélation entre une information sur le message clair et une information sur le message chiffré. L'utilisation de ce biais se fait de manière semblable à la cryptanalyse différentielle, et il n'est pas étonnant qu'une telle approche *a priori* plus simple donne un résultat sensiblement meilleur. L'innovation apportée par Mitsuru Matsui réside, ici aussi, dans l'analyse heuristique permettant la découverte de la corrélation.

La meilleure attaque proposée par la cryptanalyse linéaire a pour complexité 2^{43}. Un exemple d'une telle attaque contre une autre fonction est présenté dans le chapitre 9. La légère amélioration apportée par la cryptanalyse linéaire par rapport à la cryptanalyse différentielle n'est pas surprenante puisqu'elle est plus simple, et il est concevable que DES soit construit pour résister à cette attaque également. L'équivalence entre les deux approches n'étant pas démontrée, il est surprenant qu'une fonction développée dans les années 70 apparaisse résistante à deux types d'attaques compliquées décrites vingt ans plus tard. Ceci laisse penser que DES a été construit pour résister à une notion plus vaste d'attaques englobant les deux précédentes.

Dans ce chapitre, on montre une approche statistique de la cryptanalyse des fonctions de chiffrement. Cette description n'utilise pas les propriétés de linéarité exploitées dans les attaques différentielles et linéaires. Elle définit une notion de résistance de type statistique. On montre que si une cryptanalyse linéaire ou différentielle est efficace, une cryptanalyse semblable sur ce domaine l'est également. De plus, une nouvelle analyse heuristique est proposée. Elle a permis de retrouver une attaque contre DES strictement équivalente à celle de Mitsuru Matsui, de complexité 2^{43}, par une approche disjointe. De plus, il semblerait que cette attaque soit la meilleur, du point de vue de cette approche.

11.1 Attaque statistique

11.1.1 Modèle

On considère une fonction de chiffrement notée E^κ qui a tout message clair m^p associe un message chiffré m^c en utilisant une clef secrète κ. On décompose la fonction en $E^\kappa = E_3^\kappa \circ E_2^\kappa \circ E_1^\kappa$ où E_1^κ est appelé *premier tour* et où E_3^κ est appelé *dernier tour*. Cette décomposition est ici arbitraire, mais dans la plupart des cas, cette notion de premier et de dernier tour correspondra à la structure de tours définie dans la fonction. On note $m^{pi} = E_1^\kappa(m^p)$ et $m^{ci} = (E_3^\kappa)^{-1}(m^c)$ et l'on appelle respectivement message clair *interne* et message chiffré *interne* ces

valeurs.

Une attaque statistique est définie par plusieurs *masques*, qui sont des ensembles de positions de bits. Par convention, on dit que les bits dont les positions ne sont pas dans l'ensemble sont masqués, donc inconnus. Formellement, une attaque est définie par une *caractéristique* $(\Omega, \Omega^{pi}, \Omega^{ci})$ constituée de masques Ω^{pi} et Ω^{ci} sur les messages clairs et chiffrés internes respectivement et d'un masque supplémentaire Ω sur le message clair m^p.

On suppose que l'on dispose d'échantillons indépendants $M = (M^p, M^c)$ tels que $M^c = E^\kappa(M^p)$ vérifiant l'hypothèse suivante :

Hypothèse 59. *$M^p \wedge \bar{\Omega}$ est constant et $M^{pi} \wedge \Omega^{pi}$ est uniformément distribué sur toutes ses valeurs possibles*[1]*.*

Ainsi, soit on a une attaque à message clair connu utilisant un filtre pour obtenir $M^p \wedge \bar{\Omega}$ constant, soit on a une attaque à message clair choisi.

Pour pouvoir observer une distribution sur E_2 restreinte aux masques Ω^{pi} et Ω^{ci}, on utilise une information k sur la clef κ nécessaire pour calculer à la fois $M^{pi} \wedge \Omega^{pi}$ à partir de M^p au travers de E_1 et $M^{ci} \wedge \Omega^{ci}$ à partir de M^c au travers de E_3^{-1}. L'idée principale de l'attaque consiste à dire qu'en faisant ces calculs à l'aide de la bonne valeur de k, la distribution de $(M^{pi} \wedge \Omega^{pi}, M^{ci} \wedge \Omega^{ci})$ apparaît *moins* uniforme que pour de mauvaises valeurs. On distingue ainsi la bonne valeur de k suivant 4 phases :

- **Phase de collection.** On collectionne plusieurs échantillons vérifiant l'hypothèse 59.
- **Phase d'analyse.** Pour chaque valeur candidate de k, on calcule la table de distribution observée sur E_2 restreinte aux masques $(\Omega^{pi}, \Omega^{ci})$ et l'on *note* son uniformité.
- **Phase de classement.** On classe tous les candidats suivant leur note.
- **Phase de recherche.** On essaye de manière exhaustive toutes les clefs suivant leur classement.

11.1.2 Implémentation de l'attaque

Soit Ω^p l'ensemble des positions de bits de m^p nécessaires pour calculer la valeur $m^{pi} \wedge \Omega^{pi}$ au travers de E_1. De même, soit Ω^c l'ensemble des positions de bits de m^c nécessaires pour calculer $m^{ci} \wedge \Omega^{ci}$ au travers de E_3^{-1}. Pour toutes les valeurs de $a = m^p \wedge (\Omega^p \cap \Omega)$ et de $b = m^c \wedge \Omega^c$, on définit un compteur $r_{a,b}$. Dans la phase

[1] $x \wedge \Omega$ est la valeur des bits de x dont les positions sont dans le masque Ω, et $\bar{\Omega}$ représente le masque complémentaire de Ω, c'est-à-dire l'ensemble des positions qui ne sont pas dans Ω. Cela correspond aux opérations logiques *et* et *non* bit-à-bit respectivement.

NOTE
    Initialise tous les $r_{a,b}$ à 0
    Pour n échantillons m, incrémente $r_{m^p \wedge (\Omega^p \cap \Omega), m^c \wedge \Omega^c}$
    Pour chaque candidat k
        Calcule tous les $n_{i,j}(k)$ suivant (11.1)
        Calcule $S(k)$ et mets $D(k)$ à $\frac{S(k)-\mu}{\sigma}$
    Fin pour
Fin NOTE.

Figure 11.1 : Attaque statistique.

de collection, si M est un nouvel échantillon de couple message clair/chiffré, on doit juste incrémenter le compteur $r_{M_\ell^p \wedge (\Omega^p \cap \Omega), M_\ell^c \wedge \Omega^c}$.

Dans la phase d'analyse, pour chaque candidat k, on peut calculer les valeurs $i = M_\ell^{pi} \wedge \Omega^{pi}$ et $j = M_\ell^{ci} \wedge \Omega^{ci}$. Donc, pour chaque k, on peut calculer

$$n_{i,j}(k) = \sum_{\substack{E_1^k(a) \wedge \Omega^{pi} = i \\ \left(E_3^k\right)^{-1}(b) \wedge \Omega^{ci} = j}} r_{a,b}. \tag{11.1}$$

En utilisant une statistique appropriée S, la phase d'analyse consiste, pour chaque k, à calculer $S(k)$ en utilisant tous les $n_{i,j}(k)$. Soit μ et σ l'espérance et l'écart-type de S respectivement dans le cas où M^p et M^c sont indépendants. Pour chaque k, on définit la note $D(k) = \frac{S(k)-\mu}{\sigma}$ du candidat k.

Les deux première phases de l'attaque sont résumées sur la figure 11.1. La complexité en espace est $2^{W(\Omega^p \cap \Omega)+W(\Omega^c)} + 2^w$ où 2^w est le nombre de candidats[2]. La complexité en temps est équivalente à $2^{w+W(\Omega^p \cap \Omega)+W(\Omega^c)} + n$. La complexité en moyenne de la phase de recherche sera étudiée plus ultérieurement.

11.1.3 Analyse de l'attaque

Soient respectivement p et q les nombres de $i = M^{pi} \wedge \Omega^{pi}$ et de $j = M^{ci} \wedge \Omega^{ci}$ possibles. On note $Prob\left[M^{ci} \wedge \Omega^{ci} = j / M^{pi} \wedge \Omega^{pi} = i\right] = \frac{1}{q} + d_{i,j}$ où $d_{i,j}$ représente le biais. La *déviation* de la distribution se définit par

$$d = \sqrt{\sum_{i,j} d_{i,j}^2}.$$

[2] $W(\Omega)$ représente le poids de Hamming du masque Ω, c'est-à-dire son cardinal.

L'analyse précise de l'algorithme dépend du choix de la statistique S, mais on en donne ici les principes. Chaque $d_{i,j}$ étant dans la pratique très proche de 0, pour chaque candidat k, l'espérance et l'écart-type de $D(k)$ sont respectivement proches de 0 et 1. Si k est le bon candidat, $D(k)$ dépend de la distribution induite par E_2 sur les masques Ω^{pi} et Ω^{ci}. Si $k' \neq k$, $D(k')$ dépend de la distribution induite par la fonction

$$(E_3^{k'})^{-1} \circ E_3^k \circ E_2^k \circ E_1^k \circ (E_1^{k'})^{-1}$$

où $(E_3^{k'})^{-1} \circ E_3^k$ et $E_1^k \circ (E_1^{k'})^{-1}$ lissent la distribution de E_2. Ceci motive les approximations suivantes :

Approximation 60. *Si k' est mauvais, l'espérance de $S(k')$ est μ.*

Approximation 61. *Pour tout k (bon ou mauvais), l'écart-type de $S(k)$ est σ.*

Approximation 62. *Toutes les notes $D(k)$ sont deux-à-deux indépendantes.*

Soit μ' la *véritable* espérance de $S(k)$ avec le bon candidat k. On appelle *efficacité* de l'attaque l'espérance $\lambda = \frac{\mu' - \mu}{\sigma}$ de la bonne note. Soit

$$\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-\frac{t^2}{2}} dt$$

la distribution de la loi normale centrée réduite. Dans la suite, l'expression "*la distribution de S est asymptotiquement normale*" signifie que

$$Prob\left[\frac{S - E(S)}{\sigma(S)} < t\right] \to \Phi(t).$$

Théorème 63. *Sous les approximations, si les distributions de tous les $D(k)$ sont asymptotiquement normales, la complexité en moyenne de la phase de recherche tend vers $2^W \Phi\left(-\lambda/\sqrt{2}\right)$ où 2^W est le nombre de clefs κ.*

Preuve. Pour chaque mauvais candidat k', $(D(k') - D(k) - \lambda)/\sqrt{2}$ est centrée réduite, donc

$$Prob\left[D(k') > D(k)\right] \to \Phi\left(-\frac{\lambda}{\sqrt{2}}\right).$$

□

La décroissance de $\Phi\left(-\lambda/\sqrt{2}\right)$ est illustrée sur le tableau suivant :

λ	0	2^{-2}	2^{-1}	1	2	2^2	2^3
$\Phi\left(-\lambda/\sqrt{2}\right)$	50%	43%	36%	24%	$8\% = 2^{-3.7}$	$2^{-8.7}$	$2^{-27.0}$

11.1.4 Utilisation de plusieurs caractéristiques

Il est possible d'utiliser plusieurs caractéristiques C_s (ou la même plusieurs fois, avec des échantillons indépendants) d'efficacité λ_s pour $s = 1, \ldots, c$ en utilisant une astuce analogue à celle utilisée par Burton Kaliski et Matthew Robshaw [51]. On utilise ces idées même lorsque l'information k sur κ est différente pour chaque caractéristique. Chaque clef candidate κ obtient une note $D_s(\kappa)$ relative à chaque caractéristique. On note

$$\bar{\lambda} = \sqrt{\sum_{s=1}^{c} \lambda_s^2} \tag{11.2}$$

et l'on définit la note générale

$$D(k) = \sum_{s=1}^{c} \frac{\lambda_s}{\bar{\lambda}} D_s(k)$$

en utilisant les efficacités comme coefficients de chaque note partielle. On peut alors ensuite effectuer la recherche exhaustive suivant la liste des clefs triées suivant la note générale. Il est facile de démontrer que le théorème 63 reste vrai si l'on remplace λ par $\bar{\lambda}$ en supposant que les $D_s(\kappa)$ sont indépendantes.

11.2 Approche différentielle

Dans le cas suivant, Ω est petit : l'attaque est à message clair choisi. Pour des valeurs non nulles de a et b données, soit

$$S_D = \sum_{\substack{i \oplus i' = a \\ j \oplus j' = b}} n_{i,j} n_{i',j'} = \sum_{\ell \neq \ell'} 1_{M_\ell^{pi} \oplus M_{\ell'}^{pi} = a \text{ et } M_\ell^{ci} \oplus M_{\ell'}^{ci} = b}.$$

S_D compte le nombre de paires (M^{pi}, M^{ci}) de différence masquée (a, b) dans la liste des échantillons. Une analyse heuristique inventée par Eli Biham et Adi Shamir permet d'approcher

$$Prob[M^{ci} \oplus M'^{ci} = b / M^{pi} \oplus M'^{pi} = a] = \frac{1}{q} + \delta.$$

Théorème 64. *Sous l'hypothèse 59, l'efficacité de l'attaque en utilisant S_D est*

$$\lambda \simeq n \frac{q}{\sqrt{2(pq-1)}} \delta \leq n \frac{q}{p\sqrt{2(pq-1)}} d^2.$$

Preuve. On a $\mu = \frac{n(n-1)}{pq}$,

$$\sigma = \frac{\sqrt{n(n-1)}\sqrt{2(pq-1)}}{pq} \text{ et } \mu' = \frac{n(n-1)}{pq} + \frac{n(n-1)\delta}{p}$$

ce qui permet d'obtenir λ. En utilisant l'inégalité de Cauchy-Schwarz, on a

$$|\delta| = \frac{1}{p} \left| \sum_{\substack{i \oplus i' = a \\ j \oplus j' = b}} d_{i,j} d_{i',j'} \right| \leq \frac{d^2}{p}.$$

□

11.3 Approche linéaire

11.3.1 Cryptanalyse linéaire

Dans le cas suivant, Ω est le masque complet : l'attaque est à message clair connu. Pour des valeurs de a et b données, soit

$$S_L = \sum_{i \cdot a = j \cdot b} n_{i,j} = \sum_{\ell=1}^{n} 1_{\left(M_\ell^{pi} \wedge \Omega^{pi}\right) \cdot a = \left(M_\ell^{ci} \wedge \Omega^{ci}\right) \cdot b}$$

où $x \cdot y$ est le produit scalaire, c'est-à-dire $x \cdot y = W(x \wedge y) \bmod 2$. S_L compte le nombre d'échantillons dans un hyperplan défini par a et b. Une analyse heuristique inventée par Mitsuru Matsui permet d'approcher

$$Prob\left[(M^{pi} \wedge \Omega^{pi}) \cdot a = (M^{ci} \wedge \Omega^{ci}) \cdot b\right] = \frac{1}{2} + \delta.$$

Théorème 65. *Sous l'hypothèse 59, l'efficacité de l'attaque en utilisant S_L, qui est asymptotiquement normale, est*

$$\lambda = \sqrt{n}.2\delta \leq \sqrt{n}.\sqrt{\frac{q}{p}}d.$$

Ce théorème sera montré sous une forme plus générale ultérieurement.

11.3.2 L'attaque de Mitsuru Matsui contre DES

Comme cela est illustré sur la figure 11.2, l'attaque de Mitsuru Matsui est définie par la caractéristique

$$\begin{aligned} \Omega &= (\mathrm{ff\ ff\ ff\ ff}, \mathrm{ff\ ff\ ff\ ff}) \\ \Omega^{pi} &= (01\ 04\ 00\ 80, 00\ 00\ 00\ 00) \\ \Omega^{ci} &= (00\ 00\ 80\ 00, 21\ 04\ 00\ 80) \end{aligned}$$

en hexadécimal [53]. Pour des raisons dépendant de la structure de F, les masques sur les sous-clefs sont codés en octal. On a $a = \Omega^{pi}$ et $b = \Omega^{ci}$. Donc, $p = 2^3$ et $q = 2^5$, et le nombre de candidats est 2^{12}. Le biais est approché heuristiquement

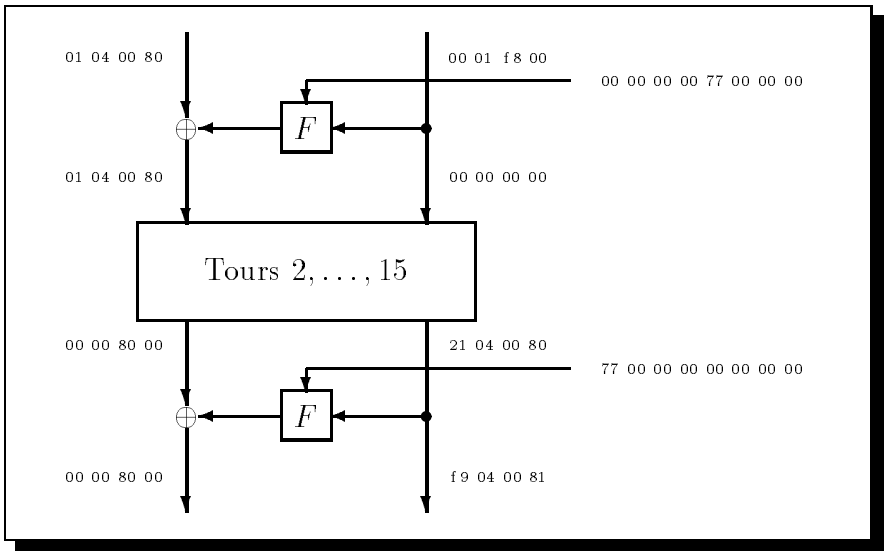

Figure 11.2 : Caractéristique de Mitsuru Matsui.

par $|\delta| = 1.19 \times 2^{-21}$. En utilisant $n = 2^{43}$, on a $\lambda = 3.37$. Donc, en utilisant deux telles caractéristiques (comme le fait Mitsuru Matsui), l'efficacité globale est donnée par l'équation (11.2) et la recherche exhaustive a une complexité améliorée d'un facteur $\Phi(-3.37) = 2^{-11.4}$. La complexité de la phase de recherche est donc $2^{44.6}$.

11.3.3 Test linéaire généralisé

On peut généraliser la statistique S_L par n'importe quelle statistique linéaire : pour des coefficients $a_{i,j}$ donnés, soit

$$S_t = \sum_{i,j} a_{i,j}.n_{i,j} = \sum_{\ell=1}^{n} a_{M_\ell^{pi} \wedge \Omega^{pi}, M_\ell^{ci} \wedge \Omega^{ci}}.$$

Théorème 66. *La statistique S_t est asymptotiquement normale. Sous l'hypothèse 59, la meilleur efficacité est obtenue avec $a_{i,j} = d_{i,j}$. C'est*

$$\lambda = \sqrt{n}.\sqrt{\frac{q}{p}}d.$$

Preuve. On a $\mu = a_1$, $\sigma = a_2\sqrt{n}$, et

$$\mu' = a_1 + \frac{n}{p}\sum_{i,j} a_{i,j}.d_{i,j} \text{ où } a_1 = \frac{1}{pq}\sum_{i,j} a_{i,j} \text{ et } a_2^2 = \frac{1}{pq}\sum_{i,j}(a_{i,j} - a_1)^2.$$

S_t est asymptotiquement normale, d'après le théorème central limite. L'inégalité de Cauchy-Schwarz permet de conclure. □

Pour la caractéristique de Mitsuru Matsui complétée par symétrie définie par

$$\begin{aligned}\Omega &= (\mathtt{ff\ ff\ ff\ ff, ff\ ff\ ff\ ff})\\ \Omega^{pi} &= (21\ 04\ 00\ 80, 00\ 00\ 80\ 00)\\ \Omega^{ci} &= (00\ 00\ 80\ 00, 21\ 04\ 00\ 80)\end{aligned}$$

on a $p = q = 2^5$. La déviation est approchée par $d \simeq 2^{-19.58}$. Donc, en utilisant $2^{42.93}$ messages clairs connus, on a $\lambda = 3.69$. Avec deux telles caractéristiques, la recherche exhaustive est accélérée d'un facteur $\Phi(-3.69) = 2^{-13.14}$ et obtient la complexité $2^{42.86}$. On a ainsi réduit le nombre de messages clairs connus d'un facteur 5% dans l'attaque de Mitsuru Matsui.

11.4 Cryptanalyse χ^2

La déviation par rapport à la distribution uniforme peut être mesurée par la statistique du χ^2 [83] :

$$S_{\chi^2} = \frac{pq}{n}\sum_{i,j}\left(n_{i,j} - \frac{n}{pq}\right)^2 = \frac{pq}{n}\sum_{i,j} n_{i,j}^2 - n.$$

Théorème 67. *Sous l'hypothèse 59, l'efficacité de l'attaque en utilisant S_{χ^2} est*

$$\lambda \simeq n\frac{q}{p\sqrt{2(pq-1)}}d^2.$$

Preuve. Pour les mauvais candidats, la statistique S_{χ^2} tend vers la loi du χ^2 avec $pq-1$ degrés de liberté : $\mu = pq-1$ et $\sigma = \sqrt{2(pq-1)}$. Comme le degré de liberté est grand, on peut approcher cette loi par une loi normale.

Soit

$$S'_{\chi^2} = \frac{pq}{n}\sum_{i,j}\left(n_{i,j} - \frac{n}{pq} - \frac{nd_{i,j}}{p}\right)^2.$$

S'_{χ^2} est une sorte de statistique χ^2 telle que $E(S'_{\chi^2}) = pq - 1 - \frac{qd^2}{p}$. Donc, on a $S_{\chi^2} = S'_{\chi^2} + 2d\sqrt{\frac{qn}{p}}S_t - n\frac{qd^2}{p}$ où S_t est le meilleur test linéaire centré réduit. Donc, on a

$$\mu' = pq - 1 + (n-1)\frac{qd^2}{p}$$

ce qui permet de calculer λ. □

Une conséquence directe de ce théorème est qu'avec la même caractéristique et le même nombre de messages clairs, la cryptanalyse χ^2 est au moins aussi efficace que la cryptanalyse différentielle. Elle est également comparable à l'efficacité de la cryptanalyse linéaire.

Avec cette statistique, on n'a pas besoin d'avoir une idée précise sur l'information qui fuit au travers de E_2 (les valeurs a et b dans les cryptanalyses différentielles et linéaires). Ici, en utilisant la caractéristique $(\Omega, \Omega^{pi}, \Omega^{ci})$, s'il existe une sous-caractéristique efficace pour la cryptanalyse différentielle ou linéaire, la statistique du χ^2 est capable de la détecter et de l'exploiter pour distinguer le bon candidat.

Par exemple, on peut essayer la caractéristique de Mitsuru Matsui avec la statistique S_{χ^2}. On a $p = 2^3$, $q = 2^5$ et $d \geq 2|\delta|\sqrt{p/q} = 1.19 \times 2^{-21}$. En utilisant $2^{45.7}$ messages clairs (6.5 fois plus que Mitsuru Matsui seulement), on obtient $\lambda = 3.26$. Dans ce cas, le χ^2 est approximativement normal, donc, en utilisant deux telles caractéristiques, on accélère la recherche exhaustive d'un facteur $\Phi(-3.26) = 2^{-10.8}$.

11.5 Analyse heuristique avec des projections

Les bits de calcul intermédiaires dans la fonction E_2 peuvent être ignorés arbitrairement et supposés uniformément distribués, indépendamment du reste du calcul. Avec une telle hypothèse heuristique, les boîtes obtiennent un comportement non déterministe, avec des tables indiquant les probabilités de transition. Il est possible de calculer ces matrices. Par exemple, si Ω^i et Ω^o sont des masques indiquant les bits restant autour d'une S-boîte respectivement en entrée et en sortie, on a la table des $S_{i,j} = Prob\,[S(x) \wedge \Omega^o = j / x \wedge \Omega^i = i]$. Pour des boîtes sur une même *couche* de E_2, on peut faire le produit tensoriel des matrices (également appelé produit de Kronecker), puis le produit matriciel des résultats. On obtient la matrice des $d_{i,j}$.

Par exemple, la fonction DES étant définie par un schéma de Horst Feistel, on a deux registres internes. En essayant tous les masques possibles de 4 bits sur le registre gauche et de 1 bit sur le registre droit, on obtient une matrice de transition de type $2^5 \times 2^5$. Celle obtenant la meilleur déviation se trouve être précisément celle obtenue par l'attaque de Mitsuru Matsui ! Cela a donc permis de retrouver la meilleur cryptanalyse linéaire de DES connue sans utiliser de propriétés de linéarité.

Conclusion

On a montré que les principales techniques utilisées dans la cryptanalyse différentielle et la cryptanalyse linéaire sont des cas particuliers de la cryptanalyse statistique. Il est donc possible de rassembler leurs efforts par un jeu de coefficients. Les deux méthodes précédentes peuvent être légèrement améliorées par l'information additionnelle de tous les coefficients $d_{i,j}$: la cryptanalyse différentielle par un test du χ^2, et la cryptanalyse linéaire par un test linéaire optimal. Ainsi, on a réalisé une économie de 5% dans le nombre de messages clairs connus

nécessaires dans la meilleur attaque de DES à ce jour.

Inversement, avec moins d'information (sans savoir quelles informations sur l'entrée et la sortie sont corrélées), la cryptanalyse par la méthode du χ^2 permet de faire sensiblement aussi bien, et donc de rechercher les attaques optimales par tâtonnement.

On a prouvé que l'aspect linéaire des S-boîtes qui semblait important dans les méthodes précédentes n'est pas indispensable, puisqu'il a été possible de retrouver une attaque équivalente à la meilleure attaque connue contre DES par de simples considérations statistiques.

Chapitre 12

Conclusion

La formalisation des primitives cryptographiques dans le cadre de la théorie des graphes permet de définir la sécurité offerte par le réseau de calcul utilisé. L'équation spécifiée par la fonctionnalité de la primitive s'écrit sous la forme d'un *graphe d'équation*. On définit la complexité de sa résolution comme étant la complexité en moyenne sur toutes les définitions possibles des boîtes de calcul, et cette valeur traduit la sécurité de la primitive.

Cette approche permet également de définir des classes génériques d'algorithmes pour résoudre les équations. Ainsi, la classe $\mathcal{A}^0(\bowtie, \gamma)$ généralise les notions de *recherche exhaustive* et de *paradoxe des anniversaires*. Comme il est d'usage de calculer de telles complexités de manière logarithmique par rapport au cardinal de l'alphabet utilisé (la *base*), pour une fonction de compression fournissant un résultat de longueur n, la complexité d'une recherche exhaustive est n. Une telle fonction sera donc dite sûre si la plus petite complexité d'un algorithme générique qui l'inverse est n, c'est-à-dire si l'on a $pC[\mathcal{A}^0(G, \bowtie, \gamma)] = n$ lorsque G modélise le problème de l'inversion de la fonction.

Des méthodes permettent de trouver des bornes inférieures de complexité vis-à-vis de la classe $\mathcal{A}^0(\bowtie, \gamma)$. Dans certains cas, une méthode pratique, inspirée de la théorie des graphes d'expansion, basée sur la seconde valeur propre du graphe de calcul, donne des bornes intéressantes. Si les boîtes ont autant d'entrées que de sorties (graphe *localement réversible*), si λ_2 est la seconde valeur propre du graphe G, le théorème 17 précise :

$$pC[\mathcal{A}^0(G, \bowtie, \gamma)] \geq \frac{2\lambda_2}{9} Card(V) - G(V)$$

où $G(V)$ est le nombre moyen (logarithmique) de solutions de l'équation. Dans le cas plus particulier des graphes symétriques, une méthode, inspirée des théories sur les problèmes algorithmiques des groupes finis, basée sur la distance moyenne du graphe de calcul, donne des bornes plus fines. Ainsi, si G est localement réversible, symétrique pour les arêtes, de degré d et de distance moyenne δ, le

corollaire 22 montre :

$$pC[\mathcal{A}^0(G, \bowtie, \gamma)] \geq \frac{d}{9\delta}\mathrm{Card}(V).$$

Des méthodes directement adaptées à des graphes particuliers peuvent être envisagées. Ainsi, pour étudier la sécurité de la famille de fonctions de compression construite sur le graphe de la transformée de Fourier rapide proposée dans [26], la méthode fournissant le meilleur résultat utilise les théorèmes usuels de la théorie des flots. Cette analyse montre qu'une double itération du graphe de la transformée de Fourier rapide est suffisant pour définir une fonction de compression sûre dans ce cadre.

Dans le cas général, des méthodes algorithmiques efficaces restent à trouver. L'intérêt de ce domaine de recherche est d'autant plus important que l'on arrive à attaquer des fonctions de hachage sur des considérations élémentaires sur le graphe de calcul. Ainsi, il est possible de casser la fonction FFT Hash II sur la simple notion de *contraction* de graphe.

Au niveau des boîtes de calcul, on a constaté l'absence de critère de sécurité sur la *diffusion*. Une mauvaise diffusion a été mise en évidence dans la fonction MD4, ce qui a permis de montrer que cette fonction de compression n'était pas *correlation-free*. Ceci a permis de trouver un critère combinatoire parfaitement adapté à la notion de "bonne diffusion" : les *multipermutations*. Celles-ci réalisent une parfaite diffusion et sont fortement liées aux codes MDS et à des objets qui intriguent les mathématiciens depuis deux siècles : les carrés latins. Il est cependant possible d'en construire de manière efficace pour être utilisées comme boîtes dans des primitives cryptographiques.

La *cryptanalyse linéaire* utilise les biais de linéarité introduits par les boîtes. Dans SAFER, on montre qu'une telle cryptanalyse est possible si l'on remplace une certaine boîte non biaisée par une autre. La propriété de non biais des boîtes de confusion, apparue ici comme accidentelle, aurait pu être fatale à la fonction de chiffrement si elle n'avait pas été présente. Cela aurait été dû à une simple mauvaise diffusion, une fois de plus.

De même, la *cryptanalyse différentielle* utilise une autre notion de biais, et l'on peut étudier la relation entre les biais maximum vis-à-vis des deux types d'attaque. Si l'on considère une boîte à p bits d'entrée et q bits de sortie, on peut définir le biais maximal Δ utilisé par la cryptanalyse différentielle, et le biais maximal Λ utilisé par la cryptanalyse linéaire. Une boîte sera d'autant plus sûre face à ces attaques que Δ et Λ seront faibles. On a les bornes inférieures suivantes :

$$\begin{aligned}
\Delta &\geq \max\left(2^{p-q}, 2\right) \\
\Lambda &\geq \max\left(\frac{1}{2}2^{p/2}, \frac{1}{2}\left(3 \times 2^p - 2 - 2\frac{(2^p-1)(2^{p-1}-1)}{2^q-1}\right)^{1/2}\right).
\end{aligned}$$

Chaque cas d'égalité correspond à un objet combinatoire particulier. Les cas $\Delta = 2^{p-q}$ et $\Lambda = \frac{1}{2}2^{p/2}$ sont équivalents (théorème 52) et correspondent aux fonctions *courbes*, possible si, et seulement si $p \geq 2q$ et p pair (théorème 53). Le cas $\Delta = 2$ correspond aux fonctions *presque parfaitement non linéaires*. Le cas d'égalité avec la (nouvelle) borne

$$\Lambda = \frac{1}{2}\left(3 \times 2^p - 2 - 2\frac{(2^p-1)(2^{p-1}-1)}{2^q-1}\right)^{1/2}$$

correspond aux fonctions *presque courbes*, possible si, et seulement si $p = q$ et p impair lorsqu'il est disjoint du cas des fonctions courbes (théorème 58). Une fonction presque courbe est presque parfaitement non linéaire, sans que la réciproque soit vraie (théorème 55).

La non linéarité des boîtes rend les primitives sûres contre les méthodes heuristiques d'analyse proposées par Eli Biham, Adi Shamir et Mitsuru Matsui. Cela n'est bien sûr pas suffisant, car les non linéarités locales peuvent très bien s'annuler par un phénomène global. On a montré une approche globale de ces méthodes dans un contexte statistique et comment retrouver la meilleure attaque connue de DES sans aucun aspect de linéarité, et comment l'améliorer sensiblement.

Ces quelques résultats forment peut-être un premier pas vers une théorie générale sur les primitives cryptographiques. Certains manques ont été comblés, et de nouvelles bases permettent d'envisager une étude plus systématique. Il est intéressant de constater que l'on peut définir une structure mathématique régulière là où il n'y en avait *a priori* pas, où régnait la loi de l'empirisme, et où toute étude était résolument vouée à l'isolement dans le cadre restrictif d'une fonction particulière. Il est fort probable que le problème général de la sécurité des primitives cryptographiques cache des problèmes ouverts de la théorie de la complexité tels le célèbre problème P *vs* NP. En découvrant ces liens, il serait alors possible de construire des primitives dont l'existence d'une attaque serait équivalente à un tel problème.

Bibliographie

Généralités sur la cryptologie

[1] G. Brassard. *Cryptologie contemporaine*, Masson, 1992.

[2] D. E. Robling-Denning. *Cryptography and data security.* Addison-Wesley, 1982.

[3] W. Diffie, M. E. Hellman. New directions in cryptography. In *IEEE Transactions on Information Theory*, vol. IT-22, pp. 644–654, 1976.

[4] D. Kahn. *The codebreakers*, Macmillan, 1967. Traduction française : *la guerre des codes*, InterEdition, 1980.

[5] H. van Tilborg. *An introduction to cryptology*, Kluwer Academic Publishers, 1988.

Fonctions de chiffrement — propositions

[6] U. S. National Bureau of Standards. Data Encryption Standard. Federal Information Processing Standard Publication 46, 1977.

[7] L. Brown, J. Pieprzyk, J. Seberry. LOKI: a cryptographic primitive for authentication and secrecy applications. In *Advances in Cryptology AUSCRYPT'90*, Sydney, Australie, Lectures Notes in Computer Science 453, pp. 229–236, Springer-Verlag, 1990.

[8] T. W. Cusick, M. C. Wood. The REDOC-2 cryptosystem. In *Advances in Cryptology CRYPTO'90*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 537, pp. 545–563, Springer-Verlag, 1991.

[9] H. Feistel. Cryptography and computer privacy. In *Scientific american*, vol. 228, pp. 15–23, 1973.

[10] X. Lai, J. L. Massey. A proposal for a new block encryption standard. In *Advances in Cryptology EUROCRYPT'90*, Aarhus, Danemark, Lectures Notes in Computer Science 473, pp. 389–404, Springer-Verlag, 1991.

[11] J. L. Massey. SAFER K-64: a byte-oriented block-ciphering algorithm. In *Fast Software Encryption – Proceedings of the Cambridge Security Workshop*, Cambridge, Royaume Uni, Lectures Notes in Computer Science 809, pp. 1–17, Springer-Verlag, 1994.

[12] R. C. Merkle. Fast software encryption functions. In *Advances in Cryptology CRYPTO'90*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 537, pp. 476–501, Springer-Verlag, 1991.

[13] S. Miyaguchi. The FEAL cipher family. In *Advances in Cryptology CRYPTO'90*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 537, pp. 627–638, Springer-Verlag, 1991.

[14] R. L. Rivest, A. Shamir, L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystem. In *Communications of the ACM*, vol. 21, pp. 120–126, 1978.

[15] A. Shimizu, S. Miyaguchi. Fast data Encryption Algorithm. In *Advances in Cryptology EUROCRYPT'87*, Amsterdam, Hollande, Lectures Notes in Computer Science 304, pp. 267–278, Springer-Verlag, 1988.

[16] G. S. Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications. In *Journal of the American Institute of Electrical Engineers*, vol. 45, pp. 109–115, 1926.

Fonctions de chiffrement — théorie

[17] X. Lai. *On the design and security of block ciphers*, ETH Series in Information Processing 1, Hartung-Gorre Verlag Konstanz, 1992.

[18] C. E. Shannon. Communication theory of secrecy systems. In *Bell system technical journal*, vol. 28, pp. 656–715, 1949.

Fonctions de hachage — propositions

[19] U. S. Department of Commerce, National Institute of Standards and Technology. Secure Hash Standard. Federal Information Processing Standard Publication 180, 1993.

[20] I. B. Damgård. A design principle for hash functions. In *Advances in Cryptology CRYPTO'89*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 435, pp. 416–427, Springer-Verlag, 1990.

[21] D. W. Davies, W. L. Price. Digital signature – an update. In *Proceedings of the International Conference on Computer Communications*, Sydney, pp. 843-847, North-Holland, 1985.

[22] R. C. Merkle. One way hash functions and DES. In *Advances in Cryptology CRYPTO'89*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 435, pp. 416–427, Springer-Verlag, 1990.

[23] S. M. Matyas, C. H. Meyer, J. Oseas. Generating strong one-way functions with cryptographic algorithm. *IBM Technical Disclosure Bulletin*, vol. 27, pp. 5658–5659, 1985.

[24] C. P. Schnorr. FFT-Hashing: an efficient cryptographic hash function. Présenté à CRYPTO'91. Non publié.

[25] C. P. Schnorr. FFT-Hash II, efficient cryptographic hashing. In *Advances in Cryptology EUROCRYPT'92*, Balatonfüred, Hongrie, Lectures Notes in Computer Science 658, pp. 45–54, Springer-Verlag, 1993.

[26] C. P. Schnorr, S. Vaudenay. Parallel FFT-Hashing. In *Fast Software Encryption – Proceedings of the Cambridge Security Workshop*, Cambridge, Royaume Uni, Lectures Notes in Computer Science 809, pp. 149–156, Springer-Verlag, 1994.

[27] R. L. Rivest. The MD4 message-digest algorithm. In *Advances in Cryptology CRYPTO'90*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 537, pp. 303–311, Springer-Verlag, 1991.

[28] R. L. Rivest. The MD5 message-digest algorithm. Présenté à CRYPTO'91. Non publié.

Fonctions de hachage — théorie

[29] R. J. Anderson. The Classification of Hash Functions. In *Proceedings of the 4th IMA Conference on Cryptography and Coding*, Cirencester, Royaume Uni, pp. 83–95, Oxford University Press, 1995.

[30] I. B. Damgård. Collision free hash functions and public key signature schemes. In *Advances in Cryptology EUROCRYPT'87*, Amsterdam, Hollande, Lectures Notes in Computer Science 304, pp. 203–216, Springer-Verlag, 1988.

[31] B. Preneel. Ph.D. thesis *analysis and design of cryptographic hash functions*, Katholieke Universiteit Leuven, Departement Elektrotechniek, 1993.

Générateurs pseudo-aléatoires

[32] W. Alexi, B. Chor, O. Gldreich, C. P. Schnorr. RSA and Rabin functions: certain parts are as hard as the whole. In *25th Symposium on Foundations of Computer Science*, Singer Island, Floride, U.S.A., pp. 449–457, IEEE Computer Society Press, 1984

[33] L. Blum, M. Blum, M. Shub. A simple unpredictable pseudo-random number generator. In *SIAM journal on computing*, vol. 15, pp. 364–383, 1986.

[34] M. Blum, S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. In *SIAM journal on computing*, vol. 13, pp. 850–864, 1984.

[35] S. Micali, C. P. Schnorr. Efficient, perfect random number generators. In *Advances in Cryptology CRYPTO'88*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 403, pp. 173–198, Springer-Verlag, 1990.

Cryptanalyse dédiée

[36] B. den Boer, A. Bosselaers. An attack on the last two rounds of MD4. In *Advances in Cryptology CRYPTO'91*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 576, pp. 194–203, Springer-Verlag, 1992.

[37] B. den Boer, A. Bosselaers. Collisions for the compression function of MD5. In *Advances in Cryptology EUROCRYPT'93*, Lofthus, Norvège, Lectures Notes in Computer Science 765, pp. 293–304, Springer-Verlag, 1994.

[38] T. Baritaud, H. Gilbert, M. Girault. FFT Hashing is not collision-free. In *Advances in Cryptology EUROCRYPT'92*, Balatonfüred, Hongrie, Lectures Notes in Computer Science 658, pp. 35–44, Springer-Verlag, 1993.

[39] W. Diffie, M. E. Hellman. Exhaustive Cryptanalysis of the NBS Data Encryption Standard. In *Computer*, vol. 10, pp. 74–84, 1977.

[40] M. E. Hellman. A Cryptanalytic Time-Memory Tradeoff. In *IEEE Transactions on Information Theory*, vol. IT-26, pp. 401–406, 1980.

[41] M. E. Hellman, R. C. Merkle, R. Schroppel, L. Washington, W. Diffie, S. Pohlig, P. Schweitzer. *Results of an Initial Attempt to Cryptanalyse the NBS Date Encryption Standard*, Stanford University, 1976.

[42] S. Vaudenay. FFT-Hash II is not yet collision-free. In *Advances in Cryptology CRYPTO'92*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 740, pp. 587–593, Springer-Verlag, 1993.

[43] S. Vaudenay. On the need for multipermutations: cryptanalysis of MD4 and SAFER. A paraître dans Fast Software Encryption, Lectures Notes in Computer Science.

Cryptanalyse différentielle

[44] E. Biham, A. Shamir. Differential cryptanalysis of DES-like cryptosystems. In *Advances in Cryptology CRYPTO'90*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 537, pp. 2–21, Springer-Verlag, 1991. In *Journal of Cryptology*, vol. 4, pp. 3–72, 1991.

[45] E. Biham, A. Shamir. Differential cryptanalysis of FEAL and N-Hash. In *Advances in Cryptology EUROCRYPT'91*, Brighton, Royaume Uni, Lectures Notes in Computer Science 547, pp. 1–16, Springer-Verlag, 1991.

[46] E. Biham, A. Shamir. Differential cryptanalysis of Snefru, Khafre, REDOC-2, LOKI. In *Advances in Cryptology CRYPTO'91*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 576, pp. 156–171, Springer-Verlag, 1992.

[47] E. Biham, A. Shamir. Differential cryptanalysis of the full 16-round DES. In *Advances in Cryptology CRYPTO'92*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 740, pp. 487–496, Springer-Verlag, 1993.

[48] E. Biham, A. Shamir. *Differential cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.

[49] D. Coppersmith. The Data Encryption Standard (DES) and its strength against attacks. Rapport technique RC 18613 (81421) du 22 décembre 1992, IBM Yorktown.

[50] X. Lai, J. L. Massey, S. Murphy. Markov ciphers and differential cryptanalysis. In *Advances in Cryptology EUROCRYPT'91*, Brighton, Royaume Uni, Lectures Notes in Computer Science 547, pp. 17–38, Springer-Verlag, 1991.

Cryptanalyse linéaire

[51] B. R. Kaliski Jr., M. J. B. Robshaw. Linear cryptanalysis using multiple approximations. In *Advances in Cryptology CRYPTO'94*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 839, pp. 26–39, Springer-Verlag, 1994.

[52] M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology EUROCRYPT'93*, Lofthus, Norvège, Lectures Notes in Computer Science 765, pp. 386–397, Springer-Verlag, 1994.

[53] M. Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In *Advances in Cryptology CRYPTO'94*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 839, pp. 1–11, Springer-Verlag, 1994.

[54] A. Tardy-Corfdir, H. Gilbert. A known plaintext attack of FEAL-4 and FEAL-6. In *Advances in Cryptology CRYPTO'91*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 576, pp. 172–182, Springer-Verlag, 1992.

Etude des boîtes

[55] C. Carlet. *Fonctions Booléennes en théorie des codes correcteurs d'erreurs et en cryptologie*, Université de Picardie, 1994.

[56] F. Chabaud, S. Vaudenay. Links between differential and linear cryptanalysis. A paraître dans *Advances in Cryptology EUROCRYPT'94*, Perugia, Italie, Lectures Notes in Computer Science.

[57] G. Lachaud, J. Wolfmann. The weights of the orthogonals of the extended quadratic binary Goppa codes. In *IEEE Transactions on Information Theory*, vol. IT-36, pp. 686–692, 1990.

[58] W. Meier, O. Staffelbach. Nonlinearity criteria for cryptographic functions. In *Advances in Cryptology EUROCRYPT'89*, Houthalen, Belgique, Lectures Notes in Computer Science 434, pp. 549–562, Springer-Verlag, 1990.

[59] K. Nyberg. Perfect nonlinear S-boxes. In *Advances in Cryptology EUROCRYPT'91*, Brighton, Royaume Uni, Lectures Notes in Computer Science 547, pp. 378–385, Springer-Verlag, 1991.

[60] K. Nyberg. Differentially uniform mappings for cryptography. In *Advances in Cryptology EUROCRYPT'93*, Lofthus, Norvège, Lectures Notes in Computer Science 765, pp. 55–64, Springer-Verlag, 1994.

[61] K. Nyberg, L. R. Knudsen. Provable security against differential cryptanalysis. In *Advances in Cryptology CRYPTO'92*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 740, pp. 566–574, Springer-Verlag, 1993.

[62] L. O'Connor. Properties of linear approximation tables. A paraître dans Fast Software Encryption, Lectures Notes in Computer Science.

[63] T. Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. In *IEEE transactions on computers*, vol. C-34, pp. 81–85, 1985.

[64] A. F. Webster, S. E. Tavares. On the design of S-boxes. In *Advances in Cryptology CRYPTO'85*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 218, pp. 523–534, Springer-Verlag, 1986.

[65] G. Xiao, J. L. Massey. A spectral characterization of correlation-immune combining functions. In *IEEE Transactions on Information Theory*, vol. IT-34, pp. 569–571, 1988.

Attaques de type "anniversaire"

[66] D. Coppersmith. Another birthday attack. In *Advances in Cryptology CRYPTO'85*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 218, pp. 14–17, Springer-Verlag, 1986.

[67] P. Flajolet, A. M. Odlyzko. Random mapping statistics. In *Advances in Cryptology EUROCRYPT'89*, Houthalen, Belgique, Lectures Notes in Computer Science 434, pp. 329–354, Springer-Verlag, 1990.

[68] M. Girault, R. Cohen, M. Campana. A generalized birthday attack. In *Advances in Cryptology EUROCRYPT'88*, Davos, Suisse, Lectures Notes in Computer Science 330, pp. 128–156, Springer-Verlag, 1988.

[69] P. C. van Oorschot, M. J. Wiener. Parallel collision search with application to hash functions and discrete logarithms. In *2nd ACM conference on computer and communications security*, Fairfax, Virginie, U.S.A., pp. 210–218, ACM press, 1994.

[70] J. M. Pollard. A Monte Carlo method for factorization. In *Bit*, vol. 15, pp. 331–334, 1975.

[71] J.-J. Quisquater, J.-P. Delescaille. How easy is collision search? Application to DES. In *Advances in Cryptology EUROCRYPT'89*, Houthalen, Belgique, Lectures Notes in Computer Science 434, pp. 429–434, Springer-Verlag, 1990.

Autres cryptanalyses

[72] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *35th Symposium on Foundations of Computer Science*, Santa Fe, Nouveau Mexique, U.S.A., pp. 124–134, IEEE Computer Society Press, 1994

[73] C. P. Schnorr, S. Vaudenay. Black box cryptanalysis of hash networks based on multipermutations. A paraître dans *Advances in Cryptology EUROCRYPT'94*, Perugia, Italie, Lectures Notes in Computer Science.

Combinatoire

[74] N. Alon. Eigenvalues and expanders. In *Combinatorica*, vol. 6, pp. 83–96, 1986.

[75] N. Alon, V. D. Milman. λ_1, isoperimetric inequalities for graphs, and superconcentrators. In *Journal of combinatorial theory*, vol. B 38, pp. 73–88, 1985.

[76] N. Biggs. In *Algebraic graph theory*, Cambridge university press, 1974.

[77] L. Babai, M. Szegedy. Local expansion of symmetrical graphs. In *Combinatorics, probability and computing*, vol. 1, pp. 1–11, 1992.

[78] U. M. Maurer, J. L. Massey. Local randomness in pseudorandom sequences. In *Journal of Cryptology*, vol. 4, pp. 135–149, 1991.

[79] O. S. Rothaus. On bent functions. In *Journal of combinatorial theory (A)*, vol. 20, pp. 300–305, 1976.

[80] R. M. Tanner. Explicit concentrators from generalized N-gons. In *SIAM journal of algebraic discrete methods*, vol. 5, pp. 287–293, 1984.

Carrés latins

[81] J. Dénes, A. D. Keedwell. *Latin squares and their applications*, Akadémiai Kiadó, Budapest, 1974.

[82] M. Hall, L. J. Paige. Complete mappings of finite groups. In *Pacific Journal of Mathematics*, vol. 5, pp. 541–549, 1955.

Théorie

[83] H. Cramer. *Mathematical Methods of Statistics*, Princeton University Press, 1946.

[84] W. Feller. *An Introduction to Probability Theory and its Applications*, vol. 1, Wiley, 1957.

[85] O. Goldreich. A uniform-complexity treatment of encryption and zero-knowledge. In *Journal of Cryptology*, vol. 6, pp. 21–53, 1993.

[86] M. R. Garey, D. S. Jonhson. *Computers and intractability : a guide to the theory of NP-completeness.* W. H. Freeman and Company, 1979.

[87] O. Goldreich, L. A. Levin. Hard-core predicate for any one-way function. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, Seattle, Washington, U. S. A., pp. 25–32, ACM Press, 1989

[88] S. Goldwasser, S. Micali. Probabilistic encryption. In*Journal of computer system science*, vol. 28, pp. 270–299, 1984.

[89] F. J. McWilliams, N. J. A. Sloane. *The theory of error-correcting codes*, North-Holland, 1977.

Autre

[90] S. Vaudenay. One-Time Identification with Low Memory. In *Proc. EUROCODE'92*, Udine (Italie) pp 217–228. CISM Courses and lectures No. 339, Springer-Verlag, 1993.

[91] D. Coppersmith, J. Stern, S. Vaudenay. Attacks on the Birational Permutation Signature. In *Advances in Cryptology CRYPTO'93*, Santa Barbara, Californie, U.S.A., Lectures Notes in Computer Science 773, pp. 435–443, Springer-Verlag, 1994.

[92] D. Naccache, D. M'Raïhi, D. Raphaeli, S. Vaudenay. Complexity tradeoffs with the Digital Signature Standard. A paraître dans *Advances in Cryptology EUROCRYPT'94*, Perugia, Italie, Lectures Notes in Computer Science.

Glossaire

attaque. Tentative de casser.

attaque à texte chiffré connu. Type d'attaque d'une fonction de chiffrement dans lequel on suppose l'accès à une liste de messages chiffrés.

attaque à texte clair connu. Type d'attaque d'une fonction de chiffrement dans lequel on suppose l'accès à une liste de messages clairs et de leur message chiffré correspondant.

attaque à texte clair choisi. Type d'attaque d'une fonction de chiffrement dans lequel on suppose l'accès à une boîte noire qui peut chiffrer tout message clair choisi.

boîte de calcul. Dispositif qui transforme une ou plusieurs informations.

bruit. Phénomène qui altère un message de manière aléatoire.

casser. Action de montrer qu'une spécification de sécurité n'est pas satisfaite dans certains cas.

chiffrement. Action de transformer un message clair sous une forme chiffrée inintelligible, sauf à un ensemble de personnes possédant un secret choisi. Il existe une fonction de déchiffrement liée au secret qui permet de rétablir le message clair.

chiffrement symétrique. Type de chiffrement dans lequel la même clef permet de chiffrer et de déchiffrer.

codage. Action de transformer un message sous une forme résistante au bruit. Il existe une fonction de déchifrement telle que deux codes légèrement différents se décodent en le même message.

collision. Deux valeurs différentes dont les images par une certaine fonction sont identiques.

complexité. Quantité de temps ou d'espace nécessaire.

compression. Action de réduire un morceau de message et une petite information en une autre petite information.

confidentialité. Propriété de rendre une information inaccessible sauf pour des personnes concernées.

confusion. Action de transformer une information élémentaire.

crypto. *voir* rigolo

cryptanalyse. Art de la résolution des problèmes cryptographiques.

cryptographie. Science des procédés de chiffrement. Par extension, science des procédés permettant d'assurer la sécurité des systèmes de traitement de l'information.

cryptologie. Science étudiant la cryptanalyse et la cryptographie.

décodage. *voir* codage.

déchiffrement. *voir* chiffrement.

décryptage. Action consistant à retrouver un message clair sans posséder le secret du déchiffrement.

DES. Fonction de chiffrement standardisée par le gouvernement américain en 1977.

difficile. De complexité élevée.

diffusion. Action de rassembler plusieurs informations élémentaires en une seule.

diversification de clef. Procédé de calcul de sous-clefs à partir d'une clef dans une fonction cryptographique.

factorisation. Action de décomposer un nombre entier naturel en produit de plusieurs autres différents de 1.

FFT Hash II. Fonction de hachage développée par Claus Schnorr.

hachage. Action de réduire un message en une petite information.

IDEA. Fonction de chiffrement développée par Xuejia Lai, James Massey et Sean Murphy.

intégrité. Propriété de prévenir un message contre son altération par un procédé malicieux.

Lucifer. Fonction de chiffrement développée par Horst Feistel.

MD4. Fonction de hachage développée par Ronald Rivest.

message chiffré. *voir* chiffrement.

message clair. *voir* chiffrement.

non linéarité. Propriété d'une fonction de ne pas ressembler à une application linéaire.

paradoxe des anniversaire. Fait que la probabilité d'obtenir deux personnes ayant la même date d'anniversaire dans une assemblé est beaucoup plus élevé que celle d'avoir une personne dont la date d'anniversaire est une date donnée.

paramètre de sécurité. Paramètre influançant directement la complexité de toute attaque.

pince. Deux valeurs dont les images par deux fonctions sont égales.

primitive cryptographique. Procédé cryptographique élémentaire.

recherche exhaustive d'une solution. Action d'essayer toute solution candidate potentielle.

réseau de calcul. Graphe composé de sommets représentant des boîtes de calcul et liés par des arêtes orientées qui représentent des données.

réversible. Propriété d'une fonction d'avoir des espaces de départ et d'arrivée de même taille.

rigolo. *voir* crypto

RSA. Fonction de chiffrement asymétrique développée par Ronald Rivest, Adi Shamir et Leonard Adleman.

SAFER. Fonction de chiffrement développée par James Massey.

sans collision. Propriété d'une fonction d'être telle qu'il est difficile de produire une collision.

SHA. Fonction de hachage standardisée par le gouvernement américain.

sens unique. Propriété d'une fonction d'être telle qu'il est en moyenne difficile de trouver une préimage d'une valeur donnée.

sous-clef. Information déduite d'une clef utilisée dans un tour.

spécification de sécurité. Propriété imposée par la fonctionnalité pour la sécurité.

tour. Etape de calcul dans une fonction cryptographique.

Index

Table des figures

Table des matières

Dans les années cinquante, Claude Shannon initia la théorie des primitives cryptographiques. Il définit les notions de diffusion et de confusion. Toutefois, cette théorie a très peu évolué jusqu'à nos jours. Récemment, la cryptanalyse différentielle et la cryptanalyse linéaire marquèrent un progrès significatif dans l'analyse des primitives. Des critères de sécurité sur les boîtes de confusion, essentiellement des critères de non-linéarité, furent proposés.

Dans cette thèse, on montre comment construire une notion de complexité sur la structure de graphe des primitives et comment l'étudier. Ceci fournit des critères de sécurité sur le réseau de calcul. On propose également de nouveaux critères sur la diffusion. On unifie enfin les deux types de cryptanalyse en les débarrassant de leur aspect linéaire par une approche statistique.

In the early fifties, Claude Shannon initiated the theory of cryptographic primitives. He defined the notion of diffusion and confusion. However, this theory did not develop very much until now. Recently, the differential cryptanalysis and the linear cryptanalysis gave a significant advance in the analysis of the primitives. Security criteria for confusion, essentially nonlinearity criteria, has been proposed.

In this thesis, we show how to define a notion of complexity on the graph structure of the primitives and how to study it. This gives security criteria of the computational network. We propose new criteria for diffusion. Finally, we unify the two types of cryptanalysis, by getting rid of their linear aspects by a statistical approach.