Cryptanalyses statistiques des algorithmes de chiffrement à clef secrète.

Benoît Gérard

Thèse de doctorat de l'université Pierre et Marie Curie

Spécialité Informatique

EDITE de Paris

présentée par

Benoît GÉRARD

pour obtenir le grade de

Docteur de l'université Pierre et Marie Curie

Cryptanalyses statistiques des algorithmes de chiffrement à clef secrète.

soutenue le 9 décembre 2010

devant le jury composé de

Directeur de thèse	
Jean-Pierre Tillich	INRIA Paris-Rocquencourt, France
Rapporteurs	
Pascal junod	HEIG-VD - IICT, Suisse
François-Xavier standaert	Université Catholique de Louvain, Belgique
Examinateurs	
Anne canteaut	INRIA Paris-Rocquencourt, France
Antoine joux	Université de Versailles Saint-Quentin - DGA, France
Kaisa nyberg	Helsinki University of Technology, Finlande
Matt robshaw	Orange Labs, France
Michèle soria	Université Pierre et Marie Curie, France
Invité	
Jacques blanc-talon	Direction Générale de l'Armement, France

Benoît Gérard

Cryptanalyses statistiques des algorithmes de chiffrement à clef secrète.

INRIA-équipe SECRET
Domaine de Voluceau
78153 Le Chesnay

Remerciements

Pour avoir maintes fois relu les remerciements des camarades qui m'ont précédé, je suis conscient que le risque d'omettre une personne, même si elle a beaucoup compté, est loin d'être négligeable étant données les circonstances dans lesquelles on se retrouve quand on écrit ces lignes. Je vais tenter, néanmoins, de penser à tout le monde.

Bien évidemment je remercie tout d'abord Jean-Pierre Tillich qui m'a proposé le paquet *stage+thèse* que je cherchais sur une thématique que j'ai aimé traiter. Je le remercie aussi pour sa patience et les nombreux *groumpfs*[1] restés dans sa gorge et bien évidemment les conseils prodigués tout au long de ces, presque, quatre années.

Je remercie aussi Pascal Junod et François-Xavier Standaert pour avoir accepté d'être rapporteurs de cette thèse et tiens, à nouveau, à m'excuser pour le peu de temps qu'il leur a été donné pour la lecture de ce document. J'en profite pour remercier François-Xavier de m'accueillir au sein de son équipe pour la suite des événements.

Un grand merci aussi aux autres membres du jury. Merci à Kaisa Nyberg d'avoir fait le déplacement depuis Helsinki, merci à Matt Robshaw d'avoir trouvé le temps de venir ainsi qu'à Anne Canteaut, Antoine Joux et Michèle Soria qui ont accepté de participer à ce jury malgré leurs nombreuses autres tâches du moment (jurys, comité de programme, ...).

Je tiens aussi à témoigner ma gratitude envers le projet CODES devenu, en cours de route l'équipe-projet INRIA SECRET et en particulier ses membres permanents Pascale, Anne, Nicolas, Daniel (à une époque), Jean-Pierre pour leur accueil et leurs précieux conseils. *Mersi bras* Christelle de t'être occupée de nous a bientôt peut être en pays breton ! Je n'oublie pas les thésards, post-doctorants, stagiaires et visiteurs de passage qui ont contribué à la bonne ambiance au sein du projet Ayca, Alexander, Andrea, Anne, Ayoub, Bassem, Bhaskar, Cédric F., Cédric L., Céline, Chrysanthi, Christina, Christophe, Deepak, Denise, Frédéric, Grégory, Jean-Christophe, Mamdouh, Maria, Mathieu, Maxime, Rafael, Richi, Rima, Stéphane J., Stéphane M., Sumanta, Sunandan Valérie, Vincent, et Yann.

Je remercie aussi Matthieu et Françoise pour leur gentillesse, Matthieu pour ses nombreux coups de mains (entre autre pour les locaux de la soutenance) et Françoise pour les trois années passées à effectuer les petites classes du MA203.

Merci aussi à Julien, Manel et les autres chargés de TP du module d'informatique générale à Polytech' Paris.

J'ai aussi une petite pensée pour les personnes extérieures au projet que j'ai eu l'occasion de côtoyer à Rocquencourt ou en conférence Cédric T., Gaëtan, Léonard, Ludovic, Marion (avec qui j'ai eu le plaisir de déguster le dîner de conférence le plus raffiné à ce jour), Morgan ainsi que les membres des équipes de Paris 8, Limoges et de Rennes.

Enfin, je remercie Baudoin, Jonathan, Miia et Joo Yeon pour les réponses apportées à

1. Je ne suis pas sûr de l'orthographe.

mes diverses questions.

Quelques ≪ dédicaces spéciales ≫. Je souhaiterais à présent effectuer quelques remerciements plus spécifiques.

- En tout premier lieu, un grand merci à Céline pour les nombreuses heures passées ensemble à préparer une présentation, un poster, un article, un résumé, à calculer un développement limité ou à programmer une attaque.
- Un grand merci à Alexander pour avoir accepté de loger chez moi pour certains de ses voyages en France. Merci de m'avoir fait découvrir les *spätzel mit linsen* et par la même occasion de m'avoir fourni de quoi manger pour le mois suivant.
- Merci à Vincent d'avoir passé son temps à me poser des questions relevant de mon *in*-compétence, à faire planter les machines et à dénigrer une équipe lorientaise sans qui, pourtant, Lyon ne serait pas en ligue des champions cette année.
- Je profite de cet intermède sportif pour regretter l'absence de Mathieu, Cédric et Yann pour l'année où leurs clubs respectifs atteignaient enfin la première place du championnat.
- Merci aux cruciverbistes en herbes ou non pour les quelques petites minutes passées à plancher sur ces grilles. En particulier merci à Jean-Pierre et Ayoub d'avoir fait l'effort d'accepter que bien que capilotractées, les définitions avaient quand même un sens.
- Un petit mot aussi pour le schtroumpf pisciphobe et notre inénarrable partie de sauna ping-pong en Norvège.
- Merci à ma mère qui depuis ma plus tendre enfance m'a toujours dit ≪ de ne pas en faire une thèse ≫. J'espère que tu es fière de toi maintenant.
- Plus généralement merci à toute ma famille ~~sans qui je ne serais pas là~~.
- Merci à ma belle famille d'avoir accepté un gendre avec une situation professionnelle si peu stable.
- Enfin, merci à Audrey de m'avoir supporté, de m'avoir ~~fait perdre un temps fou à préparer un mariage alors que j'avais autre chose à faire~~ fait vivre ce moment exceptionnel le 24 avril dernier. Je la remercie aussi pour *ces heures perdues alors que mon manuscrit était loin d'être finalisé* son soutien lors de ces dernières semaines tendues à rédiger la thèse et préparer la soutenance.

Overview

Chapter 1 to 5

The five first chapters of the document are dedicated to the introduction of the notion handled when considering statistical cryptanalyses. First, in Chapter 1, we give a quick overview of cryptology and stress on iterative block ciphers that are the cryptographic primitives that we consider in this document. Then, in order to illustrate the notion of iterative block ciphers, Chapter 2 provides the specifications of two ciphers namely the *Data Encryption Standard* (DES) and PRESENT. Then, before going more deeply into the topic of statistical cryptanalysis, we recall, in Chapter 3, some basics of coding theory that may be useful for the understanding of the document. Finally, in Chapters 4 and 5, we present the well-known linear and differential cryptanalyses. For both cryptanalyses, we present the original attacks together with the extensions that have been proposed up to now. Discussions on the classical hypotheses made for analysing these attacks are made and two refinements are proposed for two particular analyses. First, the analysis of entangled linear cryptanalysis is performed without using the Gaussian approximation of the counters what leads to a formula for its success probability that depends on the size M of the available samples. This dependency was hidden by the Gaussian approximation in the initial analysis proposed by the authors. Secondly, a formula for the success probability of a differential cryptanalysis is proposed. This one takes into account the fixed-key dependency of the differential probability and can be found in [BG10].

Chapter 6

The aim of this chapter is to provide a framework for analysing simple statistical cryptanalyses that is not depending on the kind of cryptanalysis considered. In the case of linear cryptanalysis, the complexities of an attack are computed using a Gaussian approximation of the binomial distribution. On the other hand, some analyses of differential cryptanalysis are made using a Poisson approximation that is better-suited in that case. Nevertheless, it is shown, in Sections 6.2 and 6.3, that both approximations are not well-suited at the same time. Moreover, in the case of truncated differential cryptanalysis, it is shown that, depending on the setting of the attack, both Gaussian and Poisson approximations may be non-valid at the same time. This is the reason why it is important to get formulas for the complexities of a statistical cryptanalysis that are obtained without resorting to a particular approximation.

In Section 6.3, we provide an estimate of the tails of the binomial distribution that is valid for any set of parameters. The main drawback is that this approximation is quite more difficult to manipulate than the Gaussian or the Poisson approximation. Indeed, the tail distribution

is expressed using the Kullback-Leibler divergence between two Benoulli distributions :

$$D(p||q) \stackrel{\text{def}}{=} p \ln\left(\frac{p}{q}\right) + (1-p)\ln\left(\frac{1-p}{1-q}\right).$$

Nevertheless, in Section 6.4, we obtain an estimate for the data complexity of a statistical cryptanalysis with success probability close to 0.5 that keeps ℓ over n key candidates :

$$N'' \stackrel{\text{def}}{=} -\frac{\log(2\sqrt{\pi}\ell/n)}{D(p_*||p)}.$$

The probability p_* (*resp.* p) is the probability of a counter corresponding to a correct key (*resp.* non-correct key) to be incremented by a sample. This work has been published in [BG09a, BG09b].

Then, in Section 6.5, we focus on estimating the success probability of a statistical cryptanalysis. A formula similar to the one proposed by Selçuk [Sel08] is derived in the general case. This formula involves the probability of the correct counter to be equal to i that, we denote by $f_*(i)$, and the inverse cumulative function of the wrong counters : F^{-1}. The formula obtained is

$$P_S \approx \sum_{F^{-1}\left(1-\frac{\ell-1}{n-2}\right)}^{N} f_*(i).$$

This work also suggests that taking a number of samples N of the form

$$N = -c \cdot \frac{\log(2\sqrt{\pi}\ell/n)}{D(p_*||p)},$$

the success complexity essentially depends on the value of c and is close to 1 for $c \geq 2$. These results are to be published in [BGT10].

Chapter 7

The problematic of Chapter 7 is the one of recovering the key in a multiple linear cryptanalysis of type 1. In Section 7.1 we explain that this can be viewed as a problem of decoding a linear code over a Gaussian channel. Then, in Section 7.2 we propose to use the *stochastic resonance algorithm* by Valembois to solve it. This section is an overview of the results of [Gér07]. Experiments have been done on a slightly modified version of the algorithm to show that using such a method, the number of candidates considered is reduced by a non negligible factor with a small loss in the success probability of the attack.

Chapter 8

When performing a statistical cryptanalysis with a reasonable number of samples, it is very unlikely that the correct key is not ranked among the best candidates. Nevertheless, while being very small, the probability of this event is not zero and thus influences the expected value of the rank of the correct key. The point is that running hundreds or thousands attacks, this phenomenon will not appear and hence, the mean of the ranks experimentally obtained will be smaller than the expected value. In Chapter 8, we propose another way of compute

the efficiency of a multiple linear cryptanalysis. Instead of computing the expected value of the rank of the correct key, the approach is to compute a rank for which most of the attacks will be successful.

The tool used here is the conditional entropy of the key knowing the statistic extracted from the samples. This quantity is less influenced by large deviations of the rank than the expected rank. Hence, it seems that entropy is well-suited for this problem. In Section 8.1, we provide formulas for the three types of multiple linear cryptanalyses and, moreover, apply this approach to the multidimensional attack proposed in [HCN09a]. The results obtained are matching empirical success rates. Moreover, in Section 8.2, we apply this estimate to a multiple linear cryptanalysis on 16-round DES and compare it to the results of 19 attacks we performed. Once again, the estimate matches the experiments what emphasises the relevance of this approach.

Chapter 9

Finally, in Chapter 9, we propose a framework for analysing multiple differential cryptanalysis. This work is to be submitted and thus, we recommend to the reader to refer to the paper once it will be published (if it is ever the case). This work uses the framework presented in Chapter 6 for simple statistical cryptanalysis and elaborate on to adapt it to multiple differential cryptanalysis. The experimental results on a reduced version of PRESENT match the one obtained using the theoretical framework presented in the chapter. Hence, we propose a multiple differential attack on 18-round PRESENT what is the best known differential cryptanalysis of this cipher (it improves the previous cryptanalysis by 2 rounds).

Introduction générale

Les travaux exposés dans ce document portent essentiellement sur l'étude des cryptanalyses statistiques des chiffrements par blocs que nous aurons l'occasion de définir dans les premiers chapitres de cette thèse.

L'objectif initial de ces trois ans de recherche était la continuation des travaux effectués en stage de Master [Gér07] au sujet de l'utilisation d'outils issus de la théorie des codes correcteurs d'erreur pour les cryptanalyses statistiques et plus précisément pour la cryptanalyse linéaire. Ces travaux ont donné lieu à une publication à la conférence IMACC [GT09] dont le principal résultat est l'utilisation de l'entropie pour quantifier l'information extraite des échantillons disponibles lors d'une cryptanalyse linéaire multiple ainsi que l'application de cet outil à une cryptanalyse du DES qui, pour une faible complexité en données, s'avère être plus performante que celle proposée par Matsui. On détaillera, dans le chapitre 8, l'obtention d'une borne sur cette entropie et, résultat non encore publié, l'on s'intéressera à l'utilisation de l'entropie pour l'analyse de la cryptanalyse linéaire multidimensionnelle. Dans le chapitre 7, on s'intéressera à l'autre point de ce travail qui est l'utilisation d'un algorithme de décodage d'un code linéaire de faible rendement dans la phase d'analyse d'une cryptanalyse linéaire multiple. Cet algorithme probabiliste a pour avantage de ne pas avoir à regarder tous les candidats possibles pour retrouver la bonne clef.

La seconde partie du travail présenté ici est motivé par une problématique soulevée par les travaux de stage effectués par Céline Blondeau au sein de l'équipe SECRET. Ceux-ci portent sur la cryptanalyse différentielle tronquée et il a rapidement été question de comparer cette cryptanalyse avec la cryptanalyse différentielle standard. Travaillant avec Jean-Pierre Tillich sur l'analyse de la cryptanalyse linéaire, la question nous a naturellement été soumise. Après une recherche infructueuse dans la littérature, nous avons tous trois décidé de travailler sur ce sujet et plus généralement sur la question de l'évaluation « unifiée » de la complexité en données des cryptanalyses statistiques. En effet, les analyses effectuées reposent toutes sur une approximation de la distribution binomiale par une loi de probabilité plus facile à manipuler (gaussienne pour la cryptanalyse linéaire, loi de Poisson pour la cryptanalyse différentielle par exemple). Dans son article en 2008, Selçuk propose d'utiliser la même approximation gaussienne pour les deux cryptanalyses susmentionnées mais il s'avère que pour la cryptanalyse différentielle, cette approximation ne donne pas de bons résultats. Qu'en est-il alors des autres cryptanalyses statistiques ? Faut-il, pour chaque nouvelle attaque, tester quelle approximation est la plus appropriée ? Ces questions ont débouché sur une série de résultats pour le calcul précis de la complexité en données de telles attaques. On donne ainsi, dans [BG09b, BG09a] un algorithme permettant de calculer précisément la complexité en donnée d'une attaque en fonction de la proportion de mauvais candidats gardés et de la probabilité

de succès de l'attaque. De plus, deux estimation (une plus précise et une ayant une expression plus simple) de cette complexité en données sont obtenues pour une probabilité de succès de l'ordre de 0.5. Ces résultats viennent préciser les travaux de Thomas Baignères, Pascal Junod et Serge Vaudenay qui ont été effectués dans une optique asymptotique. Enfin, l'on s'est aussi intéressé au calcul de la probabilité de succès d'une cryptanalyse statistique pour une complexité en données et une taille de liste de candidats fixées. Le résultat obtenu utilise la même démarche que celle de Selçuk (que l'on retrouve dans le cas de la linéaire dans un papier de Pascal Junod) qui consiste à utiliser la théorie des statistiques d'ordres et la distribution bêta. Nous présentons dans [BGT10] une formule pour la probabilité de succès d'une cryptanalyse statistique similaire à celle donnée par Selçuk si ce n'est qu'elle ne fait pas intervenir d'approximation gaussienne et est donc valide pour toutes les cryptanalyses statistiques. Tout cela est détaillé dans le chapitre 6.

Après avoir travaillé sur la cryptanalyse linéaire, il me semblait indispensable pour parfaire ma connaissance du sujet, de travailler sur la cryptanalyse différentielle. Avec Céline Blondeau, nous nous sommes intéressé à la cryptanalyse différentielle ainsi qu'au chiffrement PRESENT qui est en passe de s'imposer comme référence au sein de la famille des chiffrements dits « légers ». De façon similaire à la cryptanalyse linéaire, de nombreuses questions surgissent depuis que les chiffrements sont conçus pour être résistants à la cryptanalyse différentielle et certaines hypothèses qui, jusqu'à présent, étaient raisonnables ne le sont plus pour les chiffrements récents. Nous avons donc utilisé une version réduite de PRESENT pour regarder expérimentalement ce qu'il en était [2]. Ces travaux présentés dans [BG10] sont en lien avec ceux de Joan Daemen et Vincent Rijmen et confirment les hypothèses qu'ils avaient proposées.

Une fois ces confirmations empiriques obtenues, du moins pour le cas de PRESENT, nous nous sommes attaqué au problème de l'utilisation de plusieurs différentielles lors d'une attaque. Utiliser plusieurs caractéristiques lors d'une cryptanalyse statistique est devenu une extension très utilisée mais dans le cas de la cryptanalyse différentielle, l'analyse de ce type d'attaque n'a jamais encore été mené sérieusement (à la différence de la cryptanalyse linéaire). Ce travail, qui se base sur l'approche utilisée dans le chapitre 6, est en cours de soumission au moment où j'écris ces lignes et est le sujet du dernier chapitre de ce document. Les résultats théoriques obtenus ont fait l'objet de validations expérimentales sur une version réduite du chiffrement PRESENT et nous sommes donc en mesure de proposer une cryptanalyse différentielle multiple qui améliore de deux tours la meilleure cryptanalyse différentielle connue sur PRESENT.

Références.

[Gér07] Benoît Gérard : Utilisation de techniques de codage correcteur d'erreurs pour la cryptanalyse de sytèmes de chiffrement à clef secrète. Mémoire de Master, Université de Versailles - Saint-Quentin en Yvelines, Versailles, 2007.

[GT09] Benoît Gérard et Jean-Pierre Tillich : On linear cryptanalysis with many linear approximations. *In* Matthew G. Parker, éditeur : *IMA International Conference, Cryptography and Coding - IMACC 2009*, volume 5921 de *LNCS*, pages 112–132. Springer, 2009.

2. Dans le cas de la cryptanalyse linéaire, des travaux similaires ont été effectués par Baudoin Collard et François-Xavier Standaert.

[BG09a] Céline Blondeau et Benoît Gérard : On the data complexity of statistical attacks against block ciphers. EUROCRYPT 2009 POSTERSESSION, 2009.

[BG09b] Céline Blondeau et Benoît Gérard : On the data complexity of statistical attacks against block ciphers (full version). *In* Alexander Kholosha, Eirik Rosnes et Matthew G. Parker, éditeurs : *Workshop on Coding and Cryptography - WCC 2009*, pages 469–488, 2009.

[BGT10] Céline Blondeau, Benoît Gérard et Jean-Pierre Tillich : Accurate estimates of the data complexity and success probability for various cryptanalyses. *DCC special issue on Coding and Cryptography*, 2010. À paraître.

[BG10] Céline Blondeau et Benoît Gérard : Links between theoretical and effective differential probabilities : Experiments on PRESENT. *In TOOLS'10*, 2010.

Chapitre 1

Introduction à la cryptologie

Sommaire

La cryptologie est la science de la protection des données. Par protection des données, on peut entendre plusieurs choses :

- la *confidentialité* qui consiste à protéger le contenu de sa divulgation.
- l'*intégrité* ou la protection du contenu face à une modification extérieure. Concrètement, comme l'on ne peut empêcher une modification, on cherche à garantir la non modification d'une donnée.
- l'*authenticité* qui consiste à vérifier l'identité d'une personne afin d'éviter de transmettre des données au mauvais interlocuteur.

1.1 Les différentes primitives cryptographiques

Afin d'atteindre les objectifs de sécurité susmentionnés, on va faire appel à ce que l'on nomme *primitives cryptographiques* qui sont, en quelque sorte, les blocs de base permettant la construction de protocoles cryptographiques.

Cryptographie symétrique/asymétrique. La cryptographie est divisée en deux grands domaines : la cryptographie symétrique et la cryptographie asymétrique. Comme leurs noms l'indiquent, la différence entre ces deux domaines est une question de symétrie. En cryptographie symétrique, on suppose que tous les intervenants possèdent le même secret alors qu'en cryptographie asymétrique, un seul des intervenants possède un secret.

Voici une liste des principales primitives que l'on peut rencontrer.

Fonction de hachage. Si je débute par les fonctions de hachage c'est parce qu'elle sont une sorte de « couteau suisse » de la cryptographie. On les retrouve aussi dans des problèmes d'algorithmique ou de base de données. Une fonction de hachage est une fonction qui prend une chaîne de bits de longueur arbitraire finie en entrée et qui retourne une chaîne de bits de longueur fixée appelée *haché*.

$$h : \{0;1\}^* \to \{0;1\}^s.$$

Chacune des diverses utilisations des fonctions de hachage induit ses propres notions de sécurité parmi lesquelles la plus répandue est la résistance aux collisions, c'est à dire qu'il est difficile de trouver deux entrées ayant le même haché.

Code d'authentification de message *(symétrique).* Les codes d'authentification de message permettent d'assurer l'intégrité du message ainsi que de l'authentifier. Attention toutefois, l'authentification ne permet de détecter qu'un groupe d'émetteurs potentiels. En effet, le principe est de partager une clef entre émetteur(s) et récepteur(s). On se place donc dans le domaine de la cryptographie symétrique. Lors de l'envoi d'un message, on y ajoute un code d'authentification, ayant pour notation anglo-saxonne MAC, qui est l'image par une fonction h du message et de la clef. Si on fait partie du groupe ayant connaissance de la clef, on peut alors vérifier que l'image par h du message et de la clef correspond bien au MAC.

Algorithme de signature *(asymétrique).* Ce que l'on demande à une signature au sens cryptographique est ce que l'on aimerait avoir avec une signature manuscrite : qu'elle permette d'authentifier l'auteur du document, que celui-ci ne puisse nier qu'il en est l'auteur, que personne ne puisse réutiliser la signature pour un autre document, et que personne ne puisse signer à la place de quelqu'un d'autre. On est ici dans un schéma où seule une personne possède le secret qui permet de signer et on est donc dans le cadre de la cryptographie asymétrique.

Générateur de pseudo-aléa. Un générateur de pseudo-aléa doit produire une suite de nombres dont les propriétés doivent s'approcher au plus de celles d'une suite aléatoire. Pour cela, on donne à un tel algorithme une graine à partir de laquelle il va devoir générer une suite. Ces algorithmes étant déterministes, la connaissance de la graine implique la connaissance de la suite.

Algorithme de chiffrement *(symétrique/asymétrique).* On ne va pas s'attarder sur ces primitives ici car elles font l'objet de cette thèse. L'objectif d'un algorithme de chiffrement est de permettre à plusieurs parties de correspondre sans que les personnes non autorisées aient accès au contenu des messages. Dans l'approche asymétrique du chiffrement, chaque personne possède une clef privée et une clef publique. Toute personne souhaitant envoyer un message à une autre doit utiliser la clef publique du destinataire pour chiffrer son message que seul le détenteur de la clef privée correspondante pourra déchiffrer. Dans le cas symétrique, deux personnes s'échangent une clef secrète qui leur permettra de correspondre par la suite. Cela implique que l'échange de clefs ait été effectué au préalable ainsi que de posséder une clef par interlocuteur (ce qui est somme toute assez contraignant).

La contrainte de l'échange de clef peut être levée grâce à la cryptographie asymétrique. Donc, plutôt que de chiffrer un message entier avec un chiffrement asymétrique (beaucoup plus lent qu'un algorithme symétrique), on échange juste une clef qui permettra, ensuite, de chiffrer le message à l'aide d'un algorithme symétrique. C'est pourquoi, en pratique, les

algorithmes de chiffrement symétrique, qui seront étudiés dans ce document, sont largement utilisés.

1.2 Deux familles de chiffrements symétriques

Les chiffrements symétriques sont répartis en deux grandes familles.

1.2.1 Chiffrements à flot

La première est la famille des chiffrements à flot qui, comme son nom l'indique, est composée d'algorithmes qui chiffrent à la volée. L'idée sous-jacente est d'utiliser le chiffrement de Vernam, c'est-à-dire additionner une clef au message. Seulement, on ne veut pas avoir une clef aussi longue que le message car cela reste très peu pratique. On va donc utiliser une clef de taille fixée afin de générer une suite chiffrante qui sera additionnée au message pour produire le chiffré. Un chiffrement à flot est donc un algorithme qui à partir d'une clef, génère une suite chiffrante à qui l'on demande d'avoir un comportement aussi proche que possible d'une suite aléatoire afin de masquer au mieux un message. En cela, on voit facilement la similitude avec les générateurs de pseudo-aléa et, en effet, les chiffrements à flot sont de conception similaire à ces générateurs.

1.2.2 Chiffrements par blocs

Comme l'indique le titre de ce document, nous nous intéressons aux attaques statistiques des chiffrements à clef secrète. Cependant, il est plus spécialement question des chiffrements par blocs. À la différence du chiffrement à flot pour lequel on n'a pas besoin de connaître le message pour commencer à travailler (on n'en a besoin qu'en fin d'algorithme quand on additionne la suite chiffrante), on traite le message morceau par morceau (bloc par bloc). Un algorithme de chiffrement par blocs est donc définit comme suit.

Définition 1.1. Un chiffrement par blocs est une famille de permutations $(E_{\boldsymbol{K}})$ paramétrée par une clef $\boldsymbol{K} \in \mathcal{K}$ de k bits et traitant des blocs de taille s.

$$E_{\boldsymbol{K}} : \mathbb{F}_2^s \to \mathbb{F}_2^s.$$

1.3 Notions de sécurité pour les chiffrements par blocs

Deux questions se posent quand un nouvel algorithme de chiffrement voit le jour. Ce sont les questions de sécurité et de rapidité. Bien entendu, le but est d'obtenir une sécurité maximale pour un temps d'exécution minimal.

Il est important, avant de commencer, de préciser que lorsque l'on évalue la sécurité d'un algorithme de chiffrement, on suppose qu'il est connu. En effet, la sécurité ne doit pas reposer sur le fait que l'algorithme utilisé est secret car en général un tel algorithme est faible (car peu étudié) et dès que l'on réussi à faire de la rétro-ingénierie il est immédiatement cassé (on pense aux algorithmes A5/1 et 2 utilisés en téléphonie mobile). Ce principe est connu sous le nom de *Principe de Kerckhoffs*. En réalité, celui-ci a énoncé sept règles dans une revue militaire en 1883 [Ker83]. Celle qui est la plus connue (car encore d'actualité) est énoncée ainsi :

Les interlocuteurs ne doivent pas subir de dégâts au cas où le système de codage serait dévoilé.

La menace principale sur un algorithme de chiffrement par blocs est de pouvoir retrouver la clef utilisée et ainsi pouvoir déchiffrer les transmissions. On suppose donc que, pour une attaque, le cryptanalyste a accès à certains échantillons de messages chiffrés avec une certaine clef. Il existe plusieurs modèles définissant les modalités d'accès à ces échantillons car l'on peut imaginer qu'il ait accès aux clairs correspondant aux chiffrés ou même qu'il puisse choisir lui même les clairs. Une fois ces échantillons obtenus, il lui reste à retrouver la clef utilisée pour générer ces échantillons. Les quatre caractéristiques permettant d'évaluer la performance d'une attaque sont les suivantes.

- la *complexité en données* : le nombre de messages chiffrés nécessaires à l'attaque.
- la *complexité en temps* : le nombre d'opérations nécessaires à l'attaque. On donne parfois une borne, parfois une complexité moyenne. De plus, il faut faire attention à l'unité car il s'agit parfois d'opérations binaires et dans d'autre cas d'applications du système de chiffrement.
- la *complexité en mémoire* : la quantité d'espace nécessaire à l'attaque.
- la *probabilité de succès* : la probabilité de trouver la bonne clef avant l'arrêt de l'algorithme.

De nos jours, un chiffrement est considéré sûr (d'un point de vue pratique) si la complexité en temps de la meilleure attaque connue est supérieure à 2^{80} si l'on veut obtenir une probabilité significative de retrouver la clef.

Comme il a été dit plus haut, il existe plusieurs modèles d'attaquant, plus ou moins réalistes, qui définissent différentes façons d'obtenir les échantillons. Ces différents modèles d'attaque sont listés ci-après par ordre croissant, ce qui signifie que les modèles définiront des attaquants de plus en plus puissants.

Chiffré seul. Ce modèle d'attaque est le modèle qui parle le plus car il correspond à la plupart des décryptages historiques. En effet, ce modèle d'attaque suppose que le cryptanalyste dispose uniquement d'un certain nombre de messages chiffrés. On peut citer par exemple la technique de décryptage du chiffre de César en utilisant une étude statistique de la distribution des lettres (le « E » est le plus fréquent, ...).

Clair connu. Dans ce modèle, on donne un peu plus de pouvoir à l'attaquant puisque l'on suppose qu'il connaît les messages clairs correspondant aux chiffrés interceptés. On se retrouve alors comme Champollion et sa pierre de Rosette. Ce modèle n'est pas si irréaliste que cela puisqu'il correspond à certains décryptages survenus durant les guerres. On peut, en effet, intercepter le message « Attaquez » et subir une attaque quelques minutes plus tard. On saura alors que le message envoyé était celui-ci. Un exemple plus moderne est le décryptage de la norme WEP du wi-fi. Les paquets envoyés sur le réseau possèdent tous une entête qui contient le clair correspondant à une partie du chiffré (vecteur d'initialisation).

Clair choisi (adaptatif ou non). On augmente encore un peu le niveau de sécurité en autorisant l'attaquant à choisir les clairs qui seront chiffrés. Cela devient plus difficile à faire en pratique mais cela reste réalisable. On peut imaginer un accès temporaire à la « machine » servant à chiffrer et à laquelle on pourrait soumettre un certain nombre de clairs avant de se faire repérer.

La différence en cas d'attaque adaptative est que l'on s'autorise le fait de choisir le n-ième clair en fonction des $n-1$ chiffrés précédents. Si on imagine une opération commando pour accéder à la salle contenant les systèmes de communication, il faudra alors penser à emmener le cryptanalyste.

Chiffré choisi (adaptatif ou non). Ici on est dans la situation duale du modèle précédent (ce modèle n'est donc pas forcément plus puissant). On a accès à la machine qui déchiffre et on peut supposer que l'attaquant peut adapter ses requêtes ou non aux réponses obtenues.

Clair et chiffré choisis (adaptatif ou non). On arrive maintenant un modèle bien plus puissant car on a la possibilité, ici, de chiffrer et de déchiffrer les messages souhaités. Le cas adaptatif revient à avoir à sa disposition une boîte noire qui chiffre et déchiffre.

Clefs liées. Mentionnons, ici, les attaques à clefs liées qui ne se sont pas encore fait une place au tableau d'honneur de la cryptanalyse. En effet, en plus des qualificatifs décrits ci-dessus, on peut ajouter le fait d'obtenir des échantillons chiffrés avec plusieurs clefs ayant, entre elles, une relation connue du cryptanalyste (une différence fixée par exemple).

1.4 Constructions de chiffrements itératifs par blocs

Nous allons étudier les cryptanalyses statistiques des chiffrements par blocs. De telles cryptanalyses sont principalement appliquées à des chiffrement itératifs, c'est-à-dire des algorithmes qui consistent à itérer une certaine fonction (appelée fonction de tour) un certain nombre de fois. La seule différence entre les tours est la clef utilisée.

Définition 1.2. La clef utilisée pour paramétrer un algorithme de chiffrement est appelée *clef maître*. Quand le chiffrement utilisé est itératif, on utilise à chaque tour une clef différente appelée *sous-clef*. Les sous-clefs sont extraites de la clef maître en utilisant un *algorithme de cadencement de clef*.

On écrit alors $E_{\boldsymbol{K}}(X) = F_{\boldsymbol{K}_r} \circ \cdots \circ F_{\boldsymbol{K}_1}(X)$, où les sous-clefs sont numérotées de 1 au nombre de tours r.

Il existe deux principales structures de chiffrements itératifs. Dans ce document, nous nous intéresserons à un algorithme pour chacune de ces deux constructions dont voici un bref aperçu.

1.4.1 Réseaux de substitution-permutation

Un algorithme de chiffrement de type substitution-permutation (l'acronyme anglais SPN sera utilisé par la suite) est constitué d'une succession de tours de la forme $P \circ S \circ M$ où M est l'opération de mélange avec la clef (la clef est xorée à l'état essentiellement), S est une fonction de substitution (c'est-à-dire une bijection quelconque de $\mathbb{F}_2^m$ dans $\mathbb{F}_2^m$) et P une permutation (au sens de réordonner les bits). En fait, on considère comme SPN les chiffrements où P est une opération linéaire sur l'état (on autorise donc le « XOR » entre parties de l'état). Cependant, on préférera s'en tenir à la définition stricte de SPN c'est-à-dire que P est une permutation bit à bit.

Cette construction (représentée sur la figure 1.1) est utilisée dans différents algorithmes de chiffrement tels que l'AES[1] *(Advanced Encryption Standard)* [DR00], SERPENT [ABK00], PRESENT [BKL+07], PUFFIN [CHW08], ...

Notons que pour certains chiffrements, le mélange de la clef et la substitution ne font qu'un. Dans le cas de `C` [BF07], par exemple, il n'y a pas d'addition de la clef car celle-ci sert à définir la fonction de substitution.

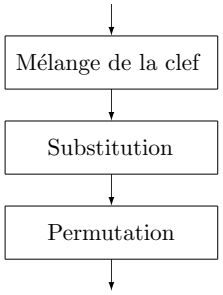

FIGURE 1.1 – Un tour d'un réseau de substitution-permutation

Boîte-S. La fonction S étant le seul élément non-linéaire du chiffrement, il est donc important que son comportement soit proche d'une permutation aléatoire. De telles fonctions sont très coûteuses en terme de temps de calcul ou bien de mémoire. On utilise alors la stratégie du « diviser pour mieux régner » en divisant l'état en petits morceaux et en utilisant ce que l'on appelle des *boîtes-S*. Une boîte-S est une bijection agissant sur un petit nombre de bits et donc prend peu de place en mémoire. La strate de substitution est alors composée d'une rangée de ces boîtes-S (identiques ou non), chaque bit de l'état étant impliqué dans une boîte. On reviendra plus tard sur ces boîtes-S et les propriétés souhaitées de celles-ci car elles sont au centre des cryptanalyses statistiques.

1.4.2 Réseaux de Feistel

Les réseaux de Feistel, dont le nom provient de l'ingénieur IBM Horst Feistel, sont des algorithmes de chiffrements itératifs agissant sur des demi blocs de message. Un tour d'un tel chiffrement est représenté dans la figure 1.2. Le message clair $\boldsymbol{M}$ est découpé en deux parties de même taille $\boldsymbol{M} = (\boldsymbol{M}^g, \boldsymbol{M}^d)$, le chiffré $\boldsymbol{C} = (\boldsymbol{C}^g, \boldsymbol{C}^d)$ est obtenu de la façon suivante :

$$\begin{cases} \boldsymbol{M}^g = \boldsymbol{M}^d, \\ \boldsymbol{C}^d = \boldsymbol{M}^g \oplus F(\boldsymbol{M}^g). \end{cases} \tag{1.1}$$

Cette construction est utilisée dans un grand nombre d'algorithmes de chiffrement tels que Lucifer, DES, Blowfish, Camellia, ...

Il est important de préciser qu'il n'y a aucune restriction sur la fonction F et en particulier qu'elle n'a pas à être inversible pour que le schéma soit inversible. En effet, on remonte un tour en utilisant les formules suivantes

$$\begin{cases} \boldsymbol{M}^g = \boldsymbol{C}^d \oplus F(\boldsymbol{C}^g), \\ \boldsymbol{M}^d = \boldsymbol{C}^g. \end{cases}$$

1. Ceci n'est vrai que si l'on considère la définition large de SPN.

Une telle construction est facilement distinguable d'une permutation aléatoire si le nombre de tours est trop petit. Il faut donc utiliser au moins 6 tours quand on fait appel à ce type de construction [Pat04].

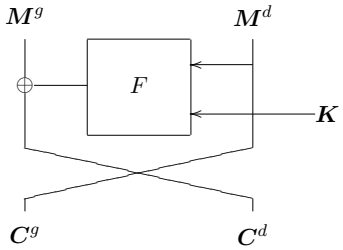

FIGURE 1.2 – Un tour d'un chiffrement de type réseau de Feistel

Quelques généralisations. Plusieurs généralisations de ce schéma ont vu le jour depuis sa création en 1971. On peut citer, par exemple, les réseaux de Feistel non équilibrés [SK96] ou bien les réseaux de Feistel généralisés [Nyb96].

1.5 Cryptanalyses statistiques des chiffrements par blocs

Toutes les cryptanalyses présentées dans ce document sont des cryptanalyses statistiques. Ces attaques reposent sur une *caractéristique statistique* du chiffrement c'est-à-dire un phénomène qui a une probabilité d'arriver différente de celle avec laquelle il arriverait si on regardait une sortie uniformément distribuée sur l'espace des chiffrés. De telles attaques ont pour but de retrouver tout ou partie de la clef utilisée pour chiffrer les échantillons interceptés. On note $\boldsymbol{k}^*$ cette sous-clef recherchée et n_k le nombre de bits qui la composent. Une cryptanalyse statistique est composée des trois étapes présentées ci-dessous.

- *Phase de distillation* : une statistique Σ est extraite des N échantillons disponibles.
- *Phase d'analyse* : de cette statistique Σ, on extrait de l'information afin de calculer la vraisemblance de chaque candidat pour $\boldsymbol{k}^*$ et générer une liste $\mathcal{L}$ des ℓ candidats les plus vraisemblables.
- *Phase de recherche* : pour chacun des candidats sélectionnés (appartenant à $\mathcal{L}$) on teste toutes les clefs maîtres correspondantes jusqu'à trouver la bonne.

Lors de la phase d'analyse, on extrait une statistique $\Sigma_{\boldsymbol{k}}$ de Σ pour chaque candidat $\boldsymbol{k}$. C'est ce $\Sigma_{\boldsymbol{k}}$ qui nous permet de calculer la vraisemblance de chaque candidat $\boldsymbol{k}$. L'efficacité d'une cryptanalyse statistique repose sur le fait que la caractéristique $\Sigma_{\boldsymbol{k}^*}$ correspondant au bon candidat a un comportement différent des autres.

On a vu que, selon les attaques, on obtenait les échantillons de façons différentes (clair connu ou choisi, etc.). Il s'avère que le nombre de messages chiffrés contenus dans un échantillon peut aussi varier d'une attaque à l'autre.

Définition 1.3. L'*ordre* d'une cryptanalyse statistique est le nombre de chiffrés contenus dans un échantillon.

Par exemple, la cryptanalyse linéaire présentée dans le chapitre 4 a pour ordre 1, la différentielle 2 et la différentielle d'ordre supérieur 2^d où d est le degré de dérivation (cf. chapitre 5).

Regardons maintenant la complexité en temps de telles attaques. On a dit qu'elle était liée à la taille ℓ de la liste des candidats gardés vu qu'il faudra effectuer une recherche sur toutes les clefs correspondant à des candidats de $\mathcal{L}$.

Le *gain* d'une attaque par rapport à la recherche exhaustive ($\mathcal{L} = \mathbb{F}_2^{n_k}$) est défini dans [BDCQ04].

Définition 1.4. (Gain). Dans le cadre d'une attaque sur un chiffrement acceptant un nombre $|\mathcal{K}|$ de clefs, on note $\bar{\Psi}$ le nombre moyen de clefs à tester pour trouver $\boldsymbol{k}^*$. Le *gain* de l'attaque est alors

$$\Gamma \stackrel{\text{def}}{=} -\log_2\left(\frac{2\bar{\Psi}-1}{|\mathcal{K}|}\right).$$

Dans le cas de la recherche exhaustive on a $\bar{\Psi} = 1 + \frac{|\mathcal{K}|-1}{2}$ et donc $\Gamma = 0$ et dans le cas d'une cryptanalyse tombant avec probabilité 1 sur $\boldsymbol{k}^*$ du premier coup, on a $\bar{\Psi} = 1$ et donc $\Gamma = n_k$. La notion d'*avantage* est définie par Selçuk dans [Sel08].

Définition 1.5. (Avantage). Dans le cadre d'une attaque retrouvant n_k bits de clef, on note ℓ le nombre maximum de candidats testés pour trouver $\boldsymbol{k}^*$. Alors, l'*avantage* de cette attaque est de

$$a \stackrel{\text{def}}{=} n_k - \log_2(\ell) \text{ bits}.$$

De même, pour la recherche exhaustive on a $\ell = n_k$ et donc $a = 0$ et pour le cas $\ell = 1$ on a bien un avantage $a = n_k$. Lorsque l'on cherche à estimer la puissance d'une attaque, on s'intéresse à sa complexité en données N, son avantage a et sa probabilité de succès P_S. Chacun de ces paramètres est une fonction des deux autres. Ainsi, pour estimer théoriquement l'avantage d'une attaque, il faut avoir fixé une probabilité de succès. Pour des probabilités de succès raisonnables et une complexité en données fixée, l'avantage ne varie pas beaucoup. De plus, dans la plupart des formules, la probabilité de succès apparaît dans le terme $\Phi^{-1}(P_S)$ où Φ^{-1} est la réciproque de la fonction de répartition de la loi normale centrée réduite. On a donc une définition alternative de l'avantage d'une attaque qui correspond mieux à l'utilisation qui en est faite dans la littérature.

Définition 1.6. (Avantage 2). Dans le cadre d'une attaque retrouvant n_k bits de clef, on note ℓ la taille de la liste de candidats à garder afin d'avoir 50% de chances d'avoir retenu $\boldsymbol{k}^*$. Alors, l'*avantage* de cette attaque est de

$$a \stackrel{\text{def}}{=} n_k - \log_2(\ell) \text{ bits} = -\log_2\left(\frac{\text{Med}(\Psi)}{|\mathcal{K}|}\right) \text{bits},$$

où $\text{Med}(\Psi)$ désigne la valeur médiane du rang de la bonne clef parmi les $|\mathcal{K}|$ candidats.

L'on va maintenant s'intéresser aux deux approches possibles pour l'analyse d'une cryptanalyse statistique.

≪ **Examen** ≫. Dans cette approche, on se fixe une règle de décision qui dit si l'on doit considérer un candidat comme valable. On est donc dans une approche de type *examen* où un candidat est reçu s'il obtient une note supérieure à un seuil fixé. La taille ℓ de la liste $\mathcal{L}$ de candidats gardés est alors non bornée (si ce n'est par le nombre total de candidats). En contrepartie, on peut facilement calculer la probabilité de succès de l'attaque qui est la probabilité que la règle de décision accepte le bon candidat.

≪ **Concours** ≫. Cette seconde approche est totalement orthogonale à la première vu que l'on fixe la taille ℓ de la liste. C'est en quoi cette approche est semblable à un *concours* où un *numerus closus* est imposé. Moralement, cela revient à définir une règle de décision a posteriori pour laquelle seuls les ℓ candidats les plus vraisemblables seront gardés. L'avantage de cette approche est que l'on peut ainsi borner la complexité en temps de l'attaque de façon précise. L'inconvénient, bien sûr, est que la probabilité de succès est un peu plus difficile à calculer car il faut calculer la probabilité que le bon candidat ne fasse pas partie des ℓ plus vraisemblables. Ce calcul peut cependant se faire plus aisément dans le cas particulier où $\ell = 1$ appelé *top ranking*.

Attaque sur le dernier tour. Pour illustrer un peu tout ce qui a été dit, on va présenter un exemple typique d'attaque statistique : les *cryptanalyses sur le dernier tour*. De telles attaques utilisent une caractéristique statistique sur r tours du chiffrement pour attaquer $r+1$ tours [2] qui arrive avec une probabilité p_0 différente de la probabilité p avec laquelle elle arriverait si on avait une permutation aléatoire. On déchiffre alors les morceaux des messages interceptés permettant de mettre en évidence ce phénomène avec toutes les sous-clefs de tour possibles. Pour la bonne sous-clef le phénomène se produit avec une probabilité p_0. Pour un mauvais candidat, on suppose que déchiffrer avec une mauvaise sous-clef revient à avoir une sortie aléatoire et donc le phénomène devrait apparaître avec probabilité p.

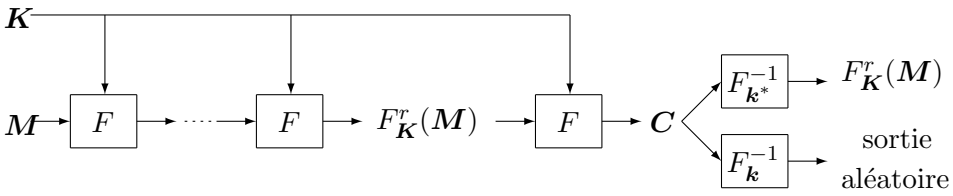

FIGURE 1.3 – Cryptanalyse sur le dernier tour.

L'hypothèse essentielle sur laquelle repose cette attaque (ou en tout cas son analyse) est le fait que le déchiffrement avec un mauvais candidat ait le comportement d'une permutation aléatoire. Cette hypothèse est appelée *hypothèse de répartition aléatoire par fausse clé*.

Hypothèse 1.7. *Hypothèse de répartition aléatoire par fausse clé (générale).*

$$\forall \boldsymbol{y} \in \mathbb{F}_2^s,\ \forall \boldsymbol{k} \neq \boldsymbol{k}^*,\ \Pr_{\boldsymbol{M}}\left[F_{\boldsymbol{k}}^{-1}\left(F_{\boldsymbol{K}}^{r+1}(\boldsymbol{M})\right) = \boldsymbol{y}\right] = 2^{-s}.$$

Et donc, dans le cas d'une cryptanalyse statistique, cela implique que

$$\forall i,\ \boldsymbol{k} \neq \boldsymbol{k}^*\ \boldsymbol{k}' \neq \boldsymbol{k}^*, \quad \Pr\left[\Sigma_{\boldsymbol{k}} = i\right] = \Pr\left[\Sigma_{\boldsymbol{k}'} = i\right].$$

Hypothèse d'équivalence à clef fixée. Terminons par une dernière remarque générale sur les cryptanalyses statistiques. Il était de coutume de supposer que le comportement de la caractéristique statistique observée est identique quelle que soit la clef utilisée pour chiffrer les échantillons. Cette hypothèse, connue sous le nom d'*hypothèse d'équivalence à clef fixée*, s'énonce de façon générale comme suit.

2. Plus généralement, on peut attaquer $r + r'$ tours avec $r' \geq 1$. Pour simplifier, on supposera que $r' = 1$.

Hypothèse 1.8. *Hypothèse d'équivalence à clef fixée (générale).*
On note $\Sigma_{\boldsymbol{k}^*} \in \mathcal{I}$ *la statistique associée à une clef* $\boldsymbol{k}^*$ *utilisée dans un chiffrement. Alors, pour* $\boldsymbol{k}_1^* \neq \boldsymbol{k}_2^*$ *et pour tout* $i \in \mathcal{I}$*,*

$$\Pr\left[\Sigma_{\boldsymbol{k}_1^*} = i\right] = \Pr\left[\Sigma_{\boldsymbol{k}_2^*} = i\right].$$

On a eu l'occasion, depuis, de s'apercevoir que cette hypothèse est loin d'être vérifiée avec les chiffrements actuels. On verra cela plus en détail dans les cas de la cryptanalyse linéaire (chapitre 4) et de la cryptanalyse différentielle (chapitre 5).

Chapitre 2

Deux exemples de chiffrements par blocs

Sommaire

Numérotation des bits. Nous donnons ici les spécifications des deux chiffrements faisant l'objet d'études dans ce document. Afin que les choses soient claires concernant les opérations sur les bits, il peut s'avérer utile de mettre les choses au point concernant la numérotation de ces bits.

On utilise ici la convention suivante : le bit de poids faible a pour indice 0 et sera représenté à droite de l'expression (ce qui colle au mieux avec la numérotation décimale latine). Par exemple, le nombre 137 s'écrira `0x89` en hexadécimal et $\texttt{10001001}_\texttt{b}$ en binaire :

Bit numéro	7	6	5	4	3	2	1	0
Valeur	1	0	0	0	1	0	0	1

2.1 Data Encryption Standard (DES)

2.1.1 Chiffrement

Le DES (*Data Encryption Standard*) est un algorithme de chiffrement par blocs lancé en 1976 par le *National Bureau of Standards* (NBS) en tant que standard américain et qui s'est vite vu devenir la référence en terme de chiffrement symétrique. Le DES est constitué d'un réseau de Feistel de 16 tours dont la fonction F, représentée dans la figure 2.1, agit sur deux blocs de 32 bits. Dans le standard, une permutation initiale est effectuée en début de

chiffrement (son inverse est ajoutée à la fin du processus) mais nous ne détaillerons pas cela ici car ces permutations n'ont aucun intérêt concernant l'étude de la sécurité du chiffrement.

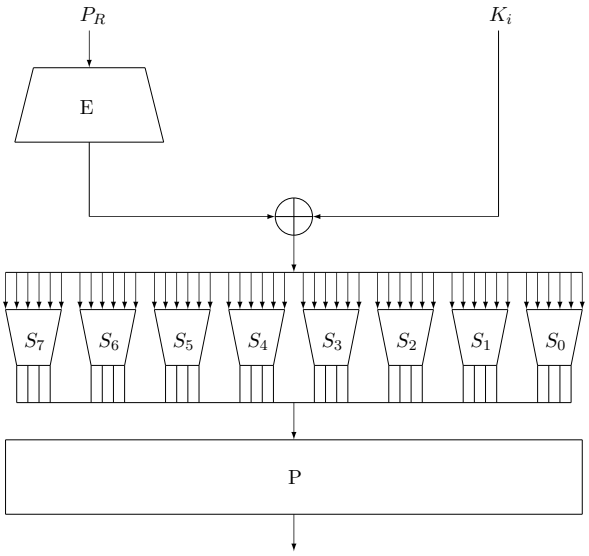

FIGURE 2.1 – Fonction F du DES.

À l'origine, le chiffrement se nommait Lucifer. Ce chiffrement agissait sur des blocs de 128 bits et avait une clef secrète constituée de 128 bits mais la NSA (*National Security Agency*) demanda à ce que ce nombre soit revu à la baisse et le DES tel qu'il fut normalisé utilise une clef de 56 bits pour des blocs de 64 bits.

Ci-après, nous donnons les caractéristiques des fonctions E (tableau 2.1) et P (tableau 2.2) du DES. Les tableaux se lisent ainsi : le bit numéro 42 du message étendu correspond au bit 27 du message en entrée de l'expansion.

$E(X)_i$	47	46	45	44	43	42	41	40	39	38	37	36
X_i	0	31	30	29	28	27	28	27	26	25	24	23
$E(X)_i$	35	34	33	32	31	30	29	28	27	26	25	24
X_i	24	23	22	21	20	19	20	19	18	17	16	15
$E(X)_i$	23	22	21	20	19	18	17	16	15	14	13	12
X_i	16	15	14	13	12	11	12	11	10	9	8	7
$E(X)_i$	11	10	9	8	7	6	5	4	3	2	1	0
X_i	8	7	6	5	4	3	4	3	2	1	0	31

TABLE 2.1 – Expansion E de la fonction F du DES.

Nous allons maintenant donner les tables des boîtes-S du DES (tableau 2.3). Ces boîtes-S, contrairement à la plupart de leurs congénères, ne sont pas carrées (comme on le voit sur la figure 2.1). Elles prennent 6 bits en entrée et ont des sorties codées sur 4 bits. Étant donnée l'expansion présentée au tableau 2.1 (pour un groupe de 6 bits, les bits extrêmes sont redondants avec les groupes voisins), la coutume veut que l'on représente les boîtes-S du DES par un tableau à double entrée. Les deux bits extrêmes de la valeur d'entrée indexent

$P(X)_i$	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
X_i	16	25	12	11	3	20	4	15	31	16	9	6	27	14	1	22
$P(X)_i$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X_i	30	24	8	18	0	5	29	23	13	19	2	26	10	21	28	7

TABLE 2.2 – Permutation P de la fonction F du DES.

les lignes et les quatre bits centraux les colonnes. Par exemple, pour la boîte S_0, l'image de $\texttt{0x29} = \texttt{101001}_\texttt{b}$ se cherche ligne `0x3` colonne `0x4`.

Notons que l'on numérote les boîtes à partir de 0 et de droite à gauche pour garder la cohérence avec la numérotation des bits et les notations qui seront utilisées pour PRESENT. Ce n'est pas le cas dans certains papiers de cryptanalyse du DES.

2.1.2 Cadencement des clefs

Occupons-nous, maintenant, de l'algorithme de cadencement de clefs. La clef ne contient que 56 bits mais elle est codée sur 64 bits. Les bits dont la position est un multiple de 8 sont utilisés comme somme de contrôle des 7 précédents. La première étape est donc d'extraire les 56 bits utiles des 64 bits de clef. Ceci est fait suivant le tableau 2.4.
L'algorithme de cadencement de clefs est le suivant.

1. On extrait les 56 bits des 64 données en utilisant le tableau 2.4.
2. On initialise le compteur de tour r à 1.
3. On sépare l'état de 56 bits en deux parts de 28.
4. On effectue sur chacun de ces morceaux une rotation vers la gauche dont le décalage est donné dans le tableau 2.5.
5. On extrait les 48 bits de la sous-clef du tour r selon le tableau 2.6.
6. On incrémente r et si $r \leq 16$ on retourne à l'étape 3.

Comme on peut le constater, cet algorithme est linéaire. On peut même aller plus loin : les bits des sous-clefs correspondent exactement aux bits de la clef maître. Il est donc facile d'utiliser l'information que l'on a sur une sous-clef afin d'obtenir de l'information sur la clef maître. On verra que c'est loin d'être les cas dans certains chiffrements récents.

2.1.3 Cryptanalyses et successeur

Il est tout d'abord intéressant de noter que le DES possède une propriété de complémentation. En effet, $\texttt{DES}_k(X) = \overline{\texttt{DES}_{\bar{k}}(\bar{X})}$ où la notation $\bar{\cdot}$ correspond au complément bit à bit (les 0 deviennent 1 et inversement). Ceci permet de diminuer de moitié l'effort lors d'une recherche exhaustive si l'on possède les chiffrés correspondant à deux clairs complémentaires. Le DES est le premier « chiffrement moderne » a avoir été étudié à travers le monde entier. Son étude a donné naissance à deux des cryptanalyses les plus connues : la cryptanalyse différentielle [BS91a, BS91b, BS93] et la cryptanalyse linéaire [TCG92, Mat94b]. C'est pourtant le faible nombre de bits de clef qui a sonné le glas de cet algorithme. En 1998 tout d'abord, Deep Crack (machine ayant coûté environ 200 000 dollars) cassa une clé en 56 heures. Vint ensuite le calcul distribué utilisant les ordinateurs des particuliers (distributed.net) qui retrouva une clef en 22 heures et 15 minutes. Récemment, les universités de Bochum et Kiel ont mis au

S_0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	D	2	8	4	6	F	B	1	A	9	3	E	5	0	C	7
1	1	F	D	8	A	3	7	4	C	5	6	B	0	E	9	2
2	7	B	4	1	9	C	E	2	0	6	A	D	F	3	5	8
3	2	1	E	7	4	A	8	D	F	C	9	0	3	5	6	B

S_1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	4	B	2	E	F	0	8	D	3	C	9	7	5	A	6	1
1	D	0	B	7	4	9	1	A	E	3	5	C	2	F	8	6
2	1	4	B	D	C	3	7	E	A	F	6	8	0	5	9	2
3	6	B	D	8	1	4	A	7	9	5	0	F	E	2	3	C

S_2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	C	1	A	F	9	2	6	8	0	D	3	4	E	7	5	B
1	A	F	4	2	7	C	9	5	6	1	D	E	0	B	3	8
2	9	E	F	5	2	8	C	3	7	0	4	A	1	D	B	6
3	4	3	2	C	9	5	F	A	B	E	1	7	6	0	8	D

S_3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	2	C	4	1	7	A	B	6	8	5	3	F	D	0	E	9
1	E	B	2	C	4	7	D	1	5	0	F	A	3	9	8	6
2	4	2	1	B	A	D	7	8	F	9	C	5	6	3	0	E
3	B	8	C	7	1	E	2	D	6	F	0	9	A	4	5	3

S_4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	7	D	E	3	0	6	9	A	1	2	8	5	B	C	4	F
1	D	8	B	5	6	F	0	3	4	7	2	C	1	A	E	9
2	A	6	9	0	C	B	7	D	F	1	3	E	5	2	8	4
3	3	F	0	6	A	1	D	8	9	4	5	B	C	7	2	E

S_5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	A	0	9	E	6	3	F	5	1	D	C	7	B	4	2	8
1	D	7	0	9	3	4	6	A	2	8	5	E	C	B	F	1
2	D	6	4	9	8	F	3	0	B	1	2	C	5	A	E	7
3	1	A	D	0	6	9	8	7	4	F	E	3	B	5	2	C

S_6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	F	1	8	E	6	B	3	4	9	7	2	D	C	0	5	A
1	3	D	4	7	F	2	8	E	C	0	1	A	6	9	B	5
2	0	E	7	B	A	4	D	1	5	8	C	6	9	3	2	F
3	D	8	A	1	3	F	4	2	B	6	7	C	0	5	E	9

S_7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
1	0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8
2	4	1	E	8	D	6	2	B	F	C	9	7	3	A	5	0
3	F	C	8	2	4	9	1	7	5	B	3	E	A	0	6	D

TABLE 2.3 – Boîtes-S de la fonction F du DES.

point une machine coûtant environ 10 000 dollars nommée COPACOBANA RIVYERA qui met, en moyenne, 6 jours et 10 heures à casser le DES et 13 jours dans le pire cas.

Suite à cela, le DES a été remplacé comme standard par l'AES *Advanced Encryption Standard* mais reste très utilisé sous la forme du Triple-DES qui consiste à chiffrer un message trois fois avec deux clefs différentes :

$$\texttt{Triple} - \texttt{DES}_{k1,k2} = \texttt{DES}_{k1} \circ \texttt{DES}_{k2}^{-1} \circ \texttt{DES}_{k1}.$$

$PC1(X)_i$	55	54	53	52	51	50	49	48
X_i	7	15	23	31	39	47	55	63
$PC1(X)_i$	47	46	45	44	43	42	41	40
X_i	6	14	22	30	38	46	54	62
$PC1(X)_i$	39	38	37	36	35	34	33	32
X_i	5	13	21	29	37	45	53	61
$PC1(X)_i$	31	30	29	28	27	26	25	24
X_i	4	12	20	28	1	9	17	25
$PC1(X)_i$	23	22	21	20	19	18	17	16
X_i	33	41	49	57	2	10	18	26
$PC1(X)_i$	15	14	13	12	11	10	9	8
X_i	34	42	50	58	3	11	19	27
$PC1(X)_i$	7	6	5	4	3	2	1	0
X_i	35	43	51	59	36	44	52	60

TABLE 2.4 – Choix des bits de la clef maître pour le DES.

Décalage(s)	1	2
Tour	$\{1, 2, 9, 16\}$	$\{3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15\}$

TABLE 2.5 – Décalage lors de la rotation dans le cadencement de clef du DES.

Cela est essentiellement dû à des questions pratiques car ainsi, il n'y a pas besoin de reprogrammer un algorithme de chiffrement. Remarquons que si l'on utilise le triple DES plutôt qu'un Double-DES, c'est pour la simple et bonne raison qu'un tel algorithme est aussi faible que le DES du fait de la possibilité d'effectuer une attaque par le milieu. Une seconde remarque est que si l'on utilise l'inverse du DES pour la seconde passe, c'est simplement pour que le Triple-DES soit une généralisation du DES. En effet, $\texttt{Triple} - \texttt{DES}_{k,k} = \texttt{DES}_k$.

2.2 PRESENT

Le chiffrement PRESENT est un chiffrement par blocs qui a été proposé en 2007 à la conférence *CHES* par Bogdanov *et al.* [BKL+07]. Sa simplicité et ses performances ont fait de lui la star des chiffrements dits « légers ». Il est constitué de 31 tours et agit sur des messages de 64 bits. La clef peut, au choix, être de 80 ou 128 bits.

PRESENT est un réseau de substitution permutation comme présenté dans la sous-section 1.4.1 où la sous-clef est additionnée (exclusivement) avec l'état qui passe ensuite dans une couche de boîtes-S avant d'être modifié par une permutation bits à bits. À la fin du dernier tour du chiffrement, on additionne une sous-clef supplémentaire (on verra que cela permet, entre autre, d'éviter à l'attaquant de gagner un tour lors d'une attaque sur le dernier tour).

2.2.1 PRESENT

La couche de boîtes-S est en fait constituée de la même boîte dupliquée 16 fois (et donc cette boîte agit sur 4 bits). Cette boîte-S est donnée par la tableau 2.7 où les valeurs sont données en hexadécimal.

$PC2(X)_i$	47	46	45	44	43	42	41	40	39	38	37	36
X_i	42	39	45	32	55	51	53	28	41	50	35	46
$PC2(X)_i$	35	34	33	32	31	30	29	28	27	26	25	24
X_i	33	37	44	52	30	48	40	49	29	36	43	54
$PC2(X)_i$	23	22	21	20	19	18	17	16	15	14	13	12
X_i	15	4	25	19	9	1	26	16	5	11	23	8
$PC2(X)_i$	11	10	9	8	7	6	5	4	3	2	1	0
X_i	12	7	17	0	12	3	10	14	16	20	27	24

TABLE 2.6 – Choix des bits de la sous-clef pour le DES.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

TABLE 2.7 – Boîte-S du chiffrement PRESENT.

Quant à la permutation P, on l'obtient de la façon suivante :

$$P(\boldsymbol{x})_i = \boldsymbol{x}_{4 \cdot i \bmod 63}.$$

Un tour de PRESENT est représenté par la figure 2.2.

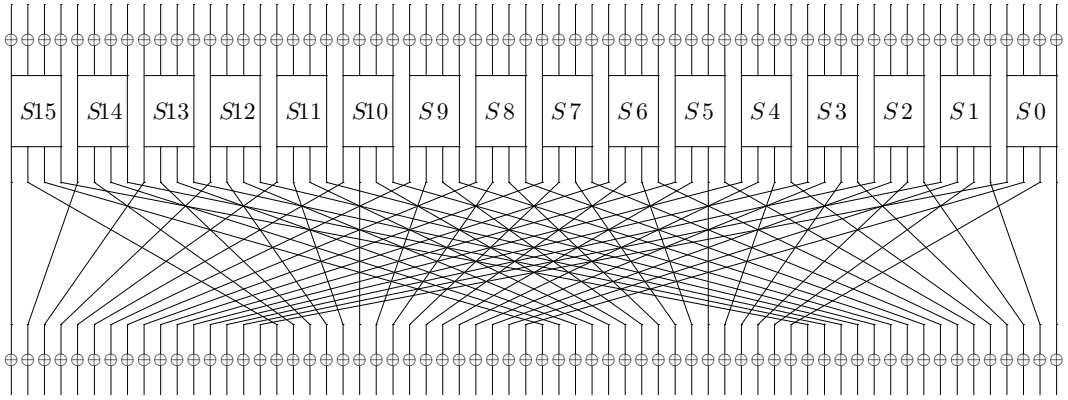

FIGURE 2.2 – Un tour de PRESENT.

Reste maintenant à regarder l'algorithme de cadencement de clefs. L'on va s'intéresser à la version 80 bits de cet algorithme, la version 128 bits étant similaire. La clef maître est représentée comme $\boldsymbol{k} = [k_{79}k_{78}, \dots k_0]$. La sous-clef de 64 bits pour le tour r est extraite de l'état en prenant les 64 positions les plus à gauche : $\boldsymbol{k}_r \stackrel{\text{def}}{=} [k_{79}k_{78} \dots k_{16}]$. L'extraction est similaire pour les clefs de 128 bits. L'état est mis à jour comme suit.

1. $[k_{79}k_{78} \dots k_1k_0] = [k_{18}k_{17} \dots k_{20}k_{19}]$;
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$;
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus r$.

Cela revient à effectuer une rotation de 61 positions vers la gauche puis de passer les 4 bits de poids fort dans la boîte-S puis d'ajouter le numéro du tour aux bits numérotés de 15 à 19. La seule différence pour 128 bits de clef est que plutôt que de passer les 4 bits de gauche dans une boîte-S, on en fait passer 8 dans deux boîtes.

2.2.2 SMALLPRESENT-[s]

Afin de pouvoir effectuer des expériences nécessitant de chiffrer tous les messages avec toutes les clefs possibles, Gregor Leander a proposé une version modulable de PRESENT nommée SMALLPRESENT-[s] [Lea10]. Le nombre $s \geq 2$ correspond aux nombres de boîtes-S du chiffrement et donc SMALLPRESENT-[s] travaille sur des messages de taille $4\,s$ (on a donc SMALLPRESENT-[16]=PRESENT). Les deux seules choses à modifier pour ces versions réduites sont la permutation et le cadencement de clefs. Pour la permutation, on voit qu'il est facile d'adapter sa définition à un nombre s quelconque de boîtes-S. On prend

$$P(\boldsymbol{x})_i = \boldsymbol{x}_{4\cdot i \bmod 4\,s-1}.$$

Pour ce qui est du cadencement de clefs, il a été proposé de garder le même que pour la version complète de PRESENT (à la différence que l'on extrait les $4\,s$ bits les plus à gauche au lieu de 64). Le problème est que dans ce cas, le nombre de tours avant que chaque bit de la clef maître soit intervenu est beaucoup trop grand et on risque d'éviter certains phénomènes dus au fait que les bits des sous-clefs ne sont pas indépendants les uns des autres.

C'est pourquoi dans [BG10] nous proposons d'étudier trois algorithmes de cadencement de clef.

1. Celui proposé dans [Lea10] qui, pour une petite variante et un petit nombre de tours s'apparente à avoir des sous-clefs indépendantes.
2. À l'autre bout du spectre, un algorithme prenant une clef maîtresse de $4\,s$ bits qui sera utilisée telle quelle en tant que sous-clef pour chacun des tours.
3. Une version qui nous paraît raisonnable vu qu'elle consiste à reprendre l'algorithme de cadencement de clefs de base en l'adaptant à la taille du chiffrement.

C'est ce dernier algorithme qui est présenté ci-après. La clef maître est représentée comme $\boldsymbol{k} = [k_{5s-1}k_{5s-2}, \ldots k_0]$. La sous-clef de $4\,s$ bits pour le tour r est extraite de l'état en prenant les $4\,s$ positions les plus à gauche : $\boldsymbol{k}_r \stackrel{\text{def}}{=} [k_{5s-1}k_{5s-2} \ldots k_s]$. L'état est mis à jour comme suit.

1. $[k_{5s-1}k_{5s-2} \ldots k_1k_0] = [k_{s+2}k_{s+1} \ldots k_{s+4}k_{s+3}]$;
2. $[k_{5s-1}k_{5s-2}k_{5s-3}k_{5s-4}] = S[k_{5s-1}k_{5s-2}k_{5s-3}k_{5s-4}]$;
3. $[k_{s+3}k_{s+2}k_{s+1}k_sk_{s-1}] = [k_{s+3}k_{s+2}k_{s+1}k_sk_{s-1}] \oplus r$.

Ci-après, la figure 2.3 et la figure 2.4 représentent respectivement un tour de SMALLPRESENT-[4] et de SMALLPRESENT-[8] qui seront utilisés pour étudier certains phénomènes.

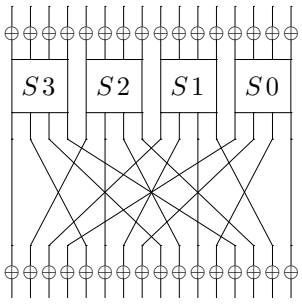

FIGURE 2.3 – Un tour de SMALLPRESENT-[4].

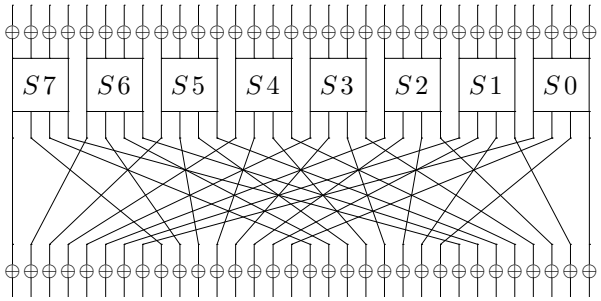

FIGURE 2.4 – Un tour de SMALLPRESENT-[8].

2.2.3 Cryptanalyses

Pour le moment, bien qu'il ait été largement étudié, PRESENT est toujours debout. Nous allons lister ici, en essayant d'être le plus exhaustif possible par rapport à la date de rédaction du document, les attaques répertoriées sur ce chiffrement. Nous verrons que, pour la plupart, elles sont en rapport avec les sujets évoqués tout au long de ce document. Les attaques sont listées dans le même ordre que dans le tableau 2.8 récapitulatif c'est-à-dire par puissance[1] croissante.

[ZRHD08]. Une cryptanalyse *intégrale*, comme proposée dans cet article, consiste à regarder la somme des chiffrés correspondants à des ensembles de clairs ayant certains bits communs fixés. Pour un certain nombre de tours, on s'attend à observer un biais statistique sur la valeur de cette somme. On utilise ce distingueur pour effectuer une attaque. Ici on obtient une cryptanalyse pour 7 tours de la version 128 bits utilisant $2^{24.3}$ clairs et avec une complexité de $2^{100.1}$ déchiffrements partiels en temps et 2^{77} octets en mémoire.

[Wan08]. La cryptanalyse de Meiqin Wang est une cryptanalyse *différentielle* utilisant 24 différentielles sur 14 tours. Nous en reparlerons plus en détails dans les chapitres consacrés à la cryptanalyse différentielle car l'analyse de l'attaque faite dans ce papier est loin d'être satisfaisante. Je vais cependant donner les caractéristique de cette attaque sur 16 tours de la version 80-bits de PRESENT. La complexité en données est 2^{64} (tous les clairs possibles), la complexité en mémoire est 2^{32} et la complexité en temps est 2^{64}.

[ÖVTK09]. Cette attaque se place dans le cadre de *clefs liées* ce qui signifie que l'on suppose que l'on a accès à deux oracles de chiffrements instanciés avec deux clefs ayant une différence Δ_K. On chiffre alors des paires de clairs ayant une différence Δ_P avec ces deux clefs obtenant ainsi un *rectangle* qui donne son nom à l'attaque. Pour le reste, cela se déroule comme n'importe qu'elle cryptanalyse statistique. On retiendra une attaque sur 17 tours de la version 128 bits utilisant 2^{63} clairs, 2^{53} octets et ayant un coût en temps de 2^{104} accès mémoire.

[AC09]. Cette attaque fait partie d'une nouvelle génération d'attaques liant cryptanalyse statistique et cryptanalyse algébrique. Ici, l'attaque présentée est une cryptanalyse *algébrique*

1. Le terme « puissance » étant assez subjectif, les critères utilisés sont le nombre de tours puis le nombre d'échantillons.

différentielle. La cryptanalyse algébrique d'un système est très coûteuse de par le degré des équations. De plus, un tel système est construit avec uniquement un couple clair/chiffré. L'idée est d'essayer de voir comment on pourrait tirer parti de l'utilisation de plus de données. En général, on ajoute des variables pour les états intermédiaires pour réduire le degré au prix d'un nombre croissant de variables. Cependant, le système reste toujours trop long à résoudre. L'idée est de réduire le nombre de variables en supposant qu'une paire de messages a suivi un certain chemin différentiel. On a alors un système bien plus petit (en nombre de variables), mais celui-ci est vrai avec une certaine probabilité et donc il faut un nombre de paires inversement proportionnel à cette probabilité pour espérer former un système correct qui se résoudra alors rapidement. Dans cet article, deux variantes de cette attaque sont proposées. nous ne rentrerons pas, ici, dans le détail de ces variantes qui jouent sur le fait que le temps pour trouver une contradiction dans un système est bien plus court que celui nécessaire pour résoudre un système. Nous retiendrons celle des cryptanalyses qui attaque le plus de tours (19 tours pour 128 bits de clef). Elle nécessite $2^{62.0}$ clairs choisis et la complexité en temps est estimée à 2^{113} cycles processeur.

[Ohk09]. La cryptanalyse *linéaire* de Kenji Ohkuma repose sur le fait que pour certaines clefs, certaines approximations linéaires ont un meilleur biais que pour d'autres (ce qui est dû au *hull effect* et que l'on détaillera section 4.5). Dans son papier, il montre (sans pour autant les donner explicitement) que pour 32% des clefs, appelées *clefs faibles*, le biais d'une approximation linéaire est en fait significativement plus élevé que prévu. Il donne l'exemple d'une approximation sur 28 tours ayant un biais théorique de 2^{-43} et qui s'avère en fait valoir $2^{-39.3}$ pour ces 32% des clefs. En utilisant ce phénomène, il propose une attaque sur 24 tours qui requiert $2^{63.5}$ couples clair/chiffré et a une complexité en temps de 2^{40}. On verra dans la section 4.5 qu'il est loin d'être clair que cette attaque est aussi puissante que cela est annoncé.

[CS09]. Comme dans l'attaque intégrale, on insère un biais statistique en fixant une grande partie des bits en entrée et on regarde le biais obtenu sur la distribution en sortie. Cette attaque est appelée *cryptanalyse statistique par saturation*. Les positions des bits fixés en entrée ne sont pas choisis aléatoirement. En effet, pour avoir un résultat intéressant, il faut éviter que les bits fixés en entrée ne se mélangent trop avec les autres. On utilise en fait des faiblesses dans la permutation du chiffrement. Les bits utilisés dans cet article sont 8 bits en entrée des boîtes 5-6-9 et 10. Cet ensemble de 8 bits est stable par la permutation de tour et donc, à chaque couche de boîtes-S, les 8 bits sortant sont influencés par seulement 8 bits venus de l'extérieur de ce chemin. L'attaque proposée est une attaque sur r tours qui utilise le biais statistique de la sortie après $r-2$ tours. Plusieurs attaques sont proposées et certaines ont fait l'objet d'expérimentations. Nous retiendrons la meilleure attaque proposée qui est sur 24 tours avec une complexité en données de 2^{57} pour une complexité en temps de 2^{57} déchiffrements sur un tour et une utilisation de 2^{32} compteurs.

[NSZW09]. Outre une nouvelle version de cryptanalyse algébrique sur 5 tours, ce papier présente une cryptanalyse *linéaire* basée, tout comme celle d'Ohkuma, sur l'effet *Hull*. Ici l'approche est différente car la complexité de l'attaque est donnée comme une sorte de moyenne sur les clefs ce qui est très discutable. On reparlera plus en détails de cette attaque à la section 4.5. L'attaque proposée sur 26 tours ayant une probabilité de succès très faible, je retiens l'attaque sur 25 tours de la version 128 bits pour une complexité en temps de $2^{98.68}$

utilisant tous les messages possibles et 2^{40} compteurs.

[Cho10]. L'attaque de Joo Yeon Cho est une cryptanalyse *linéaire multidimensionnelle* qui reprend les travaux effectués conjointement avec Miia Hermelin et Kaisa Nyberg [HCN09b, HCN08, HCN09a, HCN09c]. Il attaque 26 tours de PRESENT avec une complexité en données de $2^{64.0}$ clairs/chiffrés et de $2^{72.0}$ chiffrements pour le temps. Il utilise pour cela des approximations sur 24 tours pour une capacité totale de $2^{-55.4}$. C'est, à ce jour, la meilleure attaque sur PRESENT.

	Données	Temps	Mémoire	Version	Tours	Type
[ZRHD08]	$2^{24.3}$ ch.	$2^{100.1}$ dec-2.	$2^{77.0}$	128	8	intégrale
[Wan08]	$2^{64.0}$ ch. [2]	$2^{64.0}$ accès	$2^{32.0}$	80	16	diff. [3]
[ÖVTK09]	$2^{63.0}$ ch.	$2^{104.0}$ accès	$2^{53.0}$	128	17	clefs liées
[AC09]	$2^{62.0}$ ch.	$2^{113.0}$ cycles	n/r	128	19	alg. diff. [4]
[Ohk09]	$2^{63.5}$ co.	$2^{40.0}$ chiff.	$2^{40.0}$	80	24	linéaire
[CS09]	$2^{57.0}$ ch.	$2^{57.0}$ dec-2.	$2^{32.0}$	80	24	stat. sat. [5]
[NSZW09]	$2^{64.0}$ co.	$2^{96.7}$ chiff.	$2^{40.0}$	128	25	linéaire
[Cho10]	$2^{64.0}$ co.	$2^{72.0}$ chiff.	$2^{32.0}$	80	26	lin. multi. [6]

co. : attaque à clair connu. chiff. : chiffrement d'un message.
ch. : attaque à clair choisi. dec-2. : déchiffrement d'un message sur 2 tours.

TABLE 2.8 – Caractéristiques des attaques sur PRESENT

L'ensemble des attaques présentées dans cette sous-section sont résumées dans le tableau 2.8. Toutes ces attaques étant non-adaptatives, il est juste précisé si les clairs sont connus ou choisis. On aura l'occasion de revenir en détail sur certaines de ces cryptanalyses.

2. Ici 2^{64} clairs choisis est équivalent à 2^{64} clairs connus.
3. Cryptanalyse différentielle
4. Cryptanalyse algébrique différentielle
5. Cryptanalyse statistique par saturation
6. Cryptanalyse linéaire multidimensionnelle

Chapitre 3

Quelques éléments de théorie des codes correcteurs d'erreurs

Sommaire

Lors de l'envoi d'un message d'un émetteur à un récepteur, celui-ci est tout d'abord transformé en un signal numérique (*modulation*) puis il est transmis jusqu'au récepteur qui, après *démodulation*, récupère le message émis. La transmission peut s'effectuer sur plusieurs supports : fil de cuivre, clef USB, CD-ROM, air, ... Dans tous les cas, le signal sera modifié durant son transport, on dit alors que le signal est *bruité*. Ce bruit a pour conséquence une perte d'information irréversible. Les codes correcteurs d'erreurs servent à ajouter de la redondance au message afin de pouvoir retrouver toute l'information qui a voulu être transmise et ce, pour un niveau de bruit inférieur à un certain seuil. On peut alors schématiser la communication comme cela est fait dans la figure 3.1.

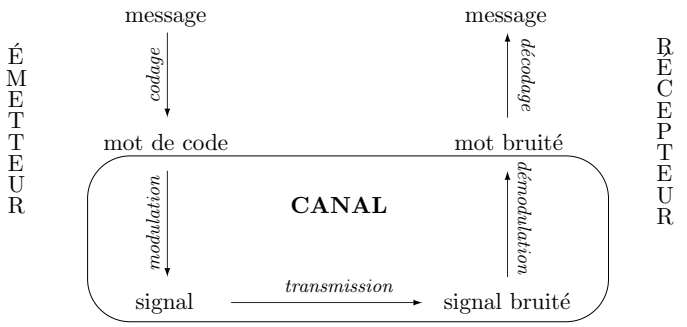

FIGURE 3.1 – Schéma (simplifié) d'une chaîne de transmission.

Dans ce document, on appellera *canal* l'ensemble constitué des opérations de modulation,

transmission et démodulation. Au cours de ce chapitre, nous allons présenter les notions fondamentales de théorie de l'information qui sont indispensables à l'étude théorique des canaux et autres codes correcteurs d'erreurs. Nous nous intéresserons ensuite aux canaux, pièces centrales du dispositif de transmission. Armés de ce bagage théorique, nous introduirons les bases de la théorie des codes correcteurs en insistant sur les codes linéaires binaires qui nous intéresseront particulièrement.

3.1 Notions de théorie de l'information

La *théorie de l'information* permet d'abstraire et de quantifier la notion d'information. Dans ce document, nous nous intéresserons à son utilisation dans le contexte des codes correcteurs et des tests statistiques afin d'appliquer certains résultats à la cryptographie.

3.1.1 Entropies

Commençons par nous intéresser à l'outil principal de la théorie de l'information : l'*entropie*.

Définition 3.1. Soit X une variable aléatoire pouvant prendre des valeurs dans un ensemble $\mathcal{X}$ et ayant une densité $f(x)$. Alors, *l'entropie binaire de* X est :

$$H(X) \stackrel{\text{def}}{=} -\int_{\mathcal{X}} f(x)\log_2(f(x))\,d\,x,$$

où, bien entendu, $\log_2(x)$ est le logarithme en base 2 de x. Dans le cas discret, on a la formule

$$H(X) = -\sum_{x\in\mathcal{X}} \Pr[X=x]\log_2\left(\Pr[X=x]\right).$$

On supposera désormais, sauf mention contraire, que les variables considérées sont continues. On va maintenant définir l'*entropie conditionnelle* d'une variable qui est l'incertitude que l'on a sur une variable Y connaissant une variable X.

Définition 3.2. Soit X et Y deux variables aléatoires pouvant prendre des valeurs dans, respectivement, $\mathcal{X}$ et $\mathcal{Y}$. On note $f(x,y)$ la densité jointe de ces variables et $f(y|x)$ la densité conditionnelle. Alors, *l'entropie conditionnelle binaire de* Y *sachant* X est :

$$H(Y|X) \stackrel{\text{def}}{=} -\int_{\mathcal{X}}\int_{\mathcal{Y}} f(x,y)\log_2\left(f(y|x)\right)\,d\,x\,d\,y.$$

Une notion sous-jacente à celle de l'entropie sur Y connaissant X est la quantité d'information contenue par X sur Y. On va maintenant définir la notion d'information mutuelle.

Définition 3.3. Soit X et Y deux variables aléatoires pouvant prendre des valeurs dans, respectivement, $\mathcal{X}$ et $\mathcal{Y}$. On note $f(x)$ et $f(y)$ les densités respectives de ces deux variables et $f(x,y)$ leur densité jointe. Alors, *l'information mutuelle binaire de* X *et* Y est :

$$I(X;Y) \stackrel{\text{def}}{=} \int_{\mathcal{X}}\int_{\mathcal{Y}} f(x,y)\log_2\left(\frac{f(x,y)}{f(x)\,f(y)}\right)\,dx\,dy.$$

Comme on le voit, cette notion est symétrique : X contient autant d'information sur Y que Y n'en contient sur X. De plus, on a bien $I(X;Y) = H(Y) - H(Y|X)$. Définissons maintenant l'entropie jointe de deux (ou plusieurs) variables.

Définition 3.4. Soit X et Y deux variables aléatoires pouvant prendre des valeurs dans, respectivement, $\mathcal{X}$ et $\mathcal{Y}$. On note $f(x,y)$ la densité jointe de ces variables. Alors, *l'entropie jointe binaire de X et Y* est :

$$H(X,Y) \stackrel{\text{def}}{=} -\int_{\mathcal{X}}\int_{\mathcal{Y}} f(x,y)\log_2\big(f(x,y)\big)\,dx\,dy.$$

L'entropie jointe de trois variables ou plus se déduit de façon évidente de cette formule.

On quantifie ici l'entropie que l'on a sur le couple de variables et, de nouveau, une relation intuitive avec les autres grandeurs présentées apparaît.

Théorème 3.5.

$$H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y).$$

Démonstration : La preuve est simple et purement calculatoire. On rappelle que $f(x)$ est la densité de la variable X et $f(y|x)$ celle de Y conditionnée par X.

$$\begin{aligned}
H(X,Y) &= -\int_{\mathcal{X}}\int_{\mathcal{Y}} f(x,y)\log_2\big(f(x,y)\big)\,dx\,dy \\
&= -\int_{\mathcal{X}}\int_{\mathcal{Y}} f(x,y)\log_2\big(f(x)\,f(y|x)\big)\,dx\,dy \\
&= -\int_{\mathcal{X}}\int_{\mathcal{Y}} f(x,y)\log_2\big(f(x)\big)\,dx\,dy - \int_{\mathcal{X}}\int_{\mathcal{Y}} f(x,y)\log_2\big(f(y|x)\big)\,dx\,dy \\
&= -\int_{\mathcal{X}} f(x)\log_2\big(f(x)\big)\,dx - \int_{\mathcal{X}}\int_{\mathcal{Y}} f(x,y)\log_2\big(f(y|x)\big)\,dx\,dy \\
&= H(X) + H(Y|X).
\end{aligned}$$

Une preuve similaire permet d'obtenir $H(X,Y) = H(Y) + H(X|Y)$. □

Corollaire 3.6. *Soient $X_1,\ldots,X_n$ des variables aléatoires. Alors,*

$$H(X_1,\ldots,X_n) = \sum_{i=1}^{n} H(X_i|X_{i-1},\ldots,X_1).$$

On peut schématiser les relations entre toutes ces grandeurs à l'aide d'un diagramme de Venn. Celui-ci est représenté dans la figure 3.2.

Pour finir, nous allons nous intéresser à une grandeur qui intervient naturellement lors de la quantification de l'éloignement de deux distributions.

Définition 3.7. Soient $f(x)$ et $g(x)$ deux densités définies sur $\mathcal{X}$, alors, l'*entropie relative* ou *divergence de Kullback-Leibler* de ces deux distributions est :

$$D(f||g) \stackrel{\text{def}}{=} \int_{\mathcal{X}} f(x)\,\log\left(\frac{f(x)}{g(x)}\right)\,dx.$$

Cette entropie conditionnelle est abusivement appelée *distance* de Kullback-Leibler bien que celle-ci ne soit pas symétrique et ne vérifie pas l'inégalité triangulaire.

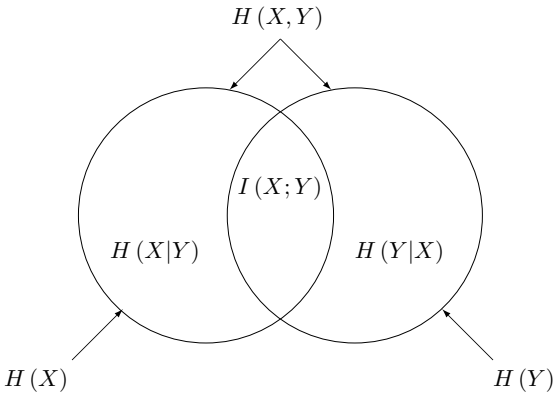

FIGURE 3.2 – Relations entre les grandeurs définies dans la section 3.1

3.1.2 Canaux et capacité

Nous allons ici nous intéresser à la caractérisation d'un canal et à la quantification de sa *capacité* à transmettre de l'information. Nous donnerons en exemple l'application de cette notion de capacité aux cas de deux canaux simples.

Définition 3.8. Un *canal* est un système composé d'un alphabet d'entrée $\mathcal{X}$, un alphabet de sortie $\mathcal{Y}$ et d'une famille de distributions $(p(Y|X = x))_{x\in\mathcal{X}}$ décrivant la façon dont le canal altère le signal.

Si les cardinaux des alphabets sont finis, alors on parlera de *canal discret*. La *matrice de transition* d'un tel canal est la matrice formée par les tables de loi des distributions $p(Y|X = x)$.

Définition 3.9. Soit $\boldsymbol{X} = (\boldsymbol{X}_i)_{1\leq i\leq n}$ une variable aléatoire correspondant à n entrées du canal et $\boldsymbol{Y} = (\boldsymbol{Y}_i)_{1\leq i\leq n}$ celle correspondant à la sortie. On dit d'un canal qu'il est *sans mémoire* si :

$$\Pr\left[\boldsymbol{Y}|\boldsymbol{X}\right] = \prod_{i=1}^{n} \Pr\left[\boldsymbol{Y}_i|\boldsymbol{X}_i\right].$$

Définition 3.10. Soit X la variable aléatoire correspondant à une entrée d'un canal sans mémoire et Y la variable contenant la sortie correspondante. Alors, la *capacité* de ce canal est définie par :

$$C \stackrel{\text{def}}{=} \max_{p(x)} I\left(X;Y\right),$$

où $p(x)$ parcourt toutes les distributions possibles pour X.

Exemple 3.11. L'exemple le plus utilisé pour illustrer cette notion est le canal binaire symétrique (voir figure 3.3). Les alphabets d'entrée et de sortie sont les mêmes $\mathcal{X} = \mathcal{Y} = \{0; 1\}$. La probabilité de changer de symbole au cours de la transmission vaut p. On s'intéresse alors à la capacité de ce canal en fonction de p. On rappelle que $p(x)$ est la distribution des entrées.

$$\begin{aligned} I\left(X;Y\right) &= H\left(Y\right) - H\left(Y|X\right) \\ &= H\left(Y\right) - \sum_x p(x) H\left(Y|X = x\right) \\ &= H\left(Y\right) - \sum_x p(x) H\left(p\right). \end{aligned}$$

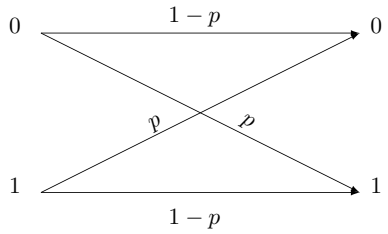

FIGURE 3.3 – Canal binaire symétrique.

La notation, $H(p)$ désigne ici l'entropie d'une variable de Bernoulli de paramètre p i.e. $H(p) \stackrel{\text{def}}{=} p\log_2(p) + (1-p)\log_2(1-p)$. On a donc $I(X;Y) = H(Y) - H(p)$. L'entropie $H(Y)$ dépend de la distribution $p(x)$ de X, son maximum est atteint pour $p(x)$ uniforme et vaut 1. On obtient donc la capacité du canal binaire symétrique de paramètre $p : C = 1 - H(p)$.

Exemple 3.12. Comme son nom l'indique, le *canal binaire à effacements* est un canal consistant à effacer un symbole (on peut penser à des rayures sur un CD-ROM). Il est à noter que l'on connaît l'emplacement des effacements. En effectuant le même calcul que pour le

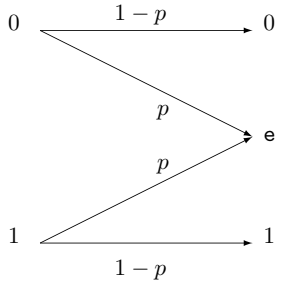

FIGURE 3.4 – Canal binaire à effacements.

canal binaire symétrique, on obtient $I(X,Y) = H(Y) - H(p)$. On utilise alors une astuce pour écrire le maximum de $H(Y)$ en fonction de p (voir [CT91, section 7.1.5]) pour obtenir finalement, $C = 1 - p$.

3.1.3 Canal binaire à bruit blanc additif gaussien

Dans ce document, l'on va s'intéresser aux liens entre le *canal binaire à bruit blanc additif gaussien* et la cryptanalyse linéaire (chapitre 7 et chapitre 8). À la différence des canaux discrets, l'alphabet $\mathcal{Y}$ de ce canal est infini. En effet, l'entrée est discrète car binaire mais le bruit ajouté suit une gaussienne (voir sous-section A.1.6) et donc l'alphabet de sortie est $\mathbb{R}$.

Définition 3.13. Soit $X \in \{0;1\}$ la variable associée à l'entrée d'un canal binaire à bruit blanc additif gaussien. Alors, la sortie Y de ce canal est

$$Y \stackrel{\text{def}}{=} (-1)^X + N\,,$$

avec N qui suit une loi normale $\mathcal{N}(0, \sigma^2)$.

La capacité de ce canal ne se calcule pas facilement, c'est pourquoi nous donnerons juste le résultat que l'on peut trouver, par exemple, dans [RU08, Chapitre 4]. Pour la preuve, le lecteur se référera à cet ouvrage.

Théorème 3.14. *[RU08] La distribution maximisant l'information mutuelle $I(X;Y)$, est la distribution uniforme. Cela permet de calculer la capacité d'un canal binaire à bruit blanc additif gaussien dont le bruit a pour variance σ^2.*

$$\mathrm{Cap}\left(\sigma^2\right) = \int_{-1}^{1} \frac{\sigma}{\sqrt{2\pi}(1-t^2)} e^{-\frac{(1-\sigma^2 \tanh^{-1}(t))^2}{2\sigma^2}} \log_2(1+t)\, dt,$$

que l'on peut aussi écrire

$$\mathrm{Cap}\left(\sigma^2\right) = 1 - \frac{\sigma}{\sqrt{8\pi}} \int_{-\infty}^{\infty} e^{-\frac{(\sigma^2 t-2)^2}{8\sigma^2}} \log_2(1+e^{-t})\, dt,$$

ou bien encore développer en série :

$$\mathrm{Cap}\left(\sigma^2\right) = \frac{1}{\ln(2)} \left(\frac{1}{2\sigma^2} - \frac{1}{4\sigma^4} + \frac{1}{6\sigma^6} + \mathcal{O}\left(\sigma^{-8}\right) \right).$$

3.2 Généralités sur les codes correcteurs d'erreurs

La théorie des codes correcteurs a pour objectif d'optimiser la transmission d'information à travers un canal. Nous allons, dans un premier temps, présenter les principes généraux relatifs aux codes correcteurs d'erreurs puis nous regarderons le cas particulier des codes linéaires qui va nous intéresser tout particulièrement tout au long de ce document.

3.2.1 Codes correcteurs

Définition 3.15. Soit n un entier strictement positif et $\mathcal{X}$ un alphabet. On s'intéresse à la transmission de n symboles de $\mathcal{X}$ et donc à $\mathcal{X}^n$.

Un *code* $(\mathcal{C}, \phi, g)$ de longueur n est constitué :
- d'un sous ensemble de $\mathcal{X}^n$ constitué des n-uplets que l'on s'autorise à transmettre $\mathcal{C} \subset \mathcal{X}^n$;
- d'une règle de « traduction » : le *codage* $\phi : [1; \#\mathcal{C}] \to \mathcal{C} \subset \mathcal{X}^n$;
- et d'un *algorithme de décodage* $g : \mathcal{Y}^n \to \mathcal{C}$.

Un élément de $\mathcal{C}$ est appelé *mot de code*.

Le fait de restreindre l'ensemble des mots que l'on s'autorise à émettre permet de détecter les erreurs voire de les corriger. Une notion importante à ce sujet est celle de *distance minimale* d'un code.

Définition 3.16. Soit d une distance sur $\mathcal{X}^n$. La *distance minimale* $d_{\min}$ d'un code $\mathcal{C} \subset \mathcal{X}^n$ est

$$d_{\min} \stackrel{\text{def}}{=} \min_{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{C}, \boldsymbol{x}_1 \neq \boldsymbol{x}_2} d(\boldsymbol{x}_1, \boldsymbol{x}_2).$$

Évidemment, plus la distance minimale d'un code est grande, plus celui-ci détecte (corrige) d'erreurs. L'objectif de tout code est donc de maximiser cette distance minimale tout en émettant le plus d'information possible lors de la transmission du n-uplet.

Définition 3.17. Le *rendement* d'un code est le rapport entre la quantité d'information émise et la quantité de symboles transmis.

$$R \stackrel{\text{def}}{=} \frac{\log_{\#\mathcal{X}}(\#\mathcal{C})}{n}.$$

Ci-après, deux exemples de codes « naïfs » qui nous serviront d'illustration pour ces notions.

Exemple 3.18. Le code de parité.
Ici $\mathcal{X} = \mathbb{F}_2$. Soit k le nombre de symboles à émettre. Le mot de code correspondant à la séquence d'information $(x_1, \ldots, x_k)$ est $\boldsymbol{c} \stackrel{\text{def}}{=} \left(x_1, \ldots, x_k, \bigoplus_{i=1}^{k} x_i\right)$. Le dernier symbole contient la somme des symboles d'information. La redondance est donc de 1 symbole et le rendement $R = \frac{k}{k+1} \underset{k\to+\infty}{\longrightarrow} 1$. Quant à sa distance minimale, elle est de $\mathrm{d}_{\min} = 2$.

Exemple 3.19. Le code de répétition.
Soit k le nombre de symboles à émettre et a un nombre entier strictement positif (impair de préférence). Le mot de code correspondant à la séquence d'information $(x_1, \ldots, x_k)$ est $\boldsymbol{c} \stackrel{\text{def}}{=} (\underbrace{x_1, \ldots, x_1}_{a \text{ fois}}, \ldots, \underbrace{x_k, \ldots, x_k}_{a \text{ fois}})$. Le rendement de ce code est donc $R = \frac{k}{a \cdot k} = \frac{1}{a}$ et donc peut être rendu arbitrairement proche de 0 par le choix de a. Il a pour distance minimale $\mathrm{d}_{\min} = a$.

Nous allons maintenant nous intéresser à l'autre côté du canal, celui où l'on reçoit le message bruité.

Définition 3.20. Un *algorithme de décodage* g prend en entrée un mot bruité $\boldsymbol{y} \in \mathcal{Y}^n$ et renvoie un mot de code $\boldsymbol{c} \in \mathcal{C} \subset \mathcal{X}^n$ dont il pense que $\boldsymbol{y}$ provient. Il existe deux principales techniques pour déterminer le mot à retourner.

- *Décodage par distance minimale :*
  L'algorithme retourne $\underset{\boldsymbol{c}\in\mathcal{C}}{\operatorname{argmin}}\, \mathrm{d}(\boldsymbol{y}, \boldsymbol{c})$.
- *Décodage au maximum de vraisemblance :*
  L'algorithme retourne $\underset{\boldsymbol{c}\in\mathcal{C}}{\operatorname{argmax}} \Pr[\boldsymbol{Y} = \boldsymbol{y} | \boldsymbol{X} = \boldsymbol{c}]$.

À noter qu'un algorithme de décodage peut renvoyer plusieurs mots de code, on l'appelle alors un algorithme de *décodage en liste*. Ceci peut s'avérer utile dans certains cas.

Lors de la transmission d'un mot de code $\boldsymbol{c} \in \mathcal{C}$, le nombre d'erreurs induites par le canal peut dépasser le nombre d'erreurs corrigées par le code. L'algorithme risque donc de ne pas renvoyer le mot de code émis. On parle alors d'erreur de décodage.

Définition 3.21. La *probabilité maximale d'erreur* d'un code $(\mathcal{C}, \phi, g)$ passant à travers un canal est :

$$P_{\text{err}}(\mathcal{C}) \stackrel{\text{def}}{=} \max_{\boldsymbol{c}\in\mathcal{C}} \Pr[g(\boldsymbol{Y}(\boldsymbol{c})) \neq \boldsymbol{c}],$$

où $\boldsymbol{Y}(\boldsymbol{c})$ est la variable aléatoire correspondant à la sortie du canal pour une entrée $\boldsymbol{c}$.

Le *théorème du codage de canal* donne une borne sur le rendement maximum atteignable par un code pour une capacité donnée d'un canal et une probabilité d'erreur raisonnable.

Théorème 3.22. *Soit C la capacité d'un canal sans mémoire et $(\mathcal{C}_n)_{n\geq 1}$ une suite de codes de longueur n, de rendement R et de probabilité d'erreur maximale $P_{\text{err}}(\mathcal{C}_n)$. Alors,*

1. $\lim_{n\to+\infty} P_{\text{err}}(\mathcal{C}_n) = 0 \implies R \leq C$;
2. *Si* $R < C$, *il existe une suite de codes* $(\mathcal{C}_n)_{n\geq 1}$ *de rendements* $R_n \leq R$ *telle que* $\lim_{n\to+\infty} P_{\text{err}}(\mathcal{C}_n) = 0$.

Une démonstration de ce théorème peut être trouvée dans [CT91, théorème 7.7.1].

3.2.2 Codes correcteurs linéaires

Comme prévu, nous allons maintenant nous intéresser aux *codes linéaires*.

Définition 3.23. Soit $\mathbb{F}$ un corps, un *code linéaire* est un sous-espace vectoriel de $\mathbb{F}^n$. La dimension de ce sous-espace est notée k, on l'appelle *dimension* du code et le rendement d'un tel code est alors $R = \frac{k}{n}$.

Dans ce contexte, la distance utilisée est la distance de Hamming.

Définition 3.24. Soit $\boldsymbol{x} \in \mathbb{F}^n$ et $\boldsymbol{y} \in \mathbb{F}^n$, alors la distance de Hamming de $\boldsymbol{x}$ à $\boldsymbol{y}$ est

$$\mathrm{d}_H(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \#\left\{i,\, \boldsymbol{x}_i \neq \boldsymbol{y}_i\right\}.$$

On note poids de Hamming de $\boldsymbol{x} \in \mathbb{F}^n$

$$\mathrm{w}_H(\boldsymbol{x}) \stackrel{\text{def}}{=} \#\{i,\, \boldsymbol{x}_i \neq 0\} = \mathrm{d}_H(\boldsymbol{x}, 0).$$

On peut noter les propriétés suivantes.

Propriété 3.25.

$$d_{\min} = \min_{\boldsymbol{c}\in\mathcal{C}\{0\}} d(\boldsymbol{c}, 0).$$

Propriété 3.26. *Soit* $\mathcal{C}$ *un code ayant une distance minimale* $d_{\min}$, *alors ce code détecte de façon certaine jusqu'à* $d_{\min} - 1$ *erreurs et en corrige jusqu'à* $\lfloor \frac{d_{\min}-1}{2} \rfloor$.

Regardons maintenant l'application de codage ϕ qui est donc une application linéaire injective de $\mathbb{F}^k$ dans $\mathbb{F}^n$. Cette application définit un code de longueur n et de dimension k : $\mathcal{C} = \mathrm{Im}(\phi)$. Au contraire, pour un code donné, plusieurs applications de codage sont possibles. Comme l'application ϕ est linéaire, on peut donc utiliser de l'algèbre linéaire pour représenter le processus de codage.

Définition 3.27. Soit ϕ une application linéaire injective de $\mathbb{F}^k$ dans $\mathbb{F}^n$ et $\mathcal{C}$ le code associé. La *matrice génératrice* du code $\mathcal{C}$ associée à ϕ est notée G_ϕ, c'est la matrice $k \times n$ représentant l'application linéaire ϕ :

$$\phi(\boldsymbol{x}) = \boldsymbol{x} \cdot G_\phi.$$

Par la suite on identifie la matrice G_ϕ et l'application ϕ et donc l'on omet l'indice de G_ϕ. Si G est une matrice génératrice du code $\mathcal{C}$, alors ses lignes forment une base de ce code et réciproquement, toute matrice dont les lignes forment une base du code est une matrice génératrice de celui-ci. Pour des raisons pratiques que l'on verra plus tard, on travaille souvent sur des matrices génératrices *sous forme systématique*.

Définition 3.28. Une application de codage étant injective, on peut mettre la matrice G associée sous la forme suivante :

$$G = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} G' .$$

On dit alors que la matrice G est *sous forme systématique*.

Pour les codes linéaires, il existe un algorithme de décodage simple et universel : le *décodage par syndrome*. Cet algorithme est à la base de celui que l'on étudiera dans le cas de codes binaires de faible rendement.

Décodage par syndrome.

Le décodage par syndrome est fondé sur le fait qu'il existe un critère simple et rapide qui permet de déterminer si un mot appartient à un code linéaire $\mathcal{C}$ ou non. Ce critère est lié au *code dual de* $\mathcal{C}$.

Définition 3.29. Le code dual d'un code linéaire $\mathcal{C}$ de longueur n et de dimension k est défini comme

$$\mathcal{C}^{\perp} \stackrel{\text{def}}{=} \{\boldsymbol{a} \in \mathbb{F}^n \mid \langle \boldsymbol{a}, \boldsymbol{b} \rangle = 0 \ , \forall \, \boldsymbol{b} \in \mathcal{C}\} .$$

On note ici $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ le produit scalaire usuel de $\mathbb{F}^n$.

Le code dual $\mathcal{C}^{\perp}$ est un code linéaire de longueur n et de dimension $n - k$. Il existe donc des matrices génératrices associées à ce code.

Définition 3.30. Soit $\mathcal{C} \subset \mathbb{F}^n$ un code linéaire et G une matrice génératrice sous forme systématique de ce code.

$$G = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} G' .$$

Alors la matrice

$$H = \begin{pmatrix} & 1 & \\ -{}^t G' & & \ddots \\ & & & 1 \end{pmatrix}$$

est une *matrice de parité* de $\mathcal{C}$ i.e. matrice génératrice du code dual $\mathcal{C}^{\perp}$.

Définissons maintenant ce qu'est le *syndrome* d'un vecteur de $\mathbb{F}^n$.

Définition 3.31. Soit $\boldsymbol{y} \in \mathbb{F}^n$, son syndrome $S_H(\boldsymbol{y}) \in \mathbb{F}^{n-k}$ relatif à la matrice de parité H est défini comme

$$S_H(\boldsymbol{y}) \stackrel{\text{def}}{=} H \, {}^t\boldsymbol{y}.$$

Ce qui va nous intéresser pour l'algorithme de décodage est la propriété suivante du syndrome qui découle de sa définition.

Propriété 3.32. *Soit $\boldsymbol{y} \in \mathbb{F}^n$, et $S_H(\boldsymbol{y}) = H\ {}^t\boldsymbol{y}$ le syndrome de $\boldsymbol{y}$ pour un code $\mathcal{C}$ dont H est une matrice de parité. Alors,*

$$S_H(\boldsymbol{y}) = 0 \Longleftrightarrow \boldsymbol{y} \in \mathcal{C}.$$

Et donc, pour $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$ avec $\boldsymbol{x} \in \mathcal{C}$, $S_H(\boldsymbol{y}) = S_H(\boldsymbol{e})$.

Le *décodage par syndrome* consiste donc à trouver le motif d'erreur $\boldsymbol{e}$ sachant que $S_H(\boldsymbol{e}) = S_H(\boldsymbol{y})$. Il existe un certain nombre de tels motifs d'erreur mais on supposera que c'est le motif de poids minimal qui est le bon.

Algorithme 1 : Décodage par syndrome.

Entrée : $\boldsymbol{y} \in \mathbb{F}^n$ mot bruité reçu;
Sortie : $\boldsymbol{x} = g(\boldsymbol{y}) \in \mathcal{C}$;
si $S_H(\boldsymbol{y}) = 0$ **alors**
  retourner $\boldsymbol{y}$;
sinon
  $w \leftarrow 1$;
  tant que *rien n'est retourné* **faire**
    pour tous les $\boldsymbol{e} \in \mathbb{F}^n$, $\mathrm{w}_H(\boldsymbol{e}) = w$ **faire**
      si $S_H(\boldsymbol{e}) = S_H(\boldsymbol{y})$ **alors**
        retourner $\boldsymbol{y} \oplus \boldsymbol{e}$;
      fin
    fin
    $w \leftarrow w + 1$;
  fin
fin

Codes poinçonnés.

Lors d'un décodage, on peut être amené à travailler sur un code *poinçonné*. Un tel code est dérivé d'un code de longueur n auquel on a retiré certaines positions. Cela peut être utile par exemple pour le décodage par syndrome car plus n est grand plus le nombre de motifs d'erreurs de poids w augmente. Il faut cependant faire attention à conserver la dimension du code k quand on supprime certaines positions.

Définition 3.33. Soit G une matrice génératrice d'un code linéaire $\mathcal{C}$ de longueur n et dimension k et soit $I \subset \{1, \ldots, n\}$, $\#I = k$. On note G_I la restriction de la matrice G à ses colonnes dont l'indice appartient à I. Alors I est un *ensemble d'information* si et seulement si,

$$\mathrm{rang}(G_I) = k.$$

Définition 3.34. Soit G la matrice génératrice d'un code linéaire $\mathcal{C}$ de longueur n et dimension k. Soit $P \subset \{1, \ldots, n\}$ un ensemble d'indices avec $\#P = k + h$ où h est un entier positif. Alors, si P est un ensemble d'information, on définit le *code poinçonné* $\mathcal{C}_P$ comme l'ensemble des restrictions des mots de $\mathcal{C}$ aux $k + h$ positions désignées par P. La matrice G_P de ce code est donc formée des $k + h$ colonnes de G dont les indices appartiennent à P.

Chapitre 4

Cryptanalyse linéaire

Sommaire

Ce chapitre a pour but de faire un tour d'horizon des divers travaux effectués au sujet de la cryptanalyse linéaire. L'idée de base d'utiliser des équations probabilistes pour attaquer un chiffrement a été présentée dans [TCG92]. On peut y voir les prémices de la cryptanalyse

linéaire qui fut d'abord présentée par Mitsuru Matsui à EUROCRYPT en 1993[1] [Mat94b] puis améliorée et expérimentée par ce même Matsui [Mat94a, Mat95]. On présentera ces attaques dans la section 4.1. On s'attardera ensuite sur les améliorations de cette cryptanalyse. Il s'agit principalement de l'utilisation de plusieurs approximations linéaires (section 4.2 et section 4.3). Cependant, on a vu apparaître des travaux qui vont un peu plus en profondeur dans l'analyse de la cryptanalyse linéaire. Par exemple, la « cryptanalyse linéaire avec approximations imbriquées » utilise des approximations non biaisées mais statistiquement liées. Ce travail, présenté dans la section 4.4, découle des travaux de Sean Murphy sur l'indépendance statistique des approximations. La technique présentée permet d'effectuer une cryptanalyse linéaire même si l'on ne connaît pas les correspondances entre clairs et chiffrés. L'autre nouveau type de cryptanalyse linéaire est la cryptanalyse linéaire basée sur le *hull effect*. On détaillera deux de ces attaques dans la section 4.5 qui traite de l'estimation du biais d'une approximation pour une clef fixée. On résumera ensuite, dans la section 4.6, les différentes attaques présentées et leurs complexités et on essaiera d'en faire un comparatif. Enfin, dans la section 4.7 on s'intéressera à la recherche de bonnes approximations linéaires d'un chiffrement.

Principe fondateur et notations. La cryptanalyse linéaire est fondée sur l'existence de bonnes approximations linéaires d'un chiffrement. Cette idée est utilisée par M. Matsui afin d'attaquer le standard de l'époque : le DES (*Data Encryption Standard*) présenté section 2.1.

Définition 4.1. Une *approximation linéaire* est un triplet (π, κ, γ) de $\mathbb{F}_2^s \times \mathbb{F}_2^k \times \mathbb{F}_2^s$ vérifiant

$$\Pr_{\boldsymbol{M},\boldsymbol{K}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \kappa, \boldsymbol{K}\rangle = \langle \gamma, E_{\boldsymbol{K}}(\boldsymbol{M})\rangle\right] = \frac{1}{2} + \varepsilon. \tag{4.1}$$

La notation $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$ est celle du produit scalaire dans les corps de caractéristique 2. La valeur $\varepsilon \in [-0,5; 0,5]$ est appelée *biais* de l'approximation et les vecteurs π, γ et κ sont respectivement appelés *masque d'entrée, de sortie* et *masque de clef*.

La notation $\Pr_{\boldsymbol{M},\boldsymbol{K}}[\cdot]$ indique que cette probabilité dépend des distributions de $\boldsymbol{M}$ et $\boldsymbol{K}$ que l'on suppose ici uniformes. Bien sûr, on ne s'intéresse pas à l'approximation triviale de biais $1/2$ consistant à dire que $0 = 0$.

La cryptanalyse linéaire est une cryptanalyse à clairs connus[2]. Le principe est d'utiliser suffisament de couples clair/chiffré pour détecter le biais et donc retrouver de l'information sur la clef. Cependant, lors d'une cryptanalyse, la clef est fixée. On utilise donc une équation de la forme

$$\Pr_{\boldsymbol{M}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \kappa, \boldsymbol{k}\rangle = \langle \gamma, E_{\boldsymbol{k}}(\boldsymbol{M})\rangle\right] = \frac{1}{2} + \varepsilon_{\boldsymbol{k}}. \tag{4.2}$$

L'hypothèse faite pour simplifier le problème est que le biais est indépendant de la clef utilisée. Cette hypothèse est appelée *hypothèse d'équivalence à clef fixée*.

Hypothèse 4.2. *Hypothèse d'équivalence à clef fixée (linéaire).*
Soit une approximation linéaire (π, κ, γ) d'un chiffrement E. Pour toute clef $\boldsymbol{k} \in \mathbb{F}_2^k$,

$$\Pr_{\boldsymbol{M},\boldsymbol{K}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \kappa, \boldsymbol{K}\rangle = \langle \gamma, E_{\boldsymbol{K}}(\boldsymbol{M})\rangle\right] \approx \Pr_{\boldsymbol{M}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \kappa, \boldsymbol{k}\rangle = \langle \gamma, E_{\boldsymbol{k}}(\boldsymbol{M})\rangle\right].$$

1. Pour l'anecdote, cette conférence a eu lieu à l'hôtel Ullensvang en Norvège où s'est déroulé le workshop WCC en 2009. Je n'ai évidemment pas manqué un tel pèlerinage.

2. On peut cependant en faire une cryptanalyse à chiffrés seuls si l'on a de l'information sur le type de clairs ayant servi à générer les échantillons. Par exemple, Matsui propose une telle attaque si les clairs sont des messages en anglais encodés en ASCII [Mat94b].

On discute, dans la section 4.5, du fait que cette hypothèse n'est pas forcément vérifiée. Cependant, pour certains chiffrements comme le DES, cette hypothèse est raisonnable. Nous allons maintenant voir en détail les différentes façons de procéder.

4.1 Cryptanalyse linéaire simple

4.1.1 Attaque directe (type 1)

La façon la plus directe d'utiliser l'existence d'une bonne approximation linéaire est celle présentée dans [Mat94b] sous le nom d'**Algorithme 1**. Nous ferons référence à ce type de cryptanalyse linéaire comme cryptanalyse *de type 1*. Réécrivons l'équation probabiliste donnée par l'approximation, pour une clef $\boldsymbol{k}$ fixée :

$$\Pr_{\boldsymbol{M}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \gamma, E_{\boldsymbol{k}}(\boldsymbol{M})\rangle = 0\right] = \frac{1}{2} + (-1)^{\langle \kappa, \boldsymbol{k}\rangle}\varepsilon.$$

L'attaquant a à sa disposition N couples clair/chiffré $(\boldsymbol{m}^i, \boldsymbol{c}^i)_{1\leq i\leq N}$ avec, donc, $\boldsymbol{c}^i \stackrel{\text{def}}{=} E_{\boldsymbol{k}^*}(\boldsymbol{m}^i)$. Son but est de retrouver la clef $\boldsymbol{k}^*$ utilisée pour chiffrer ces échantillons. On peut évaluer la partie gauche de l'égalité et calculer le biais empirique que l'on obtient pour cette approximation. On déduit ensuite la valeur de $\langle \kappa, \boldsymbol{k}^*\rangle$ en fonction du signe de ce biais empirique. On utilise un compteur t qui compte le nombre de fois que $\langle \pi, \boldsymbol{m}^i\rangle \oplus \langle \gamma, \boldsymbol{c}^i\rangle$ vaut 0 [3] et on note $\hat{\varepsilon}(t)$ le biais empirique obtenu

$$\hat{\varepsilon}(t) \stackrel{\text{def}}{=} \frac{t}{N} - \frac{1}{2}.$$

Alors, si les biais sont de même signe, on choisira $\langle \kappa, \boldsymbol{k}^*\rangle = 0$ et dans le cas contraire ce sera $\langle \kappa, \boldsymbol{k}^*\rangle = 1$.

Algorithme 2 : Cryptanalyse linéaire de type 1

Entrée : N couples clair/chiffré et une approximation linéaire (π, κ, γ) de biais ε;
Sortie : La valeur devinée de $\langle \kappa, \boldsymbol{k}^*\rangle$;

```
début
    t ⟵ 0;
    pour 1 ≤ i ≤ N faire
        t ⟵ t + ⟨π, m^i⟩ ⊕ ⟨γ, c^i⟩ ⊕ 1;
    fin
    si ε · ε̂(t) > 0 alors
        retourner 0;
    sinon
        retourner 1;
    fin
fin
```

On peut terminer l'attaque en effectuant une recherche exhaustive sur le reste des bits de la clef.

3. On aurait pu choisir 1 pour simplifier la formule pour t mais par souci de cohérence avec les papiers cités, j'ai choisi de garder cette définition.

4.1.2 Attaque sur le dernier tour (type 2)

L'attaque de type 1 a pour inconvénient de ne retrouver qu'un seul bit de clef. C'est pourquoi, dans le même papier [Mat94b], Matsui propose un second type de cryptanalyse linéaire que l'on appelle couramment *attaque sur le dernier tour* et qui a été présentée dans la section 1.5. Cette cryptanalyse sera nommée attaque *de type 2*. Supposons donc que l'on ait une approximation de $r-1$ tours du chiffrement, c'est-à-dire, une équation probabiliste de la forme

$$\Pr_{\boldsymbol{M},\boldsymbol{K}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \kappa, \boldsymbol{K}\rangle = \langle \gamma, F_{\boldsymbol{K}_{r-1}} \circ \cdots \circ F_{\boldsymbol{K}_1}(\boldsymbol{M})\rangle\right] = \frac{1}{2} + \varepsilon.$$

On dispose, de plus, de N couples $(\boldsymbol{m}^i, \boldsymbol{c}^i) = (\boldsymbol{m}^i, E_{\boldsymbol{k}^*}(\boldsymbol{m}^i)) = (\boldsymbol{m}^i, F_{\boldsymbol{k}_r^*} \circ \cdots \circ F_{\boldsymbol{k}_1^*}(\boldsymbol{m}^i))$ afin de retrouver de l'information sur la clef $\boldsymbol{k}^*$. On va, plus précisément, s'intéresser à la sous-clef de tour $\boldsymbol{k}_r^*$. Pour toutes les valeurs possibles $\boldsymbol{k}_r$ de $\boldsymbol{k}_r^*$, on va déchiffrer tous les $\boldsymbol{c}^i$ et calculer les $\langle \pi, \boldsymbol{m}^i\rangle \oplus \langle \gamma, F_{\boldsymbol{k}_r}^{-1}(\boldsymbol{c}^i)\rangle$. On va donc obtenir autant de compteurs que de valeurs possibles de $\boldsymbol{k}_r^*$, notons $t_{\boldsymbol{k}_r}$ ces compteurs

$$t_{\boldsymbol{k}_r} \stackrel{\text{def}}{=} \sum_{i=0}^{N} \langle \pi, \boldsymbol{m}^i\rangle \oplus \langle \gamma, F_{\boldsymbol{k}_r}^{-1}(\boldsymbol{c}^i)\rangle \oplus 1.$$

Pour la bonne valeur on aura $\left|\frac{t_{\boldsymbol{k}_r^*}}{N} - \frac{1}{2}\right| \approx \varepsilon$ et pour les mauvaises valeurs, on s'attend à obtenir $\left|\frac{t_{\boldsymbol{k}_r}}{N} - \frac{1}{2}\right| \approx 0$. Cette hypothèse n'est autre que l'hypothèse de répartition aléatoire par fausse clef que l'on a définie dans la section 1.5. Pour être plus précis, voici l'énoncé de cette hypothèse pour le cas particulier de la cryptanalyse linéaire.

Hypothèse 4.3. *Hypothèse de répartition aléatoire par fausse clef (cryptanalyse linéaire).*
Pour tous $\pi, \gamma, \boldsymbol{k}^*$ et pour $\boldsymbol{k}_r \neq \boldsymbol{k}_r^*$,

$$\Pr_{\boldsymbol{M}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \gamma, F_{\boldsymbol{k}_r}^{-1}(E_{\boldsymbol{k}^*}(\boldsymbol{M}))\rangle = 0\right] = \frac{1}{2}.$$

L'attaque consiste donc à déchiffrer partiellement les chiffrés pour tous les candidats possibles, calculer les compteurs correspondant et renvoyer le candidat correspondant au plus grand biais empirique observé.

Bien évidemment, cette attaque en tant que telle n'est pas forcément efficace vu qu'il faut effectuer le déchiffrement pour toutes les valeurs possibles de $\boldsymbol{k}_r^*$ ce qui peut être aussi coûteux qu'une recherche exhaustive. L'astuce vient du fait que l'on n'a pas besoin de connaître tous les bits de $F_{\boldsymbol{k}_r}^{-1}(\boldsymbol{c}^i)$ pour évaluer le produit scalaire avec γ. En effet, seules les positions correspondant à des 1 dans le masque γ sont nécessaires et donc on peut n'effectuer qu'un déchiffrement partiel du dernier tour. On peut donc diviser l'ensemble des sous-clefs en classes d'équivalence : deux clefs $\boldsymbol{k}_r$ et $\boldsymbol{k}_r'$ sont dans la même classe d'équivalence si pour tout $\boldsymbol{c} \in \mathbb{F}_2^m$, $\langle \gamma, F_{\boldsymbol{k}_r}^{-1}(\boldsymbol{c})\rangle = \langle \gamma, F_{\boldsymbol{k}_r'}^{-1}(\boldsymbol{c})\rangle$. Cela a pour principale implication que le nombre de candidats ne sera pas le nombre de valeurs possibles pour $\boldsymbol{k}_r^*$ mais le nombre de classes d'équivalence qui est égal à 2 à la puissance le nombre de bits de clef intervenant dans le processus de déchiffrement pour obtenir les bits actifs du masque de sortie. Le même concept de classes d'équivalence peut être utilisé sur les chiffrés. En effet, comme tous les bits ne sont pas utilisés pour le déchiffrement partiel, il se peut que deux chiffrés $\boldsymbol{c}_1$ et $\boldsymbol{c}_2$ soient équivalents i.e., pour tout $\boldsymbol{k}_r$, $\langle \gamma, F_{\boldsymbol{k}_r}^{-1}(\boldsymbol{c}_1)\rangle = \langle \gamma, F_{\boldsymbol{k}_r}^{-1}(\boldsymbol{c}_2)\rangle$.

Algorithme 3 : Cryptanalyse linéaire de type 2

Entrée : N couples clair/chiffré et une approximation linéaire (π, κ, γ) de biais ε;
Sortie : La valeur devinée de $\boldsymbol{k}_r^*$;
début
  pour *toutes les valeurs possibles* $\boldsymbol{k}$ *pour* $\boldsymbol{k}_r^*$ **faire**
    $t_{\boldsymbol{k}} \longleftarrow 0$;
    pour $1 \leq i \leq N$ **faire**
      $t_{\boldsymbol{k}} \longleftarrow t_{\boldsymbol{k}} + \langle \pi, \boldsymbol{m}^i \rangle \oplus \langle \gamma, F_{\boldsymbol{k}}^{-1}(\boldsymbol{c}^i) \rangle \oplus 1$;
    fin
  fin
fin
retourner $\boldsymbol{k}_r = \underset{\boldsymbol{k}}{\operatorname{argmax}} \ |\hat{\varepsilon}(t_{\boldsymbol{k}})|$;

Attaque sur le premier tour. On peut aussi décider d'effectuer une attaque sur le premier tour de la même façon que sur le dernier tour, voire même, combiner attaque sur le premier et le dernier tour. Cela permet d'utiliser des approximations linéaires sur un nombre réduit de tours qui auront donc un meilleur biais. On ne s'intéressera pas, dans ce document, à ce genre de cryptanalyse car l'analyse théorique est exactement la même que pour une attaque sur le dernier tour.

4.1.3 Combinaison de ces attaques (type 3)

La véritable attaque présentée par Matsui est en fait une combinaison des deux attaques précédentes. On effectue une attaque de type 2 et on en profite pour retrouver le bit d'information obtenu en regardant le signe de la plus grande déviation absolue. Ce type d'attaque sera appelée cryptanalyse linéaire *de type 3* et correspond à l'**Algorithme 2** dans [Mat94b].

Il est de coutume de noter $\boldsymbol{k}_{\mathrm{I}}$ la clef interne i.e. la clef retrouvée grâce au signe de la déviation (ici un bit) et $\boldsymbol{k}_{\mathrm{O}}$ la clef externe c'est-à-dire celle retrouvée grâce à la valeur absolue de la déviation [4].

4.1.4 Analyse théorique de ces attaques

Les attaques présentées plus haut sont basées sur l'observation de biais empiriques $\hat{\varepsilon}$ extraits des échantillons disponibles. On va donc s'intéresser à la distribution de ces biais pour effectuer l'analyse de ces cryptanalyses. Les compteurs T permettant le calcul des biais sont des sommes de variables de Bernoulli. Le paramètre de ces variables est $1/2 + (-1)^{\langle \kappa, \boldsymbol{K} \rangle}\varepsilon$ avec potentiellement $\varepsilon = 0$ dans le cas des mauvais candidats pour les cryptanalyses de type 2 et 3. En tant que somme de variables de Bernoulli indépendantes, ces compteurs suivent donc une distribution binomiale. Cependant, il est difficile de manipuler cette loi vu les paramètres utilisés en cryptanalyse linéaire. Matsui a donc proposé d'appliquer le théorème central limite qui nous dit que la distribution de $\hat{\varepsilon}(T)$ converge vers une loi normale $\mathcal{N}\left(\pm\varepsilon, \frac{1/4-\varepsilon^2}{N}\right)$. Avant de présenter les analyses existantes de ces attaques, on va répondre à la question suivante : *La loi normale est-elle une bonne approximation de la distribution binomiale dans le cas présent ?*

4. Le I vient « inner » de et le O de « outer »

Algorithme 4 : Cryptanalyse linéaire de type 3

Entrée : N couples clair/chiffré et une approximation linéaire (π, κ, γ) de biais ε;
Sortie : La valeur devinée de $(\boldsymbol{k}_{\mathrm{O}}^*, \boldsymbol{k}_{\mathrm{I}}^*)$;

```
début
    pour toutes les valeurs possibles k pour k*_O faire
        t_k ⟵ 0;
        pour 1 ≤ i ≤ N faire
            t_k ⟵ t_k + ⟨π, m^i⟩ ⊕ ⟨γ, F_k^{-1}(c^i)⟩ ⊕ 1;
        fin
    fin
fin
k_O ⟵ argmax_k |ε̂(t_k)|;
si ε · ε̂(t_{k_O}) > 0 alors
    retourner (k_O, 0);
sinon
    retourner (k_O, 1);
fin
```

Proposition 4.4. *Dans le cas précis de la cryptanalyse linéaire, l'approximation des distributions binomiales des compteurs par une approximation normale est précise.*

Démonstration : On va utiliser le théorème de Berry-Esséen (théorème A.11) afin de borner la vitesse de convergence de la distribution binomiale vers la gaussienne. On définit des variables centrées $\dot{X}_i$ par [5]

$$\dot{X}_i \overset{\text{def}}{=} \begin{cases} \frac{1}{2} - \varepsilon & \text{avec probabilité } \frac{1}{2} + \varepsilon, \\ -\frac{1}{2} - \varepsilon & \text{avec probabilité } \frac{1}{2} - \varepsilon. \end{cases}$$

La variance de $\dot{X}_i$ vaut donc $\sigma^2 = \frac{1}{4} - \varepsilon^2$ et l'on a $\rho = \frac{1}{8} - 2\varepsilon^4$. On peut donc majorer la borne donnée par le théorème A.11 en fonction du biais ε :

$$\begin{aligned} \frac{\rho}{\sigma^3} &= \left[\frac{1}{8} - 2\varepsilon^4\right] \cdot \left[\frac{1}{4} - \varepsilon^2\right]^{-3/2} \\ &= \left[1 - 16\varepsilon^4\right] \cdot \left[1 - 4\,\varepsilon^2\right]^{-3/2} \\ &\leq \left[1 - 16\varepsilon^4\right] \cdot \left[1 + 6\,\varepsilon^2\right] \\ &\leq 1 + 6\varepsilon^2. \end{aligned}$$

Et donc, comme $6\varepsilon^2 \ll 1$ et $N \approx \varepsilon^{-2}$ (cf. chapitre 6) [6], alors

$$C\,\frac{\rho}{\sigma^3\sqrt{N}} = \mathcal{O}\left(\varepsilon\right), \tag{4.3}$$

et donc l'utilisation de la loi normale pour estimer la distribution de T est plus que raisonnable. Pour le cas particulier des compteurs correspondant à de mauvais candidats, on a exactement le même résultat. □

5. Cette définition suppose que $\mathbb{E}\left(T\right) = N\left(\frac{1}{2} + \varepsilon\right)$. Cela revient à inclure la valeur de $\langle \kappa, \boldsymbol{K} \rangle$ dans le signe de ε afin de simplifier les formules mais cela ne change en rien le résultat.

6. On montre, dans le chapitre 6, que pour obtenir une bonne probabilité de succès lors d'une cryptanalyse linéaire simple, $N \approx \varepsilon^{-2}$ et ce, sans utiliser cette approximation normale. On n'est donc pas en train de faire une pétition de principe.

Cryptanalyse de type 1. On a vu que la distribution de $\hat{\varepsilon}$ pouvait être estimée par une distribution normale $\mathcal{N}\left((-1)^{\langle \kappa, \boldsymbol{K} \rangle}\varepsilon, \frac{\frac{1}{4}-\varepsilon^2}{N}\right)$. Comme $\varepsilon \ll 1$, on approxime la variance par $\frac{1}{4N}$. On se place donc dans le modèle suivant.

Modèle 4.5. *(Cryptanalyse linéaire simple de type 1).*
Le biais empirique $\hat{\varepsilon}$ suit une loi normale $\mathcal{N}\left((-1)^{\langle \kappa, \boldsymbol{K} \rangle}\varepsilon, \frac{1}{4N}\right)$.

On a alors le résultat suivant dû à Matsui.

Théorème 4.6. *[Mat94b] La probabilité de succès d'une cryptanalyse linéaire simple de type 1 utilisant une approximation de biais ε et N échantillons est :*

$$P_S = \Phi\left(2\sqrt{N}|\varepsilon|\right).$$

Éléments de preuve : La probabilité de succès est la probabilité que $\hat{\varepsilon}$ soit du même signe que son espérance. Supposons que $(-1)^{\langle \kappa, \boldsymbol{K} \rangle}\varepsilon > 0$, alors, $P_S = \Pr\left[\hat{\varepsilon} \geq 0\right]$ ce que l'on peut écrire $\Pr\left[2\sqrt{N}(\hat{\varepsilon} - (-1)^{\langle \kappa, \boldsymbol{K} \rangle}\varepsilon) \geq 2\sqrt{N}(-(-1)^{\langle \kappa, \boldsymbol{K} \rangle}\varepsilon)\right]$ et donc, $P_S = \Phi(2\sqrt{N}(-1)^{\langle \kappa, \boldsymbol{K} \rangle}\varepsilon)$. Si $(-1)^{\langle \kappa, \boldsymbol{K} \rangle}\varepsilon$ est négatif alors $P_S \approx \Pr\left[\hat{\varepsilon} \leq 0\right] = 1 - \Phi(2\sqrt{N}(-1)^{\langle \kappa, \boldsymbol{K} \rangle}\varepsilon) = \Phi(2\sqrt{N}|(-1)^{\langle \kappa, \boldsymbol{K} \rangle}\varepsilon|)$. □

Le tableau 4.1, que l'on peut trouver dans l'article de Matsui [Mat94b], contient quelques valeurs à titre d'exemple.

N	$1/4 \cdot \varepsilon^{-2}$	$1/2 \cdot \varepsilon^{-2}$	$1 \cdot \varepsilon^{-2}$	$2 \cdot \varepsilon^{-2}$
P_S	0,841	0,921	0,977	0,998

TABLE 4.1 – Succès d'une cryptanalyse linéaire simple de type 1.

Cryptanalyse de types 2 et 3. Ici l'on a un certain nombre 2^{n_k} de biais empiriques $\hat{\varepsilon}_{\boldsymbol{k}} \stackrel{\text{def}}{=} \hat{\varepsilon}(T_{\boldsymbol{k}})$ correspondant aux candidats pour la sous-clef. Il peut être utile de modifier l'attaque présentée précédemment en lui faisant renvoyer une liste $\mathcal{L}$ des ℓ candidats correspondant aux meilleurs compteurs. L'analyse de l'attaque s'intéresse donc au rang de la bonne clef parmi ces candidats. Selon l'hypothèse 4.3 de répartition aléatoire par fausse clef, les compteurs correspondant aux mauvais candidats suivent une loi binomiale $\mathcal{B}(N, 1/2)$. Le compteur $T_{\boldsymbol{k}^*}$ correspondant au bon candidat est lui biaisé. Pour simplifier et sans perte de généralité, on dira qu'il suit une loi binomiale $\mathcal{B}(N, 1/2 + \varepsilon)$ avec $\varepsilon > 0$. On utilise, de nouveau, l'approximation normale et on se place donc dans le modèle suivant.

Modèle 4.7. *(Cryptanalyse linéaire simple de types 2 et 3).*
Une cryptanalyse linéaire simple de type 2 ou 3 a pour but de distinguer un biais empirique $\hat{\varepsilon}_{\boldsymbol{k}^*}$ suivant une loi normale $\mathcal{N}\left(\mu^*, \frac{1}{4N}\right)$ ($\mu^* \stackrel{\text{def}}{=} (-1)^{\langle \kappa, \boldsymbol{k}^* \rangle}\varepsilon$) de $2^{n_k} - 1$ biais $\hat{\varepsilon}_1, \ldots, \hat{\varepsilon}_{2^{n_k}-1}$ indépendants suivant des lois normales $\mathcal{N}\left(0, \frac{1}{4N}\right)$. Pour la cryptanalyse de type 3, il faut, de plus, que le biais $\hat{\varepsilon}_{\boldsymbol{k}^*}$ soit du signe de $(-1)^{\langle \kappa, \boldsymbol{k}^* \rangle}\varepsilon$.

Dans son article [Mat94b], Matsui donne une formule pour la probabilité de succès de l'attaque pour un nombre de candidats gardés $\ell = 1$ (*top ranking*). La formule de Matsui est

de la forme de

$$\int_{-2\sqrt{N}|\varepsilon|}^{\infty} \phi(x) \cdot \prod_{i=1}^{2^{n_k}-1} \Pr\left[|\hat{\varepsilon}_i| < x\right] dx.$$

Cette formule est assez complexe et ne prend en compte que le cas $\ell = 1$. On va donc s'intéresser à deux autres formules respectivement données par Junod et Selçuk.

Théorème 4.8. *[Jun01] La distribution du rang Ψ de la bonne clef parmi les 2^{n_k} candidats lors d'une cryptanalyse linéaire simple de type 2 est*

$$\Pr\left[\Psi \leq \psi\right] = \int_0^{+\infty} g_{\boldsymbol{k}^*}(x) B_{2^{n_k}-\psi,\psi}(G(x))\, dx.$$

Ici, $g_{\boldsymbol{k}^*}(x)$ et $G(x)$ sont, respectivement, la densité de probabilité de la valeur absolue du compteur correspondant à la bonne clef et la fonction de répartition de la valeur absolue d'un compteur correspondant à un mauvais candidat.

$$g_{\boldsymbol{k}^*}(x) = \varphi_{\mu^*,1/4N}\left(x + \frac{1}{2}\right) + \varphi_{\mu^*,1/4N}\left(\frac{1}{2} - x\right),$$

$$G(x) = \Phi_{0,1/4N}\left(x + \frac{1}{2}\right) - \Phi_{0,1/4N}\left(\frac{1}{2} - x\right).$$

Éléments de preuve : Il faut utiliser la théorie des statistiques d'ordre présentée en section A.3. □

Théorème 4.9. *[Sel08] La probabilité de succès d'une attaque linéaire simple de type 3 gardant ℓ candidats parmi 2^{n_k}, utilisant une approximation ayant ε pour biais et N couples clair/chiffré peut être approximée par,*

$$P_S \approx \Phi\left(2\sqrt{N}|\varepsilon| + \Phi^{-1}(1 - 2^{-a-1})\right),$$

où $a \stackrel{\text{def}}{=} n_k - \log_2(\ell)$.

Éléments de preuve : La preuve repose sur trois points.

Le premier est une approximation normale de la loi du quantile d'ordre q d'un ensemble de variables.

Théorème 4.10. *[Rén07, chapitre VIII] Soient $X_1, \ldots, X_n$ des variables i.i.d., G leur fonction de répartition, g leur densité et $X_{(r)}$ leur r-ième statistique d'ordre. On suppose que G est absolument continue et que g est continue non nulle sur un intervalle $[a; b[$. Soit q un réel tel que $0 < G(a) < q < G(b) < 1$ et $i(n)$ une suite telle que $\lim_{n\to\infty} \sqrt{n}\left|\frac{i(n)}{n} - q\right| = 0$.*

Alors, la distribution de $X_{(r)}$ converge vers la distribution normale $\mathcal{N}\left(G^{-1}(q), \frac{q(1-q)}{n\, g(G^{-1}(q))^2}\right)$:

$$\lim_{n\to\infty} \Pr\left[\sqrt{n} g(G^{-1}(q)) \frac{X_{()i(n)} - G^{-1}(q)}{\sqrt{q(1-q)}} < x\right] = \Phi(x).$$

Ce théorème utilisé avec $i(n) \stackrel{\text{def}}{=} \lfloor qn \rfloor + 1$ donne la distribution du quantile d'ordre q des X_i. Une fois cette approximation utilisée, on se retrouve avec une formule pour P_S qui s'exprime uniquement à l'aide de la densité de la loi normale. On utilise alors, et c'est le second point, la stabilité de la gaussienne par addition et par multiplication par un scalaire afin de se ramener à une expression de P_S sous la forme $P_S = \Phi(x)$. Enfin, le troisième point de la démonstration est de négliger la variance de la distribution du quantile d'ordre $1 - \frac{\ell}{2^{n_k}}$ par rapport à $1/4N$. □

Alors que la formule donnée par Junod (théorème 4.8) est exacte, celle de Selçuk est dérivée en utilisant quelques heuristiques qui, étant donnés les paramètres de la cryptanalyse linéaire, sont justifiées. Cette dernière a une forme simple et surtout est facile à calculer. De plus, elle fait apparaître clairement le fait que la complexité en données et la probabilité de succès ne dépendent de ℓ et n_k qu'à travers l'avantage $a \stackrel{\text{def}}{=} n_k - \log_2(\ell)$ et donc que c'est la proportion de candidats gardés qui compte et non leur nombre.

4.2 Cryptanalyse linéaire multiple

La thématique de cette section et de la suivante est l'utilisation de plusieurs approximations linéaires dans le but d'obtenir plus d'information sur la clef à partir d'un même ensemble d'échantillons. Cette amélioration de la cryptanalyse linéaire simple peut être abordée de deux façons différentes. On présentera, dans cette section, l'approche à laquelle on a donné le nom de *cryptanalyse linéaire multiple* qui est la première chronologiquement parlant, puis, on présentera la *cryptanalyse linéaire multidimensionnelle* dans la section 4.3. La principale différence entre ces deux approches est que la première utilise le fait que les approximations considérées sont statistiquement indépendantes (hypothèse qui n'est pas forcément vérifiée) quand la second s'en affranchit.

L'idée d'utiliser plusieurs approximations linéaires est apparue un an seulement après la publication de la première attaque. Dans son second papier [Mat94a], Matsui propose une attaque utilisant deux approximations linéaires et faisant intervenir des ensembles de bits de clef disjoints. La même année, Kaliski et Robshaw proposent une cryptanalyse linéaire multiple utilisant des approximations avec, cette fois-ci, des masques de clef égaux [KR94]. En 2003, Junod et Vaudenay se penchent sur la statistique utilisée par Matsui pour trier les candidats de son attaque à deux approximations [JV03]. Ils proposent une statistique permettant de fusionner l'information recueillie par les deux approximations de façon optimale. Pour cela, ils se basent sur la théorie des tests d'hypothèse vue en section A.2. Il faut attendre l'année 2004 pour avoir une proposition de cryptanalyse linéaire multiple qui utilise n'importe quel ensemble d'approximations [BDCQ04]. Dans ce papier, les auteurs proposent une extension à chacun des deux algorithmes de Matsui (**MK1** pour l'**Algorithme 1** et **MK2** pour l'**Algorithme 2**).

On va présenter les extensions proposées dans [BDCQ04] qui correspondent à l'attaque de type 1 pour l'algorithme MK1 et au type 3 pour le MK2. Les propositions que l'on trouve dans [Mat94a, KR94] en sont des cas particuliers. On présentera ensuite l'analyse faite de ces extensions par Biryukov, De Cannière et Quisquater. Enfin, on discutera quelques aspects de ce travail, discussion qui permettra d'introduire les motivations des travaux présentés dans le chapitre 7 et le chapitre 8. Dans la suite de cette section, on supposera connaître n

approximations linéaires d'un chiffrement E

$$\Pr\left[\langle \pi_j, \boldsymbol{M}\rangle \oplus \langle \gamma_j, E_{\boldsymbol{K}}(\boldsymbol{M})\rangle = 0\right] = \frac{1}{2} + (-1)^{\langle \kappa_j, \boldsymbol{K}\rangle}\varepsilon_j.$$

4.2.1 Attaque de type 1 (MK1)

Pour chacune des approximations on définit un compteur T_j et le biais empirique associé $\hat{\varepsilon}(T_j)$.

$$T_j \stackrel{\text{def}}{=} \sum_{i=1}^{N} \langle \pi_j, \boldsymbol{M}^i\rangle \oplus \langle \gamma_j, \boldsymbol{C}^i\rangle \oplus 1 \quad , \quad \hat{\varepsilon}(T_j) \stackrel{\text{def}}{=} \frac{T_j}{N} - \frac{1}{2}.$$

On a donc un vecteur $\hat{\boldsymbol{\varepsilon}}(\boldsymbol{T}) \stackrel{\text{def}}{=} (\hat{\varepsilon}(T_j))_{1\leq j\leq n}$ de biais empiriques. On suppose que les approximations linéaires sont statistiquement indépendantes[7], i.e. les compteurs correspondants sont indépendants,

Hypothèse 4.11. *Hypothèse d'indépendance statistique des approximations linéaires.*

$$\Pr\left[(T_1, \dots T_n) = (t_1, \dots, t_n) | \boldsymbol{K} = \boldsymbol{k}^*\right] = \prod_{j=1}^{n} \Pr\left[T_j = t_j | \boldsymbol{K} = \boldsymbol{k}^*\right].$$

De plus, on approxime, de nouveau, la distribution binomiale des compteurs par une loi normale. Pour des échantillons obtenus avec une clef $\boldsymbol{k}^*$, on s'attend à ce que le vecteur $\hat{\boldsymbol{\varepsilon}}(\boldsymbol{T})$ soit proche du vecteur $\boldsymbol{\varepsilon}(\boldsymbol{k}^*) \stackrel{\text{def}}{=} ((-1)^{\langle \kappa_1, \boldsymbol{k}^*\rangle}\varepsilon_1, \dots, (-1)^{\langle \kappa_n, \boldsymbol{k}^*\rangle}\varepsilon_n)$. La vraisemblance d'un candidat $\boldsymbol{k}$ connaissant le vecteur empirique $\hat{\boldsymbol{\varepsilon}}(\boldsymbol{T})$ est donc

$$v(\boldsymbol{k}) = \prod_{j=1}^{n} \frac{e^{-2N(\hat{\varepsilon}(T_j) - (-1)^{\langle \kappa_n, \boldsymbol{k}\rangle}\varepsilon_j)^2}}{\sqrt{\pi/2N}}.$$

Le but étant de pouvoir trier les candidats par ordre de vraisemblance, on n'a pas forcément besoin de calculer $v(\boldsymbol{k})$: une fonction croissante de cette valeur suffit à conserver l'ordre et peut être plus simple à calculer. Le produit nous incite, tout d'abord, à regarder le logarithme de cette vraisemblance (*log-vraisemblance*). On se déleste des termes constants et on obtient la statistique suivante

$$L_v(\boldsymbol{k}) \stackrel{\text{def}}{=} -\sum_{j=1}^{n} (\hat{\varepsilon}(T_j) - (-1)^{\langle \kappa_n, \boldsymbol{k}\rangle}\varepsilon_j)^2.$$

Cette statistique dépend essentiellement de la distance euclidienne entre les vecteurs $\boldsymbol{\varepsilon}(\boldsymbol{k})$ et $\hat{\boldsymbol{\varepsilon}}(\boldsymbol{T})$ avec $\boldsymbol{\varepsilon}(\boldsymbol{k}) \stackrel{\text{def}}{=} ((-1)^{\langle \kappa_1, \boldsymbol{k}\rangle}\varepsilon_1, \dots, (-1)^{\langle \kappa_n, \boldsymbol{k}\rangle}\varepsilon_n)$. L'attaque MK1 est donnée par l'algorithme 5.

7. Cette hypothèse est discutable, comme on le verra section 4.4, mais pour certains chiffrements tel le DES, elle est acceptable.

Algorithme 5 : Cryptanalyse linéaire multiple de type 1 (MK1)

Entrée : N couples clair/chiffré et n approximations linéaires $(\pi_j, \kappa_j, \gamma_j)$ de biais ε_j;
Sortie : La valeur devinée de $(\langle \kappa_1, \boldsymbol{k}^* \rangle, \ldots, \langle \kappa_n, \boldsymbol{k}^* \rangle)$;
début
  $\boldsymbol{t} \longleftarrow (0, \ldots, 0)$;
  pour $1 \leq i \leq N$ *et* $1 \leq j \leq n$ **faire**
    $t_j \longleftarrow t_j + \langle \pi_j, \boldsymbol{m}^i \rangle \oplus \langle \gamma_j, \boldsymbol{c}^i \rangle \oplus 1$;
  fin
  retourner $\underset{\boldsymbol{k}}{\operatorname{argmin}} \, ||\boldsymbol{\varepsilon}(\boldsymbol{k}) - \hat{\boldsymbol{\varepsilon}}(\boldsymbol{t})||_2$;
fin

4.2.2 Attaque de type 3 (MK2)

On rappelle que pour la cryptanalyse de type 3, on obtient une série de compteurs pour chaque candidat et que l'hypothèse de répartition aléatoire par fausse clef (hypothèse 4.3) nous dit que pour les mauvais candidats, le biais empirique est nul. Pour chaque approximation et pour chaque candidat $\boldsymbol{k}_\mathrm{O}$ pour la clef externe (clef servant au déchiffrement partiel), on définit un compteur $T_{j,\boldsymbol{k}_\mathrm{O}}$ et le biais empirique associé $\hat{\varepsilon}(T_{j,\boldsymbol{k}_\mathrm{O}})$.

$$T_{j,\boldsymbol{k}_\mathrm{O}} \overset{\text{def}}{=} \sum_{i=1}^{N} \langle \pi_j, \boldsymbol{M}^i \rangle \oplus \langle \gamma_j, F_{\boldsymbol{k}_\mathrm{O}}^{-1}(\boldsymbol{C}^i) \rangle \oplus 1 \quad , \quad \hat{\varepsilon}(T_{j,\boldsymbol{k}_\mathrm{O}}) \overset{\text{def}}{=} \frac{T_{j,\boldsymbol{k}_\mathrm{O}}}{N} - \frac{1}{2}.$$

On a donc un vecteur $\hat{\boldsymbol{\varepsilon}}(\boldsymbol{T}) \overset{\text{def}}{=} (\hat{\varepsilon}(T_{j,\boldsymbol{k}_\mathrm{O}}))_{1 \leq j \leq n, \boldsymbol{k}_\mathrm{O} \neq \boldsymbol{k}_\mathrm{O}^*}$ de biais empiriques. On suppose toujours les approximations statistiquement indépendantes et les compteurs sont toujours supposés suivre des lois normales. Pour des échantillons obtenus avec une clef $\boldsymbol{k}^* = (\boldsymbol{k}_\mathrm{O}^*, \boldsymbol{k}_\mathrm{I}^*)$, on s'attend à ce que le vecteur $\hat{\boldsymbol{\varepsilon}}(\boldsymbol{T})$ soit proche du vecteur

$$\boldsymbol{\varepsilon}(\boldsymbol{k}^*) \overset{\text{def}}{=} (0, \ldots, 0, (-1)^{\langle \kappa_1, \boldsymbol{k}_\mathrm{I}^* \rangle} \varepsilon_1, \ldots, (-1)^{\langle \kappa_n, \boldsymbol{k}_\mathrm{I}^* \rangle} \varepsilon_n, 0, \ldots, 0).$$

Ce vecteur contient des 0 aux indices ne correspondant pas à la bonne clef et les biais attendus pour les compteurs correspondant à $\boldsymbol{k}_\mathrm{O}^*$. On définit donc, pour chaque candidat $\boldsymbol{k} = (\boldsymbol{k}_\mathrm{O}, \boldsymbol{k}_\mathrm{I})$.

$$\boldsymbol{\varepsilon}(\boldsymbol{k}) \overset{\text{def}}{=} (0, \ldots, 0, (-1)^{\langle \kappa_1, \boldsymbol{k}_\mathrm{I} \rangle} \varepsilon_1, \ldots, (-1)^{\langle \kappa_n, \boldsymbol{k}_\mathrm{I} \rangle} \varepsilon_n, 0, \ldots, 0),$$

où les coefficients non nuls se trouvent des indices $\boldsymbol{k}_\mathrm{O} n + 1$ à $\boldsymbol{k}_\mathrm{O}(n+1)$. Pour ce qui est de la vraisemblance d'un candidat, c'est encore la probabilité d'obtenir $\hat{\boldsymbol{\varepsilon}}(\boldsymbol{T})$ sachant que chaque composante est censée être distribuée selon une loi normale de variance $1/4N$ dont les espérances forment le vecteur $\boldsymbol{\varepsilon}(\boldsymbol{k})$. Comme pour le cas de l'attaque de type 1, les variances étant égales, on obtient de nouveau que la distance euclidienne $||\boldsymbol{\varepsilon}(\boldsymbol{k}) - \hat{\boldsymbol{\varepsilon}}(\boldsymbol{T})||_2$ est une statistique suffisante.

4.2.3 Analyse des attaques

L'analyse consiste à calculer le gain de ces attaques Le gain moyen d'une cryptanalyse statistique est

$$\Gamma \overset{\text{def}}{=} -\log_2 \left(\frac{2\bar{\Psi} - 1}{2^{n_k}} \right). \tag{4.4}$$

Algorithme 6 : Cryptanalyse linéaire multiple de type 3 (MK2)

Entrée : N couples clair/chiffré et n approximations linéaires $(\pi_j, \kappa_j, \gamma_j)$ de biais ε_j;
Sortie : La valeur devinée de $\boldsymbol{k}_{\mathrm{O}}^*$ et $\boldsymbol{k}_{\mathrm{I}}^*$;
début
  pour *toutes les valeurs possibles* $\boldsymbol{k}$ *pour* $\boldsymbol{k}_{\mathrm{O}}^*$ **faire**
    $t_{\boldsymbol{k}} \longleftarrow 0$;
    pour $1 \leq i \leq N$ *et* $1 \leq j \leq n$ **faire**
      $t_{j,\boldsymbol{k}} \longleftarrow t_{j,\boldsymbol{k}} + \langle \pi_j, \boldsymbol{m}^i \rangle \oplus \langle \gamma_j, F_{\boldsymbol{k}}^{-1}(\boldsymbol{c}^i) \rangle \oplus 1$;
    fin
  fin
fin
retourner $\underset{\boldsymbol{k}=(\boldsymbol{k}_{\mathrm{O}},\boldsymbol{k}_{\mathrm{I}})}{\operatorname{argmax}} ||\boldsymbol{\varepsilon}(\boldsymbol{k}) - \hat{\boldsymbol{\varepsilon}}(\boldsymbol{t})||_2$;

avec $\bar{\Psi}$ le rang moyen de la bonne clef parmi les 2^{n_k} candidats. Les deux résultats suivants proviennent de [BDCQ04] et donnent les formules pour le gain moyen d'une cryptanalyse linéaire multiple.

Théorème 4.12. *[BDCQ04] Le gain d'une attaque linéaire multiple de type 2 ou 3 utilisant n approximations de biais ε_j et N échantillons est*

$$\Gamma = -\log_2 \left(\frac{2 \sum_{\boldsymbol{k} \neq \boldsymbol{k}^*} \Phi\left(-\sqrt{N}\, ||\boldsymbol{\varepsilon}(\boldsymbol{k}) - \boldsymbol{\varepsilon}(\boldsymbol{k}^*)||_2\right) + 1}{2^{n_k}} \right).$$

Il existe un corollaire à ce théorème qui donne une formule plus simple en estimant la somme sur les clefs. On va juste donner le résultat du corollaire car l'on va discuter de la preuve de celui-ci dans la sous-section 4.2.4.

Corollaire 4.13. *(corollaire du théorème 4.12) Le gain d'une attaque linéaire multiple de type 2 ou 3 utilisant n approximations de biais ε_j et N échantillons peut être approximé par*

$$\Gamma \approx -\log_2 \left(\frac{2(2^{n_k} - 1)\Phi\left(-\sqrt{2\, N \sum_j \varepsilon_j^2}\right) + 1}{2^{n_k}} \right).$$

Éléments de preuve du théorème 4.12. : Le but de la preuve est de trouver une estimation de $\bar{\Psi}$ à injecter dans la formule du gain. La première étape est de montrer que la probabilité qu'un candidat $\boldsymbol{k}$ soit mieux classé que la bonne clef $\boldsymbol{k}^*$ s'exprime grâce à la loi normale : $\Pr\left[L_v(\boldsymbol{k}) > L_v(\boldsymbol{k}^*)\right] = \Phi\left(-\sqrt{N}\, ||\boldsymbol{\varepsilon}(\boldsymbol{k}) - \boldsymbol{\varepsilon}(\boldsymbol{k}^*)||_2\right)$. Si l'on suppose que le rang moyen ne dépend pas de la clef $\boldsymbol{k}^*$ utilisée, alors $\bar{\Psi} = 1 + \sum_{\boldsymbol{k} \neq \boldsymbol{k}^*} \Phi\left(-\sqrt{N}\, ||\boldsymbol{\varepsilon}(\boldsymbol{k}) - \boldsymbol{\varepsilon}(\boldsymbol{k}^*)||_2\right)$. En injectant ceci dans (4.4) on prouve le théorème. □

4.2.4 Discussion

Il y a plusieurs points de discussions possibles sur le travail que l'on vient de présenter et qui est issu de [BDCQ04].

Calcul des vraisemblances.
On a vu que pour ces cryptanalyses multiples, il faut calculer la vraisemblance de tous les candidats afin de trouver le meilleur. Cependant, si le nombre de candidats est proche du nombre de bits de clef, ce qui peut arriver assez facilement si on prend des approximations avec beaucoup de masques de clef différents, alors cela devient plus lent que la recherche exhaustive. Comme il est dit, à juste titre, dans l'article, ce problème de trouver les candidats les plus vraisemblables sans calculer les vraisemblances de tous les candidats est en fait un problème de décodage. Cette remarque nous a poussé à regarder ce qu'il en était et proposer un algorithme efficace pour les paramètres spécifiques à la cryptanalyse linéaire multiple i.e. bruit fort et rendement du code faible. Ce travail est détaillé dans le chapitre 7.

Preuve du corollaire 4.13. La stratégie de la preuve du corollaire 4.13 est assez simple puisqu'il s'agit de faire commuter la somme et la fonction Φ. Dans [BDCQ04], la preuve est faite en effectuant un développement limité de l'espérance. On remarque que le terme d'ordre 1 s'annule et donc la preuve repose sur une approximation de l'espérance par le terme d'ordre 0. Or, comme le dit Sean Murphy [Mur09b], ceci revient à transformer l'inégalité de Jensen en une égalité asymptotique. Étant donné le sens de l'inégalité de Jensen, le gain obtenu dans le corollaire est **supérieur** à celui obtenu par le théorème et donc, de ce point de vue, cette formule surestime la puissance de l'attaque.

Espérance du rang. L'analyse théorique est effectuée en regardant l'espérance du rang de la bonne clef (et donc l'espérance du gain). Or, quand on effectue une cryptanalyse, on travaille dans des zones où le bruit est tellement élevé qu'il se peut que la cryptanalyse se passe très mal (et donc que le rang soit très élevé). Ces événements sont rares mais influencent énormément la moyenne. C'est pourquoi, regarder l'espérance du rang peut amener à des prédictions pessimistes sur la performance d'une attaque. Ce phénomène a déjà été rapidement évoqué plus tôt dans le chapitre à propos des résultats expérimentaux de Junod [Jun01]. Celui-ci utilise son théorème pour estimer l'espérance du rang de la bonne clef et se rend compte que les résultats théoriques sous-estiment ceux obtenus avec les vingt-et-une cryptanalyses effectuées (qui donnent lieu à 42 biais expérimentaux). On propose dans le chapitre 8 une autre approche de l'analyse de la cryptanalyse linéaire, basée sur des outils de théorie de l'information, qui prend en compte ce problème. On verra d'ailleurs que cela permet de retrouver les résultats expérimentaux de Junod et donne une estimation de la complexité en données deux fois moindre que celle annoncée par le corollaire 4.13. La comparaison entre notre formule et celle du théorème 4.12 est bien moins aisée c'est pourquoi nous avons utilisé le corollaire 4.13 qui, de toute façon, surestime le résultat du théorème 4.12 ce qui ne fait que confirmer l'intérêt de notre approche.

Hypothèse de répartition aléatoire par fausse clef et algorithme MK2 Il faut souligner un problème important dans l'utilisation de l'hypothèse 4.3 dans le cas de la cryptanalyse multiple. Prenons un exemple simple pour expliquer le problème. Regardons le vecteur de compteurs correspondant au candidat ayant tous ses bits communs à la bonne sous-clef $\boldsymbol{k}_r^*$ sauf un. Alors, pour un certain nombre d'approximations pour lesquelles ce bit n'intervient pas lors du déchiffrement, les compteurs seront effectivement biaisés ce qui contredit l'hypothèse 4.3.

Une implication de cette remarque est que l'on s'attend à ce que les candidats les mieux

classés soient proches de la bonne sous-clef ce qui pourrait faire l'objet d'une étude plus précise afin d'améliorer l'efficacité d'une telle attaque.

4.3 Cryptanalyse linéaire multidimensionnelle

Avant de rentrer dans le vif du sujet, une petite étape s'impose afin de se familiariser avec les *fonctions booléennes* car la théorie de la cryptanalyse linéaire multidimensionnelle développée par Kaisa Nyberg *et al.* se fonde sur cette théorie.

4.3.1 Fonctions booléennes

Les fonctions booléennes sont fortement liées à la cryptanalyse linéaire comme on le verra au cours de ce chapitre. En effet, une approximation linéaire n'est jamais qu'une fonction booléenne.

Définition 4.14. Une *fonction booléenne* est une fonction de $\mathbb{F}_2^v$ dans $\mathbb{F}_2$. Celle-ci peut être représentée par sa *forme algébrique normale* ou bien sa *table de vérité*. Soit f une fonction booléenne, sa *forme algébrique normale* est l'unique vecteur $\boldsymbol{p} = (p_0, \ldots, p_{2^v-1})$ à coefficients dans $\mathbb{F}_2$ tel que

$$f(\boldsymbol{x}) = P(\boldsymbol{x}) = \sum_{\boldsymbol{u} \in \mathbb{F}_2^v} p_{\boldsymbol{u}} x_0^{u_0} \ldots x_{v-1}^{u_{v-1}},$$

les exposants u_i étant des éléments de $\{0, 1\}$. On définit le *degré* d'une fonction booléenne $d(f)$ comme le plus haut degré des monômes de coefficient non nul de sa forme algébrique normale. La *table de vérité* de f est le vecteur formé des évaluations de cette fonction en tout point i.e. : $\big(f(00\ldots00), f(00\ldots01), \ldots, f(11\ldots10), f(11\ldots11)\big)$.

Une approximation linéaire (π, κ, γ) est donc une fonction booléenne de degré 1

$$f_{(\pi,\kappa,\gamma)}(\boldsymbol{m}, \boldsymbol{k}) \stackrel{\text{def}}{=} \langle \pi, \boldsymbol{m} \rangle \oplus \langle \kappa, \boldsymbol{k} \rangle \oplus \langle \gamma, E_{\boldsymbol{k}}(\boldsymbol{m}) \rangle.$$

La notion qui va nous intéresser est la notion de corrélation entre deux fonctions booléennes.

Définition 4.15. Soient f et g deux fonctions booléennes à v variables, leur corrélation $c(f, g) = c(g, f)$ est définie comme

$$c(f, g) \stackrel{\text{def}}{=} \frac{\#\{\boldsymbol{x} \in \mathbb{F}_2^v,\ f(\boldsymbol{x}) = g(\boldsymbol{x})\} - \#\{\boldsymbol{x} \in \mathbb{F}_2^v,\ f(\boldsymbol{x}) \neq g(\boldsymbol{x})\}}{2^v}.$$

On note $c(f)$ la corrélation de la fonction f à la fonction nulle $c(f) = c(f, \mathbf{0}) = c(\mathbf{0}, f)$.

On va maintenant voir le lien qu'il existe entre la corrélation de fonctions booléennes et la transformée de Walsh.

Définition 4.16. (Transformée de Walsh). Soit f une fonction de $\mathbb{F}_2^v$ à valeurs dans $\mathbb{R}$. Alors, sa transformée de Walsh est notée $\hat{f}$ et est définie sur $\mathbb{F}_2^v$ par

$$\hat{f}(\boldsymbol{u}) \stackrel{\text{def}}{=} \sum_{\boldsymbol{x} \in \mathbb{F}_2^v} (-1)^{\langle \boldsymbol{u}, \boldsymbol{x} \rangle} f(\boldsymbol{x}).$$

Cette transformée détermine la fonction f : on peut reconstruire f avec la formule suivante

$$f(\boldsymbol{x}) = 2^{-v} \sum_{\boldsymbol{u} \in \mathbb{F}_2^v} \hat{f}(\boldsymbol{x})\, (-1)^{\langle \boldsymbol{u}, \boldsymbol{x} \rangle}.$$

Si l'on considère la fonction $F(\boldsymbol{x}) = (-1)^{f(\boldsymbol{x})}$, alors on se rend compte que $c(f) = \widehat{F}(\mathbf{0})$ et que $c(f,g) = \widehat{F \oplus G}(\mathbf{0})$. Regardons maintenant une approximation linéaire comme une fonction booléenne, alors son biais est égal à la moitié de sa corrélation :

$$\Pr\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \kappa, \boldsymbol{K}\rangle \oplus \langle \gamma, \boldsymbol{C}\rangle = 0\right] = \frac{1}{2} + \varepsilon \Leftrightarrow c(f_{(\pi,\kappa,\gamma)}) = 2\varepsilon.$$

Pour la cryptanalyse linéaire multidimensionnelle, on va utiliser des fonctions booléennes vectorielles.

Définition 4.17. Une *fonction booléenne vectorielle* f est une fonction de $\mathbb{F}_2^v$ dans $\mathbb{F}_2^d$. On peut regarder f comme un ensemble de d composantes f_i :

$$f(\boldsymbol{x}) = \big(f_1(\boldsymbol{x}), \ldots, f_d(\boldsymbol{x})\big),$$

chacune des composantes f_i étant une fonction booléenne.

On peut définir la corrélation de chaque combinaison linéaire des composantes.

Définition 4.18. Soit f une fonction booléenne vectorielle à valeurs dans $\mathbb{F}_2^d$ et $\boldsymbol{a} \in \mathbb{F}_2^d$, alors la corrélation de la combinaison linéaire des composantes de f relative à $\boldsymbol{a}$ est

$$c_{\boldsymbol{a}}(f) \stackrel{\text{def}}{=} c\left(\bigoplus_{i,a_i=1} f_i, 0\right).$$

4.3.2 Fonctions vectorielles et cryptanalyse linéaire multidimensionnelle

Ces fonctions booléennes sont utilisées par Hermelin, Nyberg et Cho dans une autre approche de la cryptanalyse linéaire utilisant plusieurs approximations [HN08, HCN09b, HCN08, HCN09a, HCN09c, Cho10, HN10, Her10]. L'objectif principal est de s'affranchir de l'hypothèse d'indépendance statistique des approximations.

Soient d approximations $(\pi_j, \kappa_j, \gamma_j)$ ayant pour biais ε_j, linéairement indépendantes i.e. la famille formée par les vecteurs $(\pi_j||\kappa_j||\gamma_j)$ est libre. On appelle ces approximations *approximations de base*. La différence avec la cryptanalyse linéaire multiple est que l'on ne va pas définir un compteur par approximation mais une unique variable T_i de $\mathbb{F}_2^d$ formée des résultats des évaluations des $\big(\langle \pi_j, \boldsymbol{M}^i\rangle \oplus \langle \gamma_j, \boldsymbol{C}^i\rangle \oplus 1\big)_{1\le j\le d}$ d'où son nom de cryptanalyse multidimensionnelle. On a donc une fonction booléenne vectorielle

$$g : (\boldsymbol{M}) \mapsto \begin{pmatrix} \langle \pi_1, \boldsymbol{M}\rangle \oplus \langle \gamma_1, E_{\boldsymbol{K}}(\boldsymbol{M})\rangle \oplus 1 \\ \vdots \\ \langle \pi_d, \boldsymbol{M}\rangle \oplus \langle \gamma_d, E_{\boldsymbol{K}}(\boldsymbol{M})\rangle \oplus 1 \end{pmatrix}.$$

La distribution de la variable $\boldsymbol{T} \stackrel{\text{def}}{=} g(\boldsymbol{M})$ est donnée dans [NH07]. On peut l'exprimer en fonction des corrélations $c_{\boldsymbol{a}}(g)$.

Lemme 4.19. *[NH07]*

$$\Pr\left[\boldsymbol{T} = \boldsymbol{t}\right] = 2^{-d} \sum_{\boldsymbol{a}\in\mathbb{F}_2^d} (-1)^{\langle \boldsymbol{a},\boldsymbol{t}\rangle} c_{\boldsymbol{a}}(g).$$

On parlera plus loin de l'estimation du biais des approximations mais il est important de se convaincre que la difficulté du calcul des corrélations combinées est la même que celle de l'évaluation des corrélation des approximations base et donc leur utilisation ne pose pas de problèmes particuliers par rapport à une autre cryptanalyse linéaire. Réciproquement, on peut retrouver ces corrélations à partir de la distribution de $\boldsymbol{T}$:

$$c_{\boldsymbol{a}}(g) = \sum_{\boldsymbol{t}\in\mathbb{F}_2^d} (-1)^{\langle \boldsymbol{a},\boldsymbol{t}\rangle} \Pr\left[\boldsymbol{T} = \boldsymbol{t}\right]. \tag{4.5}$$

Dans le reste de la section, on omettra la mention de la fonction g dans $c_{\boldsymbol{a}}(g)$ car on ne se préoccupera que d'une seule fonction g. L'attaquant possède donc N réalisations de la variable $\boldsymbol{T}$ ce qui lui permet de calculer une distribution empirique $\hat{p}$ de cette variable. On connaît la vraie distribution p^* donnée dans le lemme 4.19, il ne reste donc plus qu'à regarder la ou les distributions obtenues pour de mauvais candidats.

4.3.3 Attaque multidimensionnelle de type 1

On a vu que la distribution de $\boldsymbol{T}$ est $p^*_{\boldsymbol{t}} = 2^{-d}\sum_{\boldsymbol{a}\in\mathbb{F}_2^d}(-1)^{\langle \boldsymbol{a},\boldsymbol{t}\rangle} c_{\boldsymbol{a}}$. Les corrélations $c_{\boldsymbol{a}}$ correspondent à des approximations linéaires, on peut donc les exprimer en fonction du biais de ces approximations. Pour cela, nous allons définir la variable $\tilde{\boldsymbol{K}}$ qui, pour une clef $\boldsymbol{K}$ utilisée, contient les d bits relatifs aux masques de clef κ_j :

$$\tilde{\boldsymbol{K}} \stackrel{\text{def}}{=} \left(\langle \kappa_j, \boldsymbol{K}\rangle\right)_{1\le j\le d}.$$

On peut alors écrire les corrélations combinées de g comme

$$c_{\boldsymbol{a}} = (-1)^{\langle \boldsymbol{a},\tilde{\boldsymbol{K}}\rangle} 2\varepsilon_{\boldsymbol{a}},$$

où $\varepsilon_{\boldsymbol{a}}$ est le biais de l'approximation combinée. Ces corrélations dépendent donc de la clef $\tilde{\boldsymbol{K}}$ utilisée ce qui implique la même dépendance pour les $p^*_{\boldsymbol{t}}$. On va donc noter cette dépendance en $\tilde{\boldsymbol{K}}$, on obtient donc la formule suivante pour les probabilités $p^*_{\boldsymbol{t}}(\tilde{\boldsymbol{K}})$:

$$p^*_{\boldsymbol{t}}(\tilde{\boldsymbol{K}}) = 2^{-d+1}\sum_{\boldsymbol{a}\in\mathbb{F}_2^d} (-1)^{\langle \boldsymbol{a},\boldsymbol{t}\oplus\tilde{\boldsymbol{K}}\rangle}\varepsilon_{\boldsymbol{a}}.$$

Sous cette forme, une propriété apparaît clairement.

$$p^*_{\boldsymbol{t}}(\tilde{\boldsymbol{K}}\oplus\boldsymbol{h}) = p^*_{\boldsymbol{t}\oplus\boldsymbol{h}}(\tilde{\boldsymbol{K}}). \tag{4.6}$$

L'implication principale de cette propriété est que si l'on prend l'ensemble des clefs candidates et que l'on regarde les distributions associées, ce sont toutes les mêmes à permutation près. On ne peut donc pas, a priori, utiliser l'approche par test binaire d'hypothèses vu qu'il existe 2^d hypothèses. De plus, le problème n'est pas de distinguer une distribution parmi les 2^d-1 autres mais de savoir à quelle distribution parmi les 2^d correspond la distribution empirique. Deux façons de voir les choses sont présentées dans [HCN08, HCN09c, Her10]. La première est d'utiliser une statistique permettant de quantifier la distance d'une des 2^d distributions à la distribution empirique. La seconde est d'effectuer un test binaire pour chacune des 2^d distributions i.e. tester la vraisemblance de cette distribution par rapport à la distribution uniforme. On imagine alors que la bonne distribution sera celle qui maximise cette vraisemblance.

Méthode de la log-vraisemblance et du χ^2. Ces deux méthodes sont asymptotiquement équivalentes pour distinguer des distributions proches, c'est pourquoi elles sont regroupées au sein d'un même paragraphe. L'idée ici est d'évaluer la distance des distributions à la distribution empirique. La première grandeur qui vient à l'esprit est la divergence de Kullback-Leibler $D\left(\hat{p}\middle|\middle|p^*(\tilde{\boldsymbol{K}})\right)$. Il s'avère que cette divergence est en fait équivalente à la vraisemblance de $p^*(\tilde{\boldsymbol{K}})$ d'où le nom de méthode de log-vraisemblance.

La seconde statistique qui est utilisée dans les tests d'adéquation (quand le quotient de vraisemblance ne peut être calculé par exemple) est la statistique du χ^2

$$\chi^2\left(p^*(\tilde{\boldsymbol{K}}),\hat{p}\right) \stackrel{\text{def}}{=} \sum_{\boldsymbol{t}\in\mathbb{F}_2^d} \frac{\left(\hat{p}_{\boldsymbol{t}} - p^*_{\boldsymbol{t}}(\tilde{\boldsymbol{K}})\right)^2}{p^*_{\boldsymbol{t}}(\tilde{\boldsymbol{K}})}.$$

Dans les deux cas, on trie les candidats par ordre croissant de façon à commencer par ceux correspondant aux distributions les plus « proches » de $\hat{p}$. Si les distributions $p^*(\tilde{\boldsymbol{K}})$ sont proches alors ces deux statistiques sont équivalentes et la complexité en données est de l'ordre de

$$N = \frac{2^{d/2}}{\sum\limits_{\boldsymbol{a}\in\mathbb{F}_2^d\setminus\{\boldsymbol{0}\}} c_{\boldsymbol{a}}^2}.$$

Méthode de la `LLR`. Comme dit précédemment, on n'est pas dans un contexte de test binaire vu qu'il y a 2^d distributions. On va pourtant s'y ramener en calculant les

$$\texttt{LLR}(\hat{p}, p^*(\tilde{\boldsymbol{K}})) \stackrel{\text{def}}{=} \sum_{\boldsymbol{t}\in\mathbb{F}_2^d} N\hat{p}_{\boldsymbol{t}}\ \frac{p^*_{\boldsymbol{t}}(\tilde{\boldsymbol{K}})}{2^{-d}},$$

pour chacune des distributions possibles. On trie alors les candidats par `LLR` décroissant de façon à commencer par ceux pour lesquels la distribution $p^*(\tilde{\boldsymbol{K}})$ se démarque le plus de la distribution uniforme ayant observé $\hat{p}$. La complexité en données est

$$N = \frac{4(\Phi^{-1}(P_S) - \Phi^{-1}(1-2^{-a-1}))^2}{\sum\limits_{\boldsymbol{a}\in\mathbb{F}_2^d\setminus\{\boldsymbol{0}\}} c_{\boldsymbol{a}}^2}.$$

Méthode par convolution. Cette méthode est proposée dans [HN10] et est une adaptation de la technique utilisée par Biryukov et al. au cas particulier où les approximations choisies sont celles obtenues par combinaisons linéaires de d approximations de base. On rappelle que la statistique utilisée est la distance euclidienne entre le vecteur des biais théoriques et le vecteur des biais empiriques (voir sous-section 4.2.1). Dans le cas particulier de la cryptanalyse multidimensionnelle, cette statistique est équivalente à une autre que l'on peut calculer plus efficacement. Le carré de la distance euclidienne qui est calculée pour un candidat $\boldsymbol{k}$ et un vecteur de compteurs $\boldsymbol{t}$ est $||\boldsymbol{\varepsilon}(\boldsymbol{k}) - \hat{\boldsymbol{\varepsilon}}(\boldsymbol{t})||_2^2 = \sum_{\boldsymbol{a}\in\mathbb{F}_2^d}\left((-1)^{\langle \boldsymbol{a},\tilde{\boldsymbol{k}}\rangle}\varepsilon_{\boldsymbol{a}} - \hat{\varepsilon}(t_{\boldsymbol{a}})\right)^2$. On va remplacer les biais par les corrélations et développer le carré

$$||\boldsymbol{\varepsilon}(\boldsymbol{k}) - \hat{\boldsymbol{\varepsilon}}(\boldsymbol{t})||_2^2 = -\frac{1}{2}\sum_{\boldsymbol{a}\in\mathbb{F}_2^d}(-1)^{\langle \boldsymbol{a},\tilde{\boldsymbol{k}}\rangle} c_{\boldsymbol{a}}\hat{c}(t_{\boldsymbol{a}}) + \frac{1}{4}\sum_{\boldsymbol{a}\in\mathbb{F}_2^d} c_{\boldsymbol{a}}^2 + \hat{c}(t_{\boldsymbol{a}})^2.$$

La seconde somme ne dépend pas de $\tilde{\boldsymbol{k}}$ donc on peut se contenter de regarder les candidats qui maximisent $\sum_{\boldsymbol{a}\in\mathbb{F}_2^d}(-1)^{\langle \boldsymbol{a},\tilde{\boldsymbol{k}}\rangle}c_{\boldsymbol{a}}\hat{c}(t_{\boldsymbol{a}})$. En reprenant la formule (4.5) pour exprimer les corrélations en fonction des distributions, on obtient

$$\begin{aligned}\sum_{\boldsymbol{a}\in\mathbb{F}_2^d}(-1)^{\langle \boldsymbol{a},\tilde{\boldsymbol{k}}\rangle}c_{\boldsymbol{a}}\hat{c}(t_{\boldsymbol{a}}) &= \sum_{\boldsymbol{a}\in\mathbb{F}_2^d}(-1)^{\langle \boldsymbol{a},\tilde{\boldsymbol{k}}\rangle}\sum_{\boldsymbol{t},\boldsymbol{t}'\in\mathbb{F}_2^d}(-1)^{\langle \boldsymbol{a},\boldsymbol{t}\oplus\boldsymbol{t}'\rangle}p^*_{\boldsymbol{t}}(\boldsymbol{k}^*)\hat{p}_{\boldsymbol{t}'}\\ &= \sum_{\boldsymbol{t},\boldsymbol{t}'\in\mathbb{F}_2^d}p^*_{\boldsymbol{t}}(\boldsymbol{k}^*)\hat{p}_{\boldsymbol{t}'}\sum_{\boldsymbol{a}\in\mathbb{F}_2^d}(-1)^{\langle \boldsymbol{a},\boldsymbol{t}\oplus\boldsymbol{t}'\oplus\tilde{\boldsymbol{k}}\rangle}.\end{aligned}$$

La somme sur $\boldsymbol{a}$ est nulle si et seulement si $\boldsymbol{t}\oplus\boldsymbol{t}'\oplus\tilde{\boldsymbol{k}}$ est non nul et vaut 2^d sinon. Donc,

$$\sum_{\boldsymbol{a}\in\mathbb{F}_2^d}(-1)^{\langle \boldsymbol{a},\tilde{\boldsymbol{k}}\rangle}c_{\boldsymbol{a}}\hat{c}(t_{\boldsymbol{a}}) = 2^d\sum_{\boldsymbol{t}\in\mathbb{F}_2^d}p^*_{\boldsymbol{t}}(\boldsymbol{k}^*)\hat{p}_{\boldsymbol{t}\oplus\tilde{\boldsymbol{k}}}.$$

Cette somme correspond au $\tilde{\boldsymbol{k}}$-ième composant du produit de convolution $p^*(\boldsymbol{k}^*)*\hat{p}$ et donc ce produit de convolution est une statistique équivalente à la distance euclidienne utilisée par Biryukov (dans le cas spécial de la cryptanalyse multidimensionnelle). Cette remarque permet d'améliorer la complexité en temps de la cryptanalyse car au lieu de calculer la distance $||\boldsymbol{\varepsilon}(\boldsymbol{k})-\hat{\boldsymbol{\varepsilon}}(\boldsymbol{t})||_2^2$ pour chacun des 2^d candidats (ce qui fait au total de l'ordre de 2^{2d} opérations), il suffit de faire un produit de convolution qui s'effectue facilement à l'aide de la transformée de Fourier en $d2^d$ opérations. Il est prouvé dans [HN10] que la complexité en données de cette technique est proche de celle de la méthode `LLR`.

4.3.4 Attaque multidimensionnelle de types 2 et 3

Pour ces deux types d'attaque qui sont des attaques sur le dernier tour, il faut différencier la partie interne des clefs $\boldsymbol{k}_{\mathrm{I}}$ de la partie externe $\boldsymbol{k}_{\mathrm{O}}$. On obtient plusieurs distributions empiriques $\hat{p}(\boldsymbol{k}_{\mathrm{O}})$ correspondant aux différents candidats pour la clef externe. Parmi celles-ci, 2^d-1 suivent une distribution uniforme et une suit une distribution $p^*(\boldsymbol{k}_{\mathrm{I}})$ que l'on ne connaît qu'à une permutation près.

Cryptanalyse de type 2. Dans le cas de la cryptanalyse de type 2, on ne se préoccupe que de retrouver $\boldsymbol{k}^*_{\mathrm{O}}$ et donc on veut juste distinguer la distribution $p^*(\boldsymbol{k}_{\mathrm{I}})$ des distributions aléatoires. On peut donc utiliser des techniques de test d'adéquation (comme vues pour les cryptanalyses de type 1). Il est donc proposé dans [HCN09a] d'utiliser la statistique du χ^2 entre les distributions empiriques et la distribution uniforme et de regarder les candidats dont les distributions associées maximisent cette statistique. Pour un avantage suffisamment élevé et une probabilité de succès proche de 1, on obtient l'estimation suivante pour la complexité en données de l'attaque

$$N = \frac{2\,2^{d/2}\Phi^{-1}(1-2^{-a})+4\Phi^{-2}(2P_s-1)}{\sum\limits_{\boldsymbol{a}\in\mathbb{F}_2^d\setminus\{\boldsymbol{0}\}}c_a^2}.$$

Cryptanalyse de type 3. Pour la cryptanalyse de type 3, il est proposé d'utiliser le test optimal du `LLR`. Comme pour la cryptanalyse multidimensionnelle de type 1, on calcule, pour

chaque couple $(\boldsymbol{k}_\mathrm{I}, \boldsymbol{k}_\mathrm{O})$ la statistique

$$\mathtt{LLR}(\hat{p}(\boldsymbol{k}_\mathrm{O}), p^*(\boldsymbol{k}_\mathrm{I})) \stackrel{\text{def}}{=} \sum_{\boldsymbol{t}\in\mathbb{F}_2^d} N\hat{p}_{\boldsymbol{t}}(\boldsymbol{k}_\mathrm{O})\frac{p^*_{\boldsymbol{t}}(\boldsymbol{k}_\mathrm{I})}{2^{-d}}.$$

On regarde alors les candidats maximisant celle-ci. La complexité en données est alors dérivée en utilisant la convergence des statistiques LLR. Le résultat est qu'elle est de l'ordre de

$$N = \frac{a+d}{\sum\limits_{\boldsymbol{a}\in\mathbb{F}_2^d\setminus\{\boldsymbol{0}\}} c_a^2}.$$

On peut obtenir le même résultat en commençant par calculer le meilleur $\boldsymbol{k}_\mathrm{I}$ pour chaque valeur de $\boldsymbol{K}_\mathrm{O}$ possible en utilisant la méthode par convolution. On calculera ensuite les LLR correspondant à ces couples. On diminue ainsi la complexité en temps d'un facteur égal au nombre de valeurs possibles pour $\boldsymbol{k}_\mathrm{I}$.

Discussion de l'analyse des attaques Cette approche multidimensionnelle est intéressante vu qu'elle permet de dériver des résultats sans supposer l'indépendance statistique des approximations. Cependant, quelques heuristiques supplémentaires sont nécessaires à certaines de ces analyses, heuristiques qui ne sont pas forcément plus réalistes que cette indépendance statistique.

Par exemple, après avoir utilisé la convergence des statistiques LLR, pour les types 1 et 3, les résultats présentés s'obtiennent en utilisant les propriétés des statistiques d'ordre vues en section A.3. Or, celles-ci reposent sur le fait que les variables sont indépendantes ce qui serait le cas pour les statistiques LLR uniquement si chacune était obtenue avec des jeux d'échantillons différents. Il existe des résultats sur les statistiques d'ordre pour des variables dépendantes et il serait intéressant de voir s'il est possible d'obtenir quelque résultat grâce à eux.

Une autre heuristique, qui est assez pessimiste, est utilisée dans l'estimation de l'espérance des statistiques LLR correspondant aux mauvais candidats d'une attaque multidimensionnelle de type 1. Tout d'abord il est supposé que toutes ces statistiques ont la même distribution. Ceci est faux vu que les distributions en question sont des permutations de la bonne distribution, et donc, plus la distribution empirique s'éloigne de la distribution uniforme, plus les distributions des LLR s'éloignent entre elles. Prendre une même distribution pour ces variables est donc problématique, d'autant plus que cette distribution est choisie « au pire cas ». En effet, il est dit que l'espérance de ces statistiques est négative, ce qui est vrai selon le théorème A.18. Pour simplifier l'étude, les auteurs se placent alors dans le pire cas, c'est-à-dire qu'ils prennent cette espérance égale à 0. L'analyse faite sous-estime donc grandement la force de l'attaque (et en particulier pour des avantages élevés). Il serait, de nouveau, intéressant de regarder ce qui est dit en statistiques d'ordre pour des variables suivant une loi normale de même variance mais avec des espérances différentes. Il pourrait être aussi intéressant de regarder les résultats obtenus si l'on suppose que ces statistiques suivent la même loi mais en prenant comme espérance commune la moyenne ou le minimum des espérances. Avec l'analyse en pire cas, on obtiendrait ainsi des bornes et un comportement moyen ce qui permettrait de se rendre compte de l'ordre de grandeur de l'erreur commise.

Choix des approximations. L'approche multidimensionnelle implique une restriction sur les approximations utilisées. Prenons l'exemple du chiffrement PRESENT présenté dans la section 2.2. Si l'on prend les approximations linéaires sur $r-2$ tours qui sont obtenues en chaînant des approximations sur un tour ayant des masques de poids 1, on obtient les meilleures approximations linéaires de ce chiffrement. Cela donne un ensemble d'environ $2^{13.8}$ approximations engendrant un espace de dimension 95. Le problème est qu'il est impossible d'appliquer la cryptanalyse multidimensionnelle avec toutes ces approximations car la complexité serait de $95\,2^{95} \approx 2^{101.6}$ opérations.

Dans le cas de la cryptanalyse multiple, on peut n'utiliser que ces $2^{13.8}$ approximations et obtenir une attaque avec une complexité raisonnable. En fait, si on applique directement l'attaque de Biryukov et al., le risque est qu'il y ait autant de candidats que de clefs et donc le calcul des scores des candidats sera plus long que la recherche exhaustive. On a déjà dit que ce problème pouvait être résolu en calculant seulement les scores des candidats qui semblent intéressants. On présente, dans le chapitre 7, un tel algorithme. On va alors pouvoir tirer parti des $2^{13.8}$ meilleures approximations du chiffrement alors que la technique multidimensionnelle nous aurait limité à un nombre plus faible de ces approximations plus des combinaisons de celles-ci de biais bien moins intéressants.

Il faut cependant relativiser cette remarque car dans son attaque sur PRESENT [Cho10], Cho utilise plusieurs ensembles structurés d'approximations et les combine ce qui lui permet de bien prendre en compte les meilleures approximations tout en traitant des ensembles ayant des dimensions raisonnables.

4.4 Dépendance statistique et approximations imbriquées

Dans cette section on va traiter du problème de l'indépendance statistique des approximations. On a vu que ce problème est le point de départ de l'analyse multidimensionnelle de l'utilisation de plusieurs approximations. Il peut s'avérer que pour certains chiffrements, les approximations soient statistiquement indépendantes asymptotiquement. C'est le cas pour le DES présenté en section 2.1 comme on le verra dans le chapitre 8.

Dans un premier temps, on présentera cette notion d'« indépendance statistique asymptotique » introduite par Sean Murphy dans le cadre de la cryptanalyse linéaire [Mur06]. On présentera ensuite un résultat récent [ER10] qui dérive du travail de Murphy une attaque linéaire qui a la particularité de ne pas nécessiter la connaissance des correspondances entre clairs et chiffrés. On verra, après avoir affiné l'étude de cette attaque, que celle-ci est équivalente à une cryptanalyse linéaire où chaque couple clair/chiffré est remplacé par un couple constitué d'un ensemble de clairs et des chiffrés correspondants.

4.4.1 Indépendance statistique des approximations

Dans son article sur le sujet [Mur06], Sean Murphy commence par donner un petit exemple montrant que bien que linéairement dépendantes, des approximations linéaires peuvent être considérées comme statistiquement indépendantes pour un nombre suffisant d'échantillons. Nous allons reprendre cet exemple en utilisant les notations du présent document afin de préciser ce que l'on entend par *statistiquement indépendantes pour un nombre suffisant d'échantillons.*

Soient (π_{01}, γ_{01}) et (π_{10}, γ_{10}) deux paires de masques. Pour $r, s \in \mathbb{F}_2$, on définit les variables

$$T^i_{ab} \stackrel{\text{def}}{=} \begin{cases} 1 \text{ si} & \langle \pi_{01}, \boldsymbol{M}^i \rangle \oplus \langle \gamma_{01}, \boldsymbol{C}^i \rangle = a \text{ et} \\ & \langle \pi_{10}, \boldsymbol{M}^i \rangle \oplus \langle \gamma_{10}, \boldsymbol{C}^i \rangle = b, \\ 0 & \text{sinon.} \end{cases}$$

Le vecteur $\boldsymbol{T}^i \stackrel{\text{def}}{=} (T^i_{00}, T^i_{01}, T^i_{10}, T^i_{11})$ contient donc un seul et unique 1 correspondant à la valeur des approximations pour la paire i. On peut définir $\boldsymbol{T} \stackrel{\text{def}}{=} \sum_{i=1}^N \boldsymbol{T}^i$ le vecteur contenant, à un facteur N près, la distribution empirique relative à ces deux approximations. L'approche utilisée ici est la même que dans la cryptanalyse multidimensionnelle. Cette distribution est censée dévier de la distribution uniforme, on note ε_{ab} le biais correspondant aux variables T^i_{ab} :

$$\Pr\left[T^i_{ab} = 1\right] = \frac{1}{4} + \varepsilon_{ab}.$$

On va maintenant s'intéresser aux compteurs regardés lors d'une cryptanalyse linéaire multiple. On définit donc deux variables

$$\begin{aligned} U^i_{01} &\stackrel{\text{def}}{=} \langle \pi_{01}, \boldsymbol{M}^i \rangle \oplus \langle \gamma_{01}, \boldsymbol{C}^i \rangle \oplus 1, \\ U^i_{10} &\stackrel{\text{def}}{=} \langle \pi_{10}, \boldsymbol{M}^i \rangle \oplus \langle \gamma_{10}, \boldsymbol{C}^i \rangle \oplus 1. \end{aligned}$$

On a $U^i_{01} = T^i_{00} \oplus T^i_{01}$ et $U^i_{10} = T^i_{00} \oplus T^i_{10}$ et

$$\begin{aligned} \Pr\left[U^i_{01} = 1\right] &= \frac{1}{2} + \varepsilon_{00} + \varepsilon_{01}, \\ \Pr\left[U^i_{10} = 1\right] &= \frac{1}{2} + \varepsilon_{00} + \varepsilon_{10}. \end{aligned}$$

On obtient ainsi les corrélations des variables U^i_{ab} : $c_{ab} \stackrel{\text{def}}{=} 2(\varepsilon_{00} + \varepsilon_{ab})$. Les variables U^i_{ab} suivent donc une loi de Bernoulli d'espérance $\frac{1}{2}(1 + c_{ab})$ et de variance $\frac{1}{4}(1 - c^2_{ab})$. Leur covariance, si l'on néglige les termes de degré 2, vaut quant à elle,

$$\begin{aligned} \operatorname{Cov}\left(U^i_{01}, U^i_{10}\right) &= \mathbb{E}\left(U^i_{01} U^i_{10}\right) - \mathbb{E}\left(U^i_{01}\right) \mathbb{E}\left(U^i_{10}\right) \\ &= \mathbb{E}\left(T^i_{00}\right) - (\frac{1}{2} + \varepsilon_{00} + \varepsilon_{01})(\frac{1}{2} + \varepsilon_{00} + \varepsilon_{10}) \\ &\approx \frac{1}{4} + \varepsilon_{00} - \frac{1}{4} - \varepsilon_{00} - \frac{1}{2}(\varepsilon_{01} + \varepsilon_{10}) \\ &= -\frac{\varepsilon_{01} + \varepsilon_{10}}{2} = \frac{\varepsilon_{00} + \varepsilon_{11}}{2} = \frac{1}{4} c_{11}. \end{aligned}$$

La dernière ligne est obtenue en remarquant que les ε_{ab} sont les déviations d'une distribution par rapport à la distribution uniforme et donc $\sum \varepsilon_{ab} = 0$. Au final, si on approxime les sommes des U^i_{ab} à l'aide de la loi normale, on obtient le théorème suivant.

Théorème 4.20. *Soient $U_{01} \stackrel{\text{def}}{=} \sum_{i=1}^N U^i_{01}$ et $U_{10} \stackrel{\text{def}}{=} \sum_{i=1}^N U^i_{10}$ les compteurs correspondant à deux approximations linéaires. Alors le vecteur*

$$\frac{2}{\sqrt{N}} \begin{pmatrix} U_{01} - \frac{N}{2}(1 + c_{01}) \\ U_{10} - \frac{N}{2}(1 + c_{10}) \end{pmatrix}$$

tend vers un vecteur gaussien d'espérance $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ et de matrice de covariances $\begin{pmatrix} 1 & c_{11} \\ c_{11} & 1 \end{pmatrix}$.

L'enseignement essentiel ici est que dans le contexte de la cryptanalyse linéaire où l'approximation normale des compteurs est raisonnable, le comportement des dépendances entre les compteurs est capturé par la matrice des covariances. Dans le cas d'un nombre plus grand d'approximations, cela signifie qu'il est inutile de regarder les dépendances entre trois compteurs ou plus. Ce résultat est très fort et laisse entrevoir la possibilité de prendre en compte la dépendance des approximations dans une cryptanalyse linéaire multiple. Au passage, on remarque donc que l'hypothèse d'indépendance des approximations peut être vérifiée en estimant les valeurs des corrélations des combinaisons de deux approximations. C'est d'ailleurs ce qui sera fait chapitre 8 afin de justifier cette hypothèse d'indépendance sur le DES.

4.4.2 Un nouveau type d'attaque

L'attaque que nous allons présenter se fonde sur le fait que deux approximations non biaisées peuvent engendrer une approximation biaisée. On a, dans ce cas, $c_{01} = c_{10} = 0$ et c_{11} a une valeur non négligeable. On voit alors que les deux compteurs U^i_{01} et U^i_{10}, bien que non biaisés, sont fortement corrélés vu que leur covariance est c_{11}. La cryptanalyse présentée dans [ER10] utilise de telles approximations non biaisées mais dépendantes sous le nom d'approximations *imbriquées*. L'intérêt de cette attaque est que c'est la seule attaque existante permettant de tirer profit de l'existence d'une bonne approximation linéaire quand on se place dans le modèle suivant.

Modèle 4.21. *Le cryptanalyste a accès à un ensemble de W clairs et des chiffrés correspondants. Cependant, il lui est impossible de savoir à quel clair correspond un chiffré.*

Une solution simple pour appliquer une cryptanalyse linéaire dans ce modèle serait alors de tester toutes les correspondances possibles et garder le résultat obtenu par celle qui donne le biais le plus élevé. Cependant, on en arrive rapidement à un nombre intraitable de correspondances. L'attaque que l'on va maintenant présenter, et qui appartient à la famille des attaques de type 1, réussit à récupérer de l'information sur la clef sans avoir à tester ces correspondances.

On a donc deux approximations linéaires $(\pi_{01}, \kappa, \gamma_{01}, 0)$ et $(\pi_{10}, \kappa, \gamma_{10}, 0)$ imbriquées (le masque de clef doit être le même pour les deux approximations). La combinaison de ces approximations a pour corrélation c_{11}

$$\Pr\left[\langle \pi_{11}, \boldsymbol{M}\rangle \oplus \langle \gamma_{11}, \boldsymbol{C}\rangle = \langle \kappa, \boldsymbol{K}\rangle\right] = \frac{1}{2}(1 + c_{11}).$$

On s'intéresse au couple $\sqrt{W}(\hat{c}_{01}, \hat{c}_{10})$ qui, à un facteur près, correspond aux corrélations empiriques et suit une loi normale bivariée

$$\mathcal{N}\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1 & (-1)^{\langle \kappa_{11}, \boldsymbol{k}^*\rangle} c_{11}\\ (-1)^{\langle \kappa_{11}, \boldsymbol{k}^*\rangle} c_{11} & 1\end{pmatrix}\right).$$

On note Σ_k la matrice de covariance de cette loi pour une valeur k du produit scalaire $\langle \kappa_{11}, \boldsymbol{k}^*\rangle$. On a donc son inverse

$$\Sigma_k^{-1} = \frac{1}{1 - c_{11}^2}\begin{pmatrix}1 & -(-1)^k c_{11}\\ -(-1)^k c_{11} & 1\end{pmatrix}.$$

On peut donc calculer la vraisemblance d'une valeur k pour $\langle \kappa, \boldsymbol{k}^* \rangle$ [8] :

$$v(k) = \frac{1}{2\pi\sqrt{|\Sigma_k^{-1}|}} e^{-\frac{W}{2}(\hat{c}_{01},\hat{c}_{10})\Sigma_k^{-1}\begin{pmatrix}\hat{c}_{01}\\ \hat{c}_{10}\end{pmatrix}}.$$

On veut retrouver la valeur de $\langle \kappa_{11}, \boldsymbol{k}^* \rangle$ et donc, on se retrouve dans le cadre d'un test binaire d'hypothèses. On va donc faire appel à la log-vraisemblance. Le quotient de vraisemblance entre les deux hypothèses $\langle \kappa, \boldsymbol{k}^* \rangle = 0$ et $\langle \kappa, \boldsymbol{k}^* \rangle = 1$ est $-\frac{2\,c_{11}}{1-c_{11}^2} W\, \hat{c}_{01}\hat{c}_{10}$. On définit une statistique équivalente à ce quotient :

$$L_v(k) \stackrel{\text{def}}{=} -\hat{c}_{01}\hat{c}_{10}.$$

Pour des corrélations empiriques de même signe on optera donc pour l'hypothèse $\langle \kappa_{11}, \boldsymbol{k}^* \rangle = 1$ et vice-versa. En ce qui concerne la probabilité de succès de l'attaque, si on suppose que $\langle \kappa, \boldsymbol{k}^* \rangle = 0$, c'est la probabilité que les corrélations soient du même signe et donc, comme ces variables sont centrées,

$$P_S = \Pr\left[\hat{c}_{01}\hat{c}_{10} > 0\right] = 2\Pr\left[\hat{c}_{01} > 0 \text{ et } \hat{c}_{10} > 0\right].$$

Cette probabilité se calcule grâce à une double intégrale dont on peut faire le développement limité autour du point $c_{11} = 0$. On obtient l'estimation suivante de la probabilité de succès :

$$P_S = \frac{1}{2} + \frac{c_{11}}{\pi} + o(c_{11}^3).$$

Remarque. On remarque que la probabilité de succès ne dépend pas explicitement de M. Ceci paraît assez étrange car on a l'intuition que ce n'est pas le cas. Par exemple, si l'on suppose que l'on ait un unique couple, on se retrouve dans le cadre d'une cryptanalyse linéaire classique qui a pour probabilité de succès $\frac{1}{2} + \frac{c_{11}}{2}$. Cela provient du fait que pour W petit, l'approximation gaussienne n'est pas heureuse. C'est pourquoi les auteurs ne proposent l'attaque que pour des valeurs de W supérieures à 64 qui semble être un seuil convenable pour l'utilisation de cette approximation. Nous nous proposons, dans la sous-section 4.4.3, de refaire l'étude de cette attaque sans utiliser l'approximation gaussienne de façon à obtenir un résultat correct pour toute valeur de W. On verra que les résultats trouvés coïncident asymptotiquement avec l'analyse faite dans [ER10].

Amélioration. Supposons maintenant que l'on ait accès à plusieurs groupes de clairs/chiffrés. Notons N le nombre de ces groupes. On peut alors effectuer l'attaque présentée plus haut N fois et décider du bit $\langle \kappa, \boldsymbol{k}^* \rangle$ par vote majoritaire. Chaque groupe a une probabilité $\frac{1}{2} + \frac{c_{11}}{\pi}$ de donner le bon résultat. La probabilité de succès est donc la probabilité qu'une loi binomiale de paramètres N et $\frac{1}{2} + \frac{c_{11}}{\pi}$ soit supérieure à $N/2$. En utilisant une approximation normale pour cette variable binomiale, ce qui est justifié vu que la probabilité correspond aux paramètres de la cryptanalyse linéaire, alors on obtient

$$P_S = \Phi\left(\frac{2c_{11}\sqrt{N}}{\pi}\right).$$

On a de nouveau ce facteur π au lieu du 2 que l'on aurait pour la cryptanalyse linéaire standard.

8. Pour les détails se référer à [ER10].

4.4.3 Analyse plus précise de l'attaque

Comme le titre l'indique, on va maintenant s'atteler à analyser plus finement cette attaque et plus particulièrement à s'affranchir de l'approximation par une gaussienne du comportement des compteurs.

Nous allons reprendre la première attaque proposée dans [ER10] et aborder son analyse d'une façon différente afin de montrer, entre autre, que la probabilité de succès dépend bien de W (contrairement à ce qui est annoncé) et que pour W tendant vers l'infini, la probabilité de succès tend vers $\frac{1}{2} + \frac{\varepsilon}{\pi}$.

Revenons donc au principe de l'attaque. Soit deux approximations imbriquées correspondant à la décomposition de l'approximation linéaire $(\pi, \kappa, \gamma, \varepsilon)$. On définit les variables

$$U_{01}^i \stackrel{\text{def}}{=} \langle \pi, \boldsymbol{M}^i \rangle \oplus 1 \quad \text{et} \quad U_{10}^i \stackrel{\text{def}}{=} \langle \gamma, \boldsymbol{C}^i \rangle \oplus 1.$$

On sait que la variable $U_{11}^i \stackrel{\text{def}}{=} U_{01}^i \oplus U_{10}^i$ vaut $\langle \kappa, \boldsymbol{k}^* \rangle$ avec probabilité $\frac{1}{2} + \varepsilon$. Si on avait accès au compteur $U_{11} \stackrel{\text{def}}{=} \sum_i U_{11}^i$, alors l'attaque optimale consisterait à retourner (comme on l'a vu dans la section 4.1).

$$\langle \kappa, \boldsymbol{k}^* \rangle = \begin{cases} 0 \text{ si } U_{11} < W/2, \\ 1 \text{ si } U_{11} > W/2, \\ 0/1 \text{ avec probabilité } \frac{1}{2} \text{ sinon.} \end{cases}$$

Le 0/1 indique que dans ce cas chacune des deux valeurs pour $\langle \kappa, \boldsymbol{k}^* \rangle$ est aussi probable que l'autre et donc on renvoie une des deux valeurs avec probabilité $\frac{1}{2}$. Notons que cela n'arrive que pour des valeurs paires de W. Or, l'on n'a que les deux compteurs suivants à notre disposition :

$$U_{01} \stackrel{\text{def}}{=} \sum_i U_{01}^i \quad \text{et} \quad U_{10} \stackrel{\text{def}}{=} \sum_i U_{10}^i.$$

L'attaque optimale consiste alors à retourner

$$\langle \kappa, \boldsymbol{k}^* \rangle = \begin{cases} 0 \text{ si } (U_{01} - W/2) \cdot (U_{10} - W/2) > 0, \\ 1 \text{ si } (U_{01} - W/2) \cdot (U_{10} - W/2) < 0, \\ 0/1 \text{ avec probabilité } \frac{1}{2} \text{ sinon.} \end{cases}$$

En effet, notons $\boldsymbol{U}_{01} \stackrel{\text{def}}{=} (U_{01}^1, \ldots, U_{01}^W)$ et $\boldsymbol{U}_{10} \stackrel{\text{def}}{=} (U_{10}^1, \ldots, U_{10}^W)$. Alors les valeurs U_{01} et U_{10} correspondent aux poids de ces vecteurs. Le compteur U_{11} (auquel on n'a pas accès) est égal au poids du vecteur $U_{01} \oplus U_{10}$. On a vu que la décision optimale est de renvoyer $\langle \kappa, \boldsymbol{k}^* \rangle = 0$ pour $U_{11} < W/2$. Or, il est aisé de voir que dans le cas où $(U_{01} - W/2) \cdot (U_{10} - W/2) > 0$, alors le poids du XOR des vecteurs, et donc la valeur de U_{11}, est plus probablement inférieur à $W/2$. Plus précisément,

$$(U_{01} - W/2) \cdot (U_{10} - W/2) > 0 \iff \Pr\left[U_{11} < W/2\right] > \Pr\left[U_{11} > W/2\right].$$

Ceci montre que cette façon de procéder est bien optimale.

Supposons maintenant, sans perte de généralité, que $\langle \kappa, \boldsymbol{k}^* \rangle$ vaut 0. Alors, l'attaque renvoie la bonne valeur avec probabilité 1 si $(U_{01} - W/2) \cdot (U_{10} - W/2) > 0$ et avec probabilité $\frac{1}{2}$ si ce produit est nul. On a donc la probabilité de succès de l'attaque :

$$P_S = \Pr\left[(U_{01} - W/2) \cdot (U_{10} - W/2) > 0\right] + \frac{1}{2} \Pr\left[(U_{01} - W/2) \cdot (U_{10} - W/2) = 0\right].$$

Reste maintenant à calculer ces probabilités. Regardons donc la distribution jointe de ces variables $\Pr[U_{01} = u_{01} \text{ et } U_{10} = u_{10}]$. Pour cela, on rappelle que :

$$\Pr\left[T^i_{ab} = 1\right] = \frac{1}{4} + \varepsilon_{ab} \quad \text{où} \quad T^i_{ab} \stackrel{\text{def}}{=} \begin{cases} 1 & \text{si } \langle \pi, \boldsymbol{M}^i \rangle = a \text{ et } \langle \gamma, \boldsymbol{C}^i \rangle = b, \\ 0 & \text{sinon.} \end{cases}$$

On définit aussi $T_{ab} \stackrel{\text{def}}{=} \sum_i T^i_{ab}$. On sait que $U_{01} = T_{01} + T_{00}$ et $U_{10} = T_{10} + T_{00}$. Pour une valeur t de T_{00},

$$\begin{cases} T_{01} &= U_{01} - t, \\ T_{10} &= U_{10} - t, \\ T_{11} &= W - U_{01} - U_{10} + t. \end{cases}$$

On peut donc écrire

$$\begin{aligned} \Pr[U_{01} = u_{01} \text{ et } U_{10} = u_{10}] &= \sum_{t=0}^{W} \binom{W}{t, u_{10}-t, u_{01}-t, W-u_{10}-u_{01}+t} \left(\frac{1}{4} + \varepsilon_{00}\right)^t \\ &\cdot \left(\frac{1}{4} + \varepsilon_{10}\right)^{u_{10}-t} \left(\frac{1}{4} + \varepsilon_{10}\right)^{u_{10}-t} \left(\frac{1}{4} + \varepsilon_{11}\right)^{W-u_{01}-u_{10}+t} \end{aligned} \tag{4.7}$$

La notation $\binom{W}{a,b,c,d}$ est celle d'un coefficient multinomial défini par

$$\binom{W}{a,b,c,d} \stackrel{\text{def}}{=} \begin{cases} \binom{W}{a}\binom{W-a}{b}\binom{W-a-b}{c} & \text{si } W = a+b+c+d \text{ et } a,b,c \text{ et } d \text{ sont positifs ou nuls,} \\ 0 & \text{sinon.} \end{cases}$$

Pour simplifier cette formule, on va exprimer les ε_{ab} en fonction de ε, a et b.

Cela peut se faire « à la main » mais aussi en utilisant les résultats présentés dans section 4.3 sur les corrélations des fonctions booléennes et la distribution des compteurs : $\varepsilon_{ab} = (-1)^{a \oplus b}\, \varepsilon/2$. Et donc on peut écrire (4.7) comme

$$\begin{aligned} \Pr[U_{01} = u_{01} \text{ et } U_{10} = u_{10}] &= \frac{1}{2^W} \sum_{t=0}^{W} \binom{W}{t, u_{10}-t, u_{01}-t, W-u_{10}-u_{01}+t} \\ &\cdot \left(\frac{1}{2} + \varepsilon\right)^{W-u_{01}-u_{10}+2t} \left(\frac{1}{2} - \varepsilon\right)^{u_{10}+u_{10}-2t}. \end{aligned} \tag{4.8}$$

Si on néglige les puissances de ε supérieures à 1, on obtient alors des probabilités de succès de la forme $\frac{1}{2} + \lambda(W)\varepsilon$. Voici un tableau (4.2) qui donne les premières valeurs de $\lambda(W)$. L'analyse

i	0	1	2	3	4
$\lambda(2i)$	-	0.500	0.562	0.586	0.598
$\lambda(2i+1)$	1.00	0.750	0.703	0.684	0.673

TABLE 4.2 – Valeurs des $\lambda(W)$ intervenant dans la probabilité de succès d'une cryptanalyse linéaire par approximations imbriquées pour de petites valeurs de W.

(asymptotique) de [ER10] donnait une probabilité de succès $\frac{1}{2} + \frac{c_{11}}{\pi}$ ce qui correspond ici à $\frac{1}{2} + \frac{2}{\pi}\varepsilon$. On remarque que les suites $(\lambda(2i))_i$ et $(\lambda(2i+1))_i$ sont des sous suites de la suite formée par l'inverse du produit de Wallis. Elles sont adjacentes et convergent vers $\frac{2}{\pi} \approx 0.637$.

4.5 Remarques sur l'estimation du biais d'une approximation

Cette section a pour thème l'estimation du biais des approximations linéaires. On commencera, sous-section 4.5.1, par présenter la technique classique utilisée depuis la présentation par Matsui de la cryptanalyse linéaire pour calculer ces biais. Comme on l'a évoqué en tête de ce chapitre, on sait estimer la probabilité

$$\Pr_{\boldsymbol{M},\boldsymbol{K}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \kappa, \boldsymbol{K}\rangle = \langle \gamma, E_{\boldsymbol{K}}(\boldsymbol{M})\rangle\right]$$

mais lors d'une attaque, la clef est fixée. Cette problématique se retrouve aussi en cryptanalyse différentielle mais le cas de la cryptanalyse linéaire est, on va le voir, bien plus problématique. Certaines attaques essayent de tirer profit du fait que pour certaines clefs, le biais de l'approximation est bien plus élevé comme on le verra dans les sous-sections 4.5.2 et 4.5.3.

4.5.1 Chaînage des approximations

Nous avons présenté les attaques en supposant que nous connaissions le biais des approximations utilisées. On va maintenant s'intéresser à l'estimation de ce biais. L'outil principal à ce propos est le fameux « Piling-up lemma » que l'on trouve dans [Mat94b].

Lemme 4.22. *Piling-up lemma.*
Soient r variables binaires X_i indépendantes. On note $\varepsilon \stackrel{\text{def}}{=} \Pr\left[X_1 \oplus \cdots \oplus X_r = 0\right] - \frac{1}{2}$ et $\varepsilon_i \stackrel{\text{def}}{=} \Pr\left[X_i = 0\right] - \frac{1}{2}$. Alors,

$$2\varepsilon - 1 = \prod_{i=1}^{r}(2\varepsilon_i - 1).$$

Si l'on suppose que les tours du chiffrement sont indépendants, alors on peut appliquer ce lemme aux approximations linéaires. Le principe exposé ci-après est appelé *chaînage* des approximations linéaires. Soit r le nombre de tours auquel on s'intéresse et soient r approximations linéaires (une par tour du chiffrement) $(\pi_j, \kappa_j, \gamma_j, \varepsilon_j)_{1\leq j\leq r}$ avec, pour $j > 1$, $\pi_j = \gamma_{j-1}$. Alors, si on définit $X_i \stackrel{\text{def}}{=} \langle \pi_j, \boldsymbol{M}\rangle \oplus \langle \kappa_j, \boldsymbol{K}\rangle \oplus \langle \gamma_j, E_{\boldsymbol{K}}(\boldsymbol{M})\rangle$, on obtient

$$\Pr\left[\langle \pi_1, \boldsymbol{M}\rangle \oplus \langle \bigoplus_{j=1}^{r} \kappa_j, \boldsymbol{K}\rangle \oplus \langle \gamma_n, E_{\boldsymbol{K}}(\boldsymbol{M})\rangle = 0\right] = \frac{1}{2} + 2^{r-1}\prod_{j=1}^{r}\varepsilon_j. \tag{4.9}$$

On a ainsi calculé le biais d'une approximation linéaire sur n tours.

Ce procédé nous permet donc de trouver des triplets (π, κ, γ) tels que

$$\Pr_{\boldsymbol{M},\boldsymbol{K}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \kappa, \boldsymbol{K}\rangle = \langle \gamma, E_{\boldsymbol{K}}(\boldsymbol{M})\rangle\right] = \frac{1}{2} + \varepsilon.$$

Cependant, comme on l'a déjà mentionné, lors d'une attaque, ce qui nous intéresse est la probabilité pour une clef $\boldsymbol{k}$ fixée. Il est d'usage de faire l'hypothèse suivante.

Hypothèse 4.23. *Hypothèse d'équivalence à clef fixée (linéaire).*
Soit une approximation linéaire (π, κ, γ) d'un chiffrement E. Pour toute clef $\boldsymbol{k} \in \mathbb{F}_2^{n_k}$,

$$\Pr_{\boldsymbol{M},\boldsymbol{K}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \kappa, \boldsymbol{K}\rangle = \langle \gamma, E_{\boldsymbol{K}}(\boldsymbol{M})\rangle\right] \approx \Pr_{\boldsymbol{M}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \kappa, \boldsymbol{k}\rangle = \langle \gamma, E_{\boldsymbol{k}}(\boldsymbol{M})\rangle\right].$$

Bien que pour le DES cette hypothèse soit vérifiée, ce n'est plus le cas des nouveaux systèmes conçus pour résister à la cryptanalyse linéaire et en particulier pas du tout le cas pour PRESENT.

Le problème de l'équivalence à clef fixée pour la cryptanalyse linéaire est assez complexe. Le problème fut soulevé en premier par Nyberg [Nyb95] qui introduit la notion de *hull effect*. Pour une paire de masques (π, γ), il est possible que l'on ait plusieurs approximations linéaires biaisées correspondant à des masques de clef différents. Prenons un exemple, soient deux approximations linéaires

$$\Pr_{\boldsymbol{M},\boldsymbol{K}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \gamma, E_{\boldsymbol{K}}(\boldsymbol{M})\rangle = \langle \kappa_1, \boldsymbol{K}\rangle\right] = \frac{1}{2} + \varepsilon,$$

et

$$\Pr_{\boldsymbol{M},\boldsymbol{K}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \gamma, E_{\boldsymbol{K}}(\boldsymbol{M})\rangle = \langle \kappa_2, \boldsymbol{K}\rangle\right] = \frac{1}{2} + \varepsilon.$$

Supposons maintenant que la clef utilisée $\boldsymbol{k}^*$ soit telle que

$$\langle \kappa_1, \boldsymbol{k}^*\rangle = 0 \quad \text{et} \quad \langle \kappa_2, \boldsymbol{k}^*\rangle = 1.$$

La question est alors, que vaut la probabilité

$$\Pr_{\boldsymbol{M}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \gamma, E_{\boldsymbol{k}^*}(\boldsymbol{M})\rangle = 0\right]?$$

Cette dernière est appelée probabilité fondamentale dans [Mur09a] car c'est elle que l'on estime empiriquement à l'aide des échantillons disponibles lors d'une attaque.

On voit bien que ce phénomène de *hull* induit de nombreux problèmes pour la cryptanalyse de type 1 (que déduire sur $\boldsymbol{k}^*$ du fait que le biais soit positif ?) ainsi que, dans une moindre mesure, pour la cryptanalyse de type 2. Le problème pour la cryptanalyse de type 2 est que l'on ne sait, a priori, pas quelle sera la valeur du biais pour la clef utilisée.

On peut trouver, dans les travaux de Nyberg [Nyb01], quelques réponses à ce sujet.

Théorème 4.24. *Théorème 4 dans [Nyb01]*

$$\begin{aligned} 2^{-n_k} \cdot & \sum_{\boldsymbol{k}\in\mathbb{F}_2^{n_k}} \left|2\Pr_{\boldsymbol{X}}\left[\langle \gamma, E_{\boldsymbol{k}}(\boldsymbol{X})\rangle = \langle \pi, \boldsymbol{X}\rangle\right] - \frac{1}{2}\right|^2 \\ = & \sum_{\kappa\in\mathbb{F}_2^{n_k}} \left|2\Pr_{\boldsymbol{X},\boldsymbol{K}}\left[\langle \gamma, E_{\boldsymbol{K}}(\boldsymbol{X})\rangle = \langle \pi, \boldsymbol{X}\rangle \oplus \langle \kappa, \boldsymbol{K}\rangle\right] - \frac{1}{2}\right|^2. \end{aligned}$$

Ce théorème nous permet d'estimer la moyenne du carré du biais à clef fixée. Le théorème suivant permet d'estimer le biais d'une approximation pour une clef fixée.

Théorème 4.25. *On s'intéresse à r tours d'un chiffrement E ayant pour fonction de tour $F_{\boldsymbol{k}}$ où $\boldsymbol{k}$ est une clef fixée. Si le « Piling-up lemma » est valide, alors on peut calculer le biais à clef fixée grâce à la formule suivante :*

$$\left|2\Pr_{\boldsymbol{X}}\left[\langle \pi, \boldsymbol{X}\rangle = \langle \gamma, E_{\boldsymbol{k}}(\boldsymbol{X})\rangle\right] - \frac{1}{2}\right| = \sum_{\boldsymbol{m}} \prod_{i=1}^{r} \left|2\Pr_{\boldsymbol{X}}\left[\langle \boldsymbol{m}_i, \boldsymbol{X}\rangle = \langle \boldsymbol{m}_{i+1}, F_{\boldsymbol{k}}(\boldsymbol{X})\rangle\right] - \frac{1}{2}\right|, \tag{4.10}$$

où $\boldsymbol{m}$ est un vecteur de masques $(m_1, \ldots, m_{r+1})$ tel que $m_1 = \pi$ et $m_{r+1} = \gamma$.

Ce théorème est prouvé par récurrence dans le cas des chiffrements de Feistel dans [Nyb01]. La preuve repose sur l'utilisation de l'expression des corrélations à l'aide de la transformée de Walsh.

4.5.2 Brève présentation et discussion de [Ohk09]

Chacun des vecteurs de masques du théorème 4.25 est appelé chemin (*trail* en anglais). Si on note $\boldsymbol{k}_1, \dots \boldsymbol{k}_r$ les clefs de tours et que le chiffrement à la propriété dite « key alternating » i.e. $F_{\boldsymbol{k}}(\boldsymbol{x}) = F_{\boldsymbol{0}}(\boldsymbol{x} \oplus \boldsymbol{k})$, alors on peut écrire

$$\Pr_{\boldsymbol{M}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \gamma, E_{\boldsymbol{k}^*}(\boldsymbol{M})\rangle = 0\right] \approx \frac{1}{2} + \sum_{\boldsymbol{m}} (-1)^{\bigoplus_{i=1}^{r} \langle m_i, \boldsymbol{k}_i\rangle} \varepsilon_{\boldsymbol{m}},$$

où $\varepsilon_{\boldsymbol{m}}$ est le biais du chemin $\boldsymbol{m}$:

$$\varepsilon_{\boldsymbol{m}} = \prod_{i=1}^{r} \left(\Pr_{\boldsymbol{X}}\left[\langle \boldsymbol{m}_i, \boldsymbol{X}\rangle = \langle \boldsymbol{m}_{i+1}, F_{\boldsymbol{k}}(\boldsymbol{X})\rangle\right] - \frac{1}{2} \right).$$

Ohkuma utilise ce résultat pour effectuer une cryptanalyse linéaire de PRESENT. Il regarde pour cela tous les masques correspondant à des approximations linéaires n'ayant qu'un bit d'actif à chaque tour [9] qui ont, selon les caractéristiques de PRESENT, toutes le même biais ε. La probabilité fondamentale dépend donc du nombre de masques pour lesquels le produit scalaire avec la clef donne 0. Il note n_+ ce nombre et n_- le nombre de masques pour lesquels on a 0. Le biais de la probabilité fondamentale est alors $(n_+ - n_-)\,\varepsilon$. En supposant que les approximations sont indépendantes, il approxime la distribution des $n_\pm$ par une loi normale et en utilisant les grandes déviations de celle-ci, il conclut que pour une proportion 0.32 des clefs, le biais réel est supérieur à $2^{-39.3}$. Le biais attendu si on ne prend pas en compte ce *hull effect* est $2^{-43.0}$

On a alors l'impression que ce phénomène est en fait un avantage pour l'attaquant. Cependant, deux problèmes font que cette attaque est sûrement bien loin de fonctionner. Le premier vient à l'esprit assez rapidement quand on connaît un peu PRESENT. Pour deux masques (π, γ) de poids 1 bien choisis [10] il existe certes un grand nombre d'approximations de biais maximum obtenues en chaînant des approximations de poids 1 mais il existe un nombre encore plus grand d'approximations ayant un biais à peine plus faible qui sont composées d'approximations de poids 1 sauf pour deux tours. La même chose est encore plus vraie si on diminue encore un peu le biais pour s'autoriser 3 tours avec des approximations avec plus d'un bit actif, etc... Il s'avère que si l'on ne prend pas en compte ces approximations les résultats sont totalement erronés. J'ai d'ailleurs effectué un test sur 4 tours de SMALLPRESENT-[32] pour lequel on peut énumérer toutes les approximations correspondant à deux masques (π, γ). Il y a un nombre impair d'approximations ayant le biais maximum $\varepsilon_{\max}$ et si l'on applique strictement la méthode d'Okhuma, on s'attend à ce que pour toute clef, le biais $|\varepsilon|$ soit supérieur à $\varepsilon_{\max}$. Cependant, du fait de ces approximations aux biais plus faible, on a, pour certaines clefs, des biais bien en dessous de $\varepsilon_{\max}$. En imaginant maintenant la croissance exponentielle du nombre de ces approximations sur un grand nombre de tours, on comprend vite que la chose est en fait bien plus compliquée que ce qui a été présenté.

Le second problème dont nous allons parler est traité dans [DWS10]. On suppose que l'on a un chiffrement pour lequel la première remarque ne s'applique pas. Pour un couple (π, γ) fixé, on a donc un grand nombre d'approximations avec le biais ε et seulement celles-ci. Étant donné que le nombre d'approximations linéaires biaisées correspondant à un couple (π, γ)

9. Cela signifie que les approximations sur 1 tour qui ont été chainées ont des masques de poids 1.
10. Ceux-ci ne doivent pas correspondre aux 4 premières boîtes-S ni aux bits de poids faible en entrée/sortie d'une de ces boîtes.

croît avec le nombre de tours, il arrive un moment où les masques de clefs correspondant ne sont plus linéairement indépendants. Cela implique, comme il est prouvé dans [DWS10], que la proportion de bonnes clefs est en fait bien plus faible que prévu.

4.5.3 Brève présentation et discussion de [NSZW09]

Une partie de [NSZW09] est consacrée à la cryptanalyse linéaire et au hull effect. Dans celle-ci, il est proposé de prendre en compte ce phénomène dans l'analyse d'une cryptanalyse linéaire. Pour cela, le théorème 4.24 est utilisé. On note

$$\eta^2 \stackrel{\text{def}}{=} \sum_{\boldsymbol{m}} \varepsilon_{\boldsymbol{m}}^2.$$

Cette valeur η^2 est la moyenne des carrés des biais à clef fixée et les auteurs en déduisent que la complexité en données d'une cryptanalyse linéaire utilisant l'approximation formée par les masques (m_1, m_{r+1}) est alors de l'ordre de η^{-2}.

Dans [Mur09a], Murphy rappelle que regarder η^{-2} revient à ignorer le fait que l'inégalité de Jensen n'est pas une égalité. En effet, la complexité en données moyenne est $\mathbb{E}\left(\frac{1}{\varepsilon_{\boldsymbol{k}}^2}\right)$ alors qu'en utilisant η, on calcule $\frac{1}{\mathbb{E}\left(\varepsilon_{\boldsymbol{k}}^2\right)}$. Cependant, pour certaines clefs, le biais peut être nul et la complexité donc infinie.

La moralité est que dans le cas du hull effect, il semble plus naturel d'utiliser l'approche d'Ohkuma qui consiste à fixer une borne sur la complexité en données puis à quantifier le nombre de clefs pour lesquelles on a bien une complexité inférieure à cette borne.

4.6 Bilan

Dans cette section on va résumer les complexités des différentes attaques et comparer leurs utilisations. Le tableau 4.3 contient les complexités des différentes attaques présentées. À noter que les complexités en temps données pour les attaques multidimensionnelles prennent en compte l'utilisation de la technique de la convolution présentée dans [HN10] et adaptée à la cryptanalyse de type 3 dans [Her10] (l'exception de la cryptanalyse de type 2).

Attaque	N	N approché	C. en temps	Source(s)
Type 1	$\frac{\Phi^{-1}(P_S)}{4\varepsilon^2}$	$\mathcal{O}\left(\frac{1}{4\varepsilon^2}\right)$	$\mathcal{O}(N)$	[Mat94b]
Type 2 et 3	$\frac{(\Phi^{-1}(P_S)-\Phi^{-1}(1-2^{-a-1}))^2}{4\varepsilon^2}$	$\mathcal{O}\left(\frac{a\,\ln(2)}{2\varepsilon^2}\right)$	$\mathcal{O}(N+3n_O\,2^{n_O})$	[CSQ07, Sel08]
MK1	$\frac{\Phi^{-1}(1-2^{-a-1})^2}{2\sum_{j=1}^{n}\varepsilon_j^2}$	$\mathcal{O}\left(\frac{a\,\ln(2)}{\sum_{j=1}^{n}\varepsilon_j^2}\right)$	$\mathcal{O}(n(N+2^{n_I}))$	[BDCQ04]
MK2	$\frac{\Phi^{-1}(1-2^{-a-1})^2}{2\sum_{j=1}^{n}\varepsilon_j^2}$	$\mathcal{O}\left(\frac{a\,\ln(2)}{\sum_{j=1}^{n}\varepsilon_j^2}\right)$	$\mathcal{O}(N+2\,n\,n_O\,2^{n_O})$	[BDCQ04, CSQ07]
Multid. 1 χ^2	$\frac{\lambda(d,P_S)+2^{\frac{d-1}{2}}\sqrt{\lambda(d,P_S)}}{\sum_{j=1}^{2^d-1}\varepsilon_j^2}$	$\mathcal{O}\left(\frac{2^{d/2}}{\sum_{j=1}^{2^d-1}\varepsilon_j^2}\right)$	$\mathcal{O}\left(d\,2^d\right)$	[HCN09c, HN10]
Multid. 1 `LLR`	$\frac{(\Phi^{-1}(P_S)-\Phi^{-1}(1-2^{-a-1}))^2}{\sum_{j=1}^{2^d-1}\varepsilon_j^2}$	$\mathcal{O}\left(\frac{2\,a\,\ln(2)}{\sum_{j=1}^{2^d-1}\varepsilon_j^2}\right)$	$\mathcal{O}\left(d\,2^d\right)$	[HCN09c, HN10]
Multid. 2 χ^2	$\frac{2^{d/2}\Phi^{-1}(1-2^{-a})+2\Phi^{-2}(2P_s-1)}{2\sum_{j=1}^{2^d-1}\varepsilon_j^2}$	$\mathcal{O}\left(\frac{2^{d/2}\,a^{1/2}}{2\sum_{j=1}^{2^d-1}\varepsilon_j^2}\right)$	$\mathcal{O}\left(2^{d+n_O}\right)$	[HCN09a]
Multid. 3 `LLR`	$\frac{\left(\Phi^{-1}(\sqrt[2^d-1]{1-2^{-a}})+\Phi^{-1}(P_{12})\right)^2}{\sum_{j=1}^{2^d-1}\varepsilon_j^2}$	$\mathcal{O}\left(\frac{a+d}{2\sum_{j=1}^{2^d-1}\varepsilon_j^2}\right)$	$\mathcal{O}\left(n_k\,2^{d+n_O}\right)$	[HCN09a, Her10]

TABLE 4.3 – Complexités en donnée des différentes cryptanalyses linéaires.

Avec $\lambda(d,P_S)\stackrel{\text{def}}{=}d-\ln(\sqrt{2\pi}\ln(P_S^{-1}))$, n_O le nombre de bits de clef externe et n_I le nombre de bits de clefs interne. La probabilité P_{12} est celle que la bonne clef externe $\boldsymbol{k}_O$ soit retenue sachant qu'on lui a appairé la bonne clef interne $\boldsymbol{k}_I$ et donc $P_{12}<P_S$.

4.7 Recherche d'approximations linéaires d'un chiffrement

Voyons maintenant comment l'on peut automatiser la recherche de bonnes approximations linéaires. Nous présentons ici deux approches très différentes qui ont pour point commun de retourner les approximations ayant un biais supérieur à une borne donnée. La première est un raffinement de l'algorithme présenté par Matsui dans [Mat95] que l'on peut retrouver dans [BDCQ04]. Il est basé sur un parcours d'arbre en profondeur avec une évaluation à chaque nœud de l'intérêt d'explorer une sous-branche ou non. La seconde est basée sur un algorithme de décodage des codes de Reed-Muller de grande longueur [FLT09a, FLT09b]. Elle a deux inconvénients majeurs, le premier est le fait que l'on doive fixer le masque de sortie pour lancer l'algorithme et le second est son temps d'exécution. Cependant, cette méthode a un point fort : l'on n'est pas obligé de décortiquer l'algorithme de chiffrement et juste le considérer comme une boîte noire. Point fort qui semble sans intérêt étant donné le principe de Kerckhoffs mais qui prend tout son sens dans le contexte des attaques dites « par canaux cachés » [RT09].

4.7.1 Algorithme par séparation et évaluation

Pour aider à la simplicité de l'explication, on se place dans le cadre de la recherche d'approximations linéaires d'un chiffrement itératif de type substitution-permutation [11]. On note $\varepsilon_{\min}$ la borne (en valeur absolue) sur le biais des approximations que l'on souhaite trouver, R le nombre de tours sur lequel on recherche des approximations, $S(\cdot)$ la fonction de substitution et $P(\cdot)$ la permutation.

La première étape est de calculer les biais de toutes les approximations linéaires de la fonction de substitution S afin de gagner en efficacité. On note $\varepsilon(\pi, \gamma)$ la fonction qui renvoie le biais de l'approximation linéaire de S ayant π comme masque d'entrée et γ comme masque de sortie. Pour simplifier, on va commencer par supposer que l'on fixe le masque d'entrée π des approximations recherchées. L'idée est de faire parcourir à l'algorithme l'arbre défini comme suit.

- Chaque nœud/feuille contient un masque.
- La racine contient π.
- Les fils d'un nœud contenant un masque μ correspondent aux masques μ' tels que $\varepsilon(\mu, \mu') \neq 0$. L'arête reliant le père au fils est alors pondérée par la valeur de $\varepsilon(\mu, \mu')$.
- La profondeur de l'arbre vaut R.

À une feuille contenant un masque γ correspond donc une approximation (π, γ) ayant pour biais 2^{R-1} fois le produit des poids des arêtes sur le chemin liant la feuille à la racine (selon le « Piling-up lemma »). À noter que l'on peut aussi calculer la valeur du masque de clef en prenant en compte les différences intermédiaires rencontrées.

Cet arbre étant bien trop grand, on ne peut le parcourir en entier. C'est pourquoi on demande une borne $\varepsilon_{\min}$ sur la valeur absolue du biais. Ceci va nous permettre d'élaguer cet arbre ou plutôt d'éviter de parcourir des branches dénuées d'intérêt. Une première chose (simple) à faire pour éviter de parcourir des branches inutiles est de ne pas descendre dans un sous-arbre quand le poids du chemin du nœud courant à la racine est déjà inférieur à $\varepsilon_{\min}$. Cependant, l'on peut faire mieux. On construit de manière itérative un tableau contenant les biais des meilleures approximations pour tous les nombres de tours inférieurs à R. Pour 1

11. Cependant, cet algorithme peut être adapté à tout type de chiffrement itératif ainsi qu'à la cryptanalyse différentielle.

tour, les meilleures approximations correspondent aux meilleures approximations d'une boîte-S. Supposons donc que l'on a ce tableau à notre disposition, on note $\varepsilon_{\max}^{(r)}$ le biais maximal pouvant être obtenu sur r tours. On peut alors décider de stopper la descente à un nœud de profondeur r si le biais du chemin courant multiplié par $2 \cdot \varepsilon_{\max}^{(R-r)}$ est inférieur à $\varepsilon_{\min}$. En effet, cela signifie que même si l'on arrivait à chaîner la meilleure approximation sur les $R-r$ tours restants, le biais final serait inférieur à la borne et donc cela implique que toutes les feuilles découlant de ce nœud ne nous intéressent pas.

La deuxième chose est que le nombre d'approximations d'un tour est bien trop élevé pour que l'on puisse calculer et stocker les biais de toutes ces approximations. On découpe donc chaque tour en plusieurs morceaux (dans les cas qui nous intéressent, on découpe selon les boîtes-S). Cela revient à remplacer chaque nœud par un arbre ayant pour profondeur le nombre de boîtes-S. La profondeur de l'arbre est donc multipliée par le nombre de boîtes-S composant la fonction de substitution.

L'algorithme 7 résume tout ce qui a été dit précédemment. Les notations utilisées sont les suivantes :

- le nombre de boîtes-S est noté n_s ;
- pour un masque μ, la notation $\mu_{|s}$ représente les bits de μ qui correspondent à la boîte-S numéro s ;
- la notation $\varepsilon_s(a, b)$ correspond au biais de l'approximation linéaire de la boîte-S numéro s ayant pour masques a et b (respectivement masque d'entrée et de sortie).
- P est la permutation linéaire du chiffrement.

Le lancement de l'algorithme pour trouver les approximations sur R tours ayant un biais minimum de $\varepsilon_{\min}$ et un masque d'entrée π s'effectue par l'appel `Recherche(1,0,1,`R`,`$\varepsilon_{\min}$`,`π`,0)`.

Algorithme 7 : Recherche d'approximations linéaires (séparation-évaluation).

```
Recherche(r, s, ε, R, ε_min, π, γ)
si s = n_s + 1 alors
    r ⟵ r + 1;
    si r ≤ R alors
        s ⟵ 0;
        π ⟵ P(γ);
        γ ⟵ 0;
    sinon
        Afficher (π, γ, ε);
    fin
fin
si 2 ε ε_max^(R−r) ≥ ε_min alors
    si π_|s = 0 alors
        Recherche(r, s + 1, ε, R, ε_min, π, γ);
    sinon
        γ_|s ⟵ i;
        pour chaque i, ε_s(π_|s, i) ≠ 0 faire
            Recherche(r, s + 1, 2 ε ε_s(π_|s, i), R, ε_min, π, γ);
        fin
    fin
fin
```

Bien entendu, il est possible de ne pas fixer le masque d'entrée. Il faut, pour cela, différencier les appels de la fonction avec $r = 0$ des autres. On remplace alors la seconde partie de la fonction par une boucle regardant tous les couples (i, j) tels que $\varepsilon_s(i, j)$ est non nul. De plus, l'on peut aussi obtenir le masque de clef des approximations trouvées. Il suffit pour cela d'ajouter un argument à la fonction `Recherche`. Cet argument contiendra, en fait, la liste des bits de sous-clef étant intervenus dans l'approximation. Il sera ensuite possible, pour les chiffrements ayant un algorithme de cadencement de clef linéaire, de traduire cette information en un masque de clef κ.

On peut imaginer plusieurs petites améliorations techniques de cet algorithme qui relèvent de l'implémentation comme le fait d'incrémenter s tant que $\pi_{|s}$ est nul sans faire d'appel afin de limiter la profondeur de récursion. On peut aussi, dans le même but, regarder les boîtes-S deux par deux (ou plus selon le chiffrement et les ressources). On peut aussi imaginer d'autres critères d'arrêt comme par exemple donner une borne supérieure sur le biais que l'on peut obtenir sur r tours qui dépende du nombre de boîtes-S actives au tour actuel et bien d'autres choses encore.

4.7.2 Décodage du code de Reed-Muller d'ordre 1

Nous présentons ici l'algorithme basé sur le décodage des codes de *Reed-Muller* de grande longueur. La première étape est donc de présenter cette famille de codes d'un intérêt particulier en cryptographie vu qu'ils sont liés aux fonctions booléennes.

Définition 4.26. (Code de Reed-Muller). Le *code de Reed-Muller* d'ordre r à v variables est le sous-espace vectoriel de $\mathbb{F}_2^{2^v}$ constitué des tables de vérité des fonctions booléennes à v variables de degré au plus r.

Le code de Reed-Muller d'ordre r à v variables est de longueur 2^v et a pour dimension la somme $\sum_{i=0}^{r} \binom{v}{i}$.

Décodage des codes de Reed-Muller par transformée de Walsh. On va voir ici comment on peut, de façon efficace, effectuer un décodage complet d'un code de Reed-Muller d'ordre 1 pour un nombre de variables raisonnable. Soit f une fonction booléenne à v variables de degré 1. Alors il existe $\boldsymbol{\zeta} \in \mathbb{F}_2^v$ et $b \in \mathbb{F}_2$ tels que l'on puisse écrire $f(x) = \langle \boldsymbol{\zeta}, x \rangle + b$. Supposons donc que l'on transmette un mot de code correspondant à une fonction booléenne f de degré 1. Soit f_e la fonction booléenne correspondant au vecteur reçu, on va s'intéresser à la fonction g définie par $g(\boldsymbol{x}) \stackrel{\text{def}}{=} (-1)^{f_e(\boldsymbol{x})}$. Alors, la transformée de Walsh de f_e est

$$\hat{g}(\boldsymbol{u}) = \sum_{\boldsymbol{x} \in \mathbb{F}_2^v} g(\boldsymbol{x}) \, (-1)^{\langle \boldsymbol{u}, X \rangle} = \sum_{\boldsymbol{x} \in \mathbb{F}_2^v} (-1)^{f_e(\boldsymbol{x}) + \langle \boldsymbol{u}, X \rangle}.$$

On obtient donc les corrélations entre f_e et toutes les fonctions booléennes de degré 1. Le mot le plus vraisemblablement envoyé correspond donc à la fonction booléenne affine $f(x) = \langle \boldsymbol{\zeta}, x \rangle + b$ avec

$$\boldsymbol{\zeta} = \underset{\boldsymbol{u} \in \mathbb{F}_2^v}{\operatorname{argmax}} \, |\hat{g}(\boldsymbol{u})| \quad \text{et} \quad b = \begin{cases} 0 \text{ si } \hat{g}(\boldsymbol{u}) > 0, \\ 1 \text{ sinon.} \end{cases}$$

Le calcul de la transformée de Walsh d'une fonction s'effectue efficacement sur un ordinateur. En effet, il suffit de construire la table de vérité de la fonction considérée puis en utilisant un algorithme du type *diviser pour mieux régner*, on effectue la transformation en place et on

récupère la table de vérité de la transformée en $\mathcal{O}(v\,2^v)$ opérations. En triant ce tableau, on obtient donc une liste ordonnée des mots les plus vraisemblables et le décodage en liste est alors trivial. La restriction vient de la taille de la table de vérité de la fonction regardée (2^v) et donc pour un v trop grand on sera limité par la mémoire.

Codes de Reed-Muller d'ordre 1 et approximations linéaires. Un code de Reed-Muller d'ordre 1 est composé de toutes les fonctions affines à un nombre de variables donné. La table de vérité d'une fonction booléenne f donnée de v variables peut être considérée comme un mot bruité du code de Reed-Muller d'ordre 1 à v variables. Effectuer un décodage par distance minimale revient alors à trouver la fonction $\tilde{f}$ linéaire à v variables coïncidant avec f en un nombre maximum de points. Cette fonction $\tilde{f}$ est donc la meilleure approximation linéaire de f. Le biais de l'approximation linéaire est alors

$$\varepsilon = \frac{\#\left\{x \in \mathbb{F}_2^v,\, f(x) = \tilde{f}(x)\right\} - 2^{v-1}}{2^v}.$$

On a donc envie d'utiliser l'algorithme susmentionné afin d'obtenir rapidement des approximations linéaires d'un chiffrement. On a cependant deux problèmes :

– un chiffrement par blocs n'est pas une fonction booléenne ;
– la longueur du code serait 2^{n_k+s}.

Pour ce qui est du premier point, il suffit pour cela de considérer la fonction $f(\boldsymbol{x}, \boldsymbol{k}) = \langle E_{\boldsymbol{k}}(\boldsymbol{x}), \gamma\rangle$. Il faudra donc, pour utiliser cette technique, fixer le masque de sortie γ. C'est donc un premier désavantage de la méthode. Pour le second problème, il est clair que pour un chiffrement raisonnable, on a $n_k + s \geq 144$. Il est donc totalement impossible ne serait-ce que de stocker une infime partie du vecteur à décoder. Il est cependant possible de décoder, de façon probabiliste, sur un code de cette longueur.

Décodage probabiliste du code de Reed-Muller d'ordre 1 en grande longueur Nous présentons ici les travaux que l'on peut retrouver dans [Tav04, FLT09a, FLT09b] sur le décodage des codes de Reed-Muller de grande longueur et de l'application de cet algorithme à la recherche d'approximations linéaires d'un chiffrement par blocs. Cet algorithme se base sur le fait que si l'on a une relation

$$\Pr\left[\langle \alpha, \boldsymbol{A}\rangle \oplus \langle \beta, \boldsymbol{B}\rangle = 0\right] = \frac{1}{2} + \varepsilon,$$

alors, pour une valeur $\boldsymbol{b}$ fixée de $\boldsymbol{B}$ et si l'ensemble des valeurs pouvant être prises par $\boldsymbol{A}$ est suffisamment grand,

$$\Pr\left[\langle \alpha, \boldsymbol{A}\rangle \oplus \langle \beta, \boldsymbol{b}\rangle = 0\right] \approx \frac{1}{2} + \varepsilon.$$

On divise alors le problème en deux étapes :

– un décodage complet d'un sous-code du code Reed-Muller d'ordre 1 à v variables ;
– puis la reconstruction des mots de codes entiers à partir des résultats de ce décodage.

Un mot de code étant bien trop grand à stocker, on identifiera un mot de code de $\mathbb{F}_2^v$ au vecteur (a_0, a_{v-1}, a_v) de $\mathbb{F}_2^{v+1}$ contenant les coefficients de sa forme algébrique normale.

La première étape consiste alors à effectuer un décodage complet sur un sous-code en ne considérant qu'un nombre réduit v' de variables et en utilisant la transformée de Walsh. Si l'on cherche à trouver toutes les approximations linéaires ayant un biais supérieur à $\varepsilon_{\min}$, il faudra

alors que la transformée de Walsh sur v' variables puisse détecter ce biais i.e. $2^{v'} \approx \varepsilon_{\min}^{-2}$. On reconstruit ensuite, variable par variable, le mot de code à l'aide d'un processus de filtrage. Pour cela, on a à notre disposition une liste d'approximations tronquées obtenues après la transformée de Walsh. Chaque étape de reconstruction va chercher à déterminer la valeur d'un des coefficients de l'approximation. La liste $\mathcal{L}_{i+1}$ à l'itération $i+1$ sera obtenue en ajoutant un coefficient à chacune des approximations de $\mathcal{L}_i$. Ce coefficient pouvant prendre la valeur 0 ou 1, on aura donc une liste de taille double car l'on aura deux possibilités pour étendre une approximation. On estime ensuite le biais de chacune des estimations ainsi formées et on garde uniquement celles pour lesquelles on a détecté un biais significatif. Cela implique donc une complexité de $O(\varepsilon_{\min}^{-2})$ si l'on suppose que la taille de la liste est $O(1)$ [12]. Ce dernier point peut être vu comme un second inconvénient de l'algorithme car trouver des approximations a un coût aussi élevé que de les utiliser pour une vraie attaque. Cependant, nous avons vu dans la section 4.5 que l'estimation théorique du biais d'une approximation est loin d'être évidente et le fait de calculer le biais empiriquement peut ne pas être une si mauvaise idée au final.

Pour plus de détails il est, bien entendu, conseillé de se reporter aux ouvrages cités [Tav04, FLT09a]. L'algorithme 8 récapitule l'ensemble du processus. On note $\hat{\boldsymbol{F}}_{f(\cdot,\boldsymbol{s})}$ la transformée de Walsh de la fonction f restreinte à ses premières variables, $\boldsymbol{s}$ étant la valeur fixe prise par les autres. La notation $\mathcal{L} + e_i$ signifie l'ensemble des approximations de $\mathcal{L}$ auxquelles on a donné la valeur 1 pour le i-ème coefficient.

Avantages et utilisation de cette technique. On a donc vu deux gros inconvénients de cette technique :

- il faut fixer le masque de sortie ;
- l'estimation du biais étant empirique, la complexité de l'algorithme est en $O(\varepsilon_{\min}^{-2})$.

Le fait de fixer un masque est plus gênant vu que cela signifie qu'il faut deviner les bons masques de sortie étant donnée la taille des chiffrements par blocs utilisés.

Malgré cela, il s'avère que cet algorithme peut être d'une grande utilité car il existe des circonstances dans lesquelles les biais considérés sont suffisamment élevés pour que le coût en $O(\varepsilon_{\min}^{-2})$ soit raisonnable et que le nombre de masque de sortie soit faible. L'exemple le plus criant aujourd'hui est celui des attaques par canaux auxiliaires. Ce type de cryptanalyse recueille de l'information en utilisant des fuites dues à l'implémentation physique de l'algorithme. Selon le modèle de fuite utilisé, cette information prend la forme d'un poids de Hamming d'un état, une distance de Hamming entre deux états, ... La fonction qui, à un couple message/clef, associe cette information est appelée *fonction de fuite*. Une attaque présentée dans [RT09, Roc10] utilise des approximations linéaires d'une fonction de fuite. Le problème est qu'une telle fonction n'est pas connue de l'attaquant car elle dépend de bien trop de paramètres. En effet, elle ne sera pas la même selon la puce, la température, l'état du matériel, ... Cependant, cette fonction a à peu près toujours le même comportement. Si on a accès à une implémentation d'un chiffrement, on peut alors évaluer cette fonction autant de fois que nécessaires et donc appliquer l'algorithme 8 afin d'obtenir des approximations linéaires.

On voit bien ici que l'on profite au maximum du fait que cet algorithme se sert de la fonction comme d'une boîte noire. De plus, la fonction de fuite étant souvent un poids ou une distance de Hamming, le nombre de masques en sortie est relativement faible.

12. Ceci est le cas pour des paramètres bien choisis.

Algorithme 8 : Recherche d'approximations linéaire (Reed-Muller).

Entrées : La fonction booléenne f dont on veut une approximation.

```
// Décodage complet
```

$\mathcal{L} \longleftarrow \emptyset$;
$\boldsymbol{h} \longleftarrow (0, \ldots, 0)$;
pour *un paramètre fixé d'itérations* **faire**
  Choisir $\boldsymbol{s} \in \mathbb{F}_2^{v-v'}$ aléatoirement;
  Calculer la transformée de Walsh $\hat{\boldsymbol{F}}_{f(\cdot,\boldsymbol{s})}$;
  $\boldsymbol{h} \longleftarrow (h_0 + |\hat{F}(0)|, \ldots, h_{2^{v'}-1} + |\hat{F}(2^{v'}-1)|)$;
fin
pour $0 \leq i \leq 2^{v'} - 1$ **faire**
  si h_i *est supérieur à un seuil fixé* **alors**
    $\mathcal{L} \longleftarrow \mathcal{L} \cup i$;
  fin
fin

```
// Processus de filtrage
```

pour i *allant de* $v' + 1$ *à* v **faire**
  $\mathcal{L} \longleftarrow \mathcal{L} \cup (\mathcal{L} + e_i)$;
  pour chaque *approximation* $\boldsymbol{a}$ *dans* $\mathcal{L}$ **faire**
    pour *un paramètre fixé d'itérations* **faire**
      Choisir $\boldsymbol{s} \in \mathbb{F}_2^{v-i}$ aléatoirement;
      Estimer le biais de $\boldsymbol{a}$ en prenant un nombre aléatoire suffisant de valeurs pour les i premières variables ;
      si *le biais n'est pas significatif* **alors**
        $\mathcal{L} \longleftarrow \mathcal{L} \backslash \{\boldsymbol{a}\}$;
      fin
    fin
  fin
fin
retourner $\mathcal{L}$

Chapitre 5

Cryptanalyse différentielle

Sommaire

Ce chapitre présente la cryptanalyse différentielle standard ainsi que ses principales variantes. Le but est double, tout d'abord cela va permettre d'introduire la cryptanalyse différentielle et les problématiques liées à son analyse (section 5.1) que l'on étudiera plus en profondeur dans le chapitre 9. La seconde chose est d'avoir en tête différents types de cryptanalyses statistiques afin de pouvoir mieux appréhender le chapitre 6 où il sera question de l'étude générale de ces attaques. Ces variantes seront présentées à la section 5.2. Dans la section 5.3, on s'intéressera, enfin, au problème du calcul de la probabilité du phénomène observé dans les attaques différentielles.

Principe fondateur et notations. La première apparition de la cryptanalyse différentielle est due à Eli Biham et Adi Shamir [BS91a, BS91b]. L'attaque présentée est une cryptanalyse statistique qui utilise le phénomène suivant que l'on appelle différentielle.

Définition 5.1. Une *différentielle* d'un chiffrement E est un couple (δ_a, δ_b) de $\mathbb{F}_2^s \times \mathbb{F}_2^s$ tel que

$$\Pr_{\boldsymbol{M},\boldsymbol{K}}\left[E_{\boldsymbol{K}}(\boldsymbol{M}) \oplus E_{\boldsymbol{K}}(\boldsymbol{M} \oplus \delta_a) = \delta_b\right] = p^*.$$

On appelle respectivement δ_a et δ_b les *différences en entrée et sortie* et p^* la probabilité de la différentielle.

Comme on s'intéresse à des chiffrements itératifs, on utilisera plutôt la notation δ_0 et δ_r pour les différences en entrée et en sortie pour r tours d'un chiffrement. On note qu'ici, on a une attaque à clairs choisis vu qu'il faut prendre des clairs avec une certaine différence en entrée. Cependant, on peut aussi effectuer une cryptanalyse différentielle avec des clairs connus. On s'attachera alors (si possible) à utiliser une différentielle ayant la différence en entrée permettant de former le plus de paires possibles. Notons aussi qu'ici, et à la différence de la cryptanalyse linéaire, $2N$ couples clair/chiffré ne donnent lieu qu'à N échantillons utiles pour tester si $E_{\boldsymbol{K}}(\boldsymbol{M}) \oplus E_{\boldsymbol{K}}(\boldsymbol{M} \oplus \delta_a) = \delta_b$. C'est ce qui fait de cette attaque une cryptanalyse statistique d'ordre 2 [1].

5.1 Cryptanalyse différentielle

La cryptanalyse différentielle présentée par Biham et Shamir [BS91a, BS91b] est une attaque sur un réseau de Feistel : le DES. Elle est assez différente des attaques sur le dernier tour comme celle de [Wan08]. On présentera donc les deux types de cryptanalyse qui s'analysent de façon similaire.

5.1.1 Description de l'attaque sur un réseau de Feistel

On présente ici le principe de la cryptanalyse différentielle d'un réseau de Feistel. Supposons que nous ayons une différentielle sur r tours. On note δ_0^g et δ_0^d (resp. δ_r^g et δ_r^d) les deux parties des différentielles correspondant aux deux moitiés de l'état comme le montre la figure 5.1. Le concept est que la structure de Feistel du chiffrement nous permet de connaître

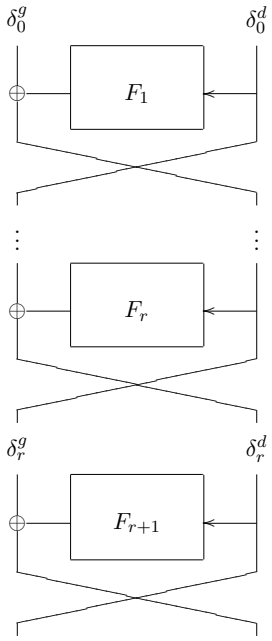

FIGURE 5.1 – Différentielle sur r tours d'un réseau de Feistel

la différence et la valeur en entrée de F_{r+1}. De plus, si on suppose que la différentielle sur

1. Cette notion d'ordre est définie, on le rappelle, dans la section 1.5.

r tours est vérifiée, alors on connaît aussi la différence en sortie de F_{r+1}. Pour une paire de clairs $\boldsymbol{m}$ et $\boldsymbol{m}'$ tels que $\boldsymbol{m} \oplus \boldsymbol{m}' = (\delta_0^g || \delta_0^d)$ et leurs chiffrés après $r+1$ tours $\boldsymbol{c}_1$ et $\boldsymbol{c}_2$, si la différence après r tours est bien δ_r, alors on a

1. la différence en entrée de F_{r+1} est $\boldsymbol{c}_1^g \oplus \boldsymbol{c}_2^g = \delta_r^d$;
2. la valeur en entrée de F_{r+1} pour le premier couple est $\boldsymbol{c}_1^g$;
3. la différence en sortie de F_{r+1} est $\boldsymbol{c}_1^d \oplus \boldsymbol{c}_2^d \oplus \delta_r^g$;

La connaissance des différences en entrée et sortie de F_{r+1}, restreint le nombre de possibilités pour les valeurs en entrée des boîtes-S. Le tableau 5.1 est la *table des différences* de la boîte-S de PRESENT [2]. Il contient l'ensemble des valeurs de $\#\{x \in \mathbb{F}_2^4, S[x] \oplus S[x \oplus a] = b\}$ où $S[x]$ est l'image de x par la boîte-S regardée. Bien évidemment les nombres indiqués sont pairs car

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	-	4	-	-	-	4	-	4	-	-	-	4	-	-
2	-	-	-	2	-	4	2	-	-	-	2	-	2	2	2	-
3	-	2	-	2	2	-	4	2	-	-	2	2	-	-	-	-
4	-	-	-	-	-	4	2	2	-	2	2	-	2	-	2	-
5	-	2	-	-	2	-	-	-	-	2	2	2	4	2	-	-
6	-	-	2	-	-	-	2	-	2	-	-	4	2	-	-	4
7	-	4	2	-	-	-	2	-	2	-	-	-	2	-	-	4
8	-	-	-	2	-	-	-	2	-	2	-	4	-	2	-	4
9	-	-	2	-	4	-	2	-	2	-	-	-	2	-	4	-
A	-	-	2	2	-	4	-	-	2	-	2	-	-	2	2	-
B	-	2	-	-	2	-	-	-	4	2	2	2	-	2	-	-
C	-	-	2	-	-	4	-	2	2	2	2	-	-	-	2	-
D	-	2	4	2	2	-	-	2	-	-	2	2	-	-	-	-
E	-	-	2	2	-	-	2	2	2	2	-	-	2	2	-	-
F	-	4	-	-	4	-	-	-	-	-	-	-	-	-	4	4

TABLE 5.1 – Table des différences de la boîte-S de PRESENT.

si la relation $S[x] \oplus S[x \oplus a] = b$ est vraie pour x elle l'est aussi pour $x \oplus a$. L'autre propriété de cette table est que la somme d'une ligne ou d'une colonne doit donner 16 qui est le nombre de valeurs en entrée ou en sortie de la boîte. Enfin, la case $(0, 0)$ vaut 16 pour des raisons évidentes.

Revenons-en à l'attaque. On voit avec le tableau 5.1 que pour une différence en entrée et une différence en sortie fixées, on n'a qu'un certain nombre de valeurs possibles en entrée de la boîte-S. Connaissant la valeur en entrée de F_{r+1} et vu que la clef de tour est ajoutée avant de passer dans les boîtes-S, alors cet ensemble d'entrées possibles d'une boîte-S induit un ensemble de clefs possibles. L'idée est donc de compter le nombre de fois qu'une clef fait partie de ces clefs possibles et, avec suffisamment d'échantillons, on doit voir la bonne clef sortir du lot. L'algorithme 9 récapitule l'attaque présentée dans ce paragraphe. On rappelle que la fonction de tour $F(\cdot)$ du DES est composée d'une expansion $E(\cdot)$, d'une addition avec la clef, une couche de boîtes-S $S(\cdot)$ et une permutation $P(\cdot)$.

Cette attaque peut, cependant, être améliorée. Supposons, par exemple, que la différence δ_r^d ait des 0 partout sauf sur certains bits correspondant à la première boîte-S de F_{r+1}. La différence en sortie des boîtes-S non actives est forcément 0 et donc on sait que certains bits de $(\boldsymbol{c} \oplus \boldsymbol{c}')^d \oplus \delta_r^g$ doivent être nuls. Cela permet donc de mettre de côté certaines des paires

2. On a choisi la boîte-S de PRESENT car elle est plus petite que celles du DES.

Algorithme 9 : Cryptanalyse différentielle d'un réseau de Feistel.

Entrée : N couples clair/chiffré;
Sortie : La valeur devinée de $\boldsymbol{k}^*_{r+1}$;
début
  pour *toutes les valeurs possibles* $\boldsymbol{k}$ *pour* $\boldsymbol{k}^*_{r+1}$ **faire**
    $t_{\boldsymbol{k}} \longleftarrow 0$;
  fin
  pour $1 \leq i \leq N$ *et* $\boldsymbol{k} \in \mathbb{F}_2^{n_k}$ **faire**
    si $P\left(S(E(\boldsymbol{c}_1^{i,g}) \oplus \boldsymbol{k}) \oplus S(E(\boldsymbol{c}_2^{i,g}) \oplus \boldsymbol{k})\right) = \delta_r^g \oplus c_1^{i,d} \oplus c_2^{i,d}$ **alors**
      $t_{\boldsymbol{k}} \longleftarrow t_{\boldsymbol{k}} + 1$;
    fin
  fin
fin
retourner $\boldsymbol{k}_{r+1} = \underset{\boldsymbol{k}}{\operatorname{argmax}}\, t_{\boldsymbol{k}}$;

qui ne suivent pas la différentielle. Ce procédé est appelé *crible* et son efficacité dépend de la différentielle utilisée ainsi que du chiffrement cible.

D'un point de vue de l'analyse probabiliste, on a deux probabilités en jeu. La probabilité qu'une paire incrémente le compteur de la bonne clef que l'on notera p_0 et la probabilité que le compteur d'un mauvais candidat soit incrémenté par une paire que l'on notera p. Pour calculer ces probabilités, on va distinguer deux types de paires parmi les échantillons passant le crible. Soit p^* la probabilité de la différentielle. Alors, avec probabilité p^*, une paire de messages avec différence δ_0 en entrée donnera des chiffrés avec différence δ_r en sortie. On appelle une telle paire une *bonne paire*. Les autres paires sont appelées *mauvaises paires*. Pour reprendre les notations de [BS91a, BS91b], on note α le nombre de clefs comptées par paire et β la proportion de paires ayant passé le crible.

Bonne paire : Pour une bonne paire, la bonne clef sera incrémentée et l'on suppose que les $\alpha - 1$ autres seront incrémentées de façon aléatoire uniforme.

Mauvaise paire : Pour une mauvaise paire, n'importe quelle clef peut être incrémentée (si la paire passe le crible) et les clefs incrémentées sont supposées être réparties de façon aléatoire uniforme sur l'ensemble des candidats.

On obtient donc, les probabilités suivantes

$$p = p^* \frac{\alpha - 1}{2^{n_k} - 1} + (\beta - p^*)\frac{\alpha}{2^{n_k}}, \tag{5.1}$$

$$p_0 = p^* + (\beta - p^*)\frac{\alpha}{2^{n_k}} \approx p^* + p. \tag{5.2}$$

L'approximation vient du fait que si le chiffrement n'est pas trop mal conçu, alors $\beta \gg p^*$. Une fois le calcul de α et β effectué, la démarche est la même que pour la cryptanalyse différentielle d'un SPN. On va donc commencer par présenter celle-ci puis on s'intéressera de façon commune à la suite de l'analyse.

5.1.2 Description de l'attaque sur le dernier tour

Le problème est que pour la plupart des chiffrements, il n'est pas possible d'utiliser l'attaque telle que présentée dans [BS91a]. On présente dans cette section une attaque sur le

dernier tour utilisant une différentielle qui peut s'appliquer, a priori, à tout chiffrement itératif. Pour attaquer $r+1$ tours de chiffrement, on utilise une différentielle sur les r premiers tours. On déchiffre alors les chiffrés sur un tour avec tous les candidats possibles pour la sous-clef du dernier tour. Pour la bonne sous-clef $\boldsymbol{k}_r^*$ on s'attend à observer le biais statistique dû à la différentielle et pour les mauvais candidats, on suppose, comme vu en section 1.5, que déchiffrer avec un mauvais candidat donne une distribution uniforme. On va donner précisément la forme de cette hypothèse de *répartition aléatoire par fausse clef* dans ce cas précis.

Hypothèse 5.2. *Hypothèse de répartition aléatoire par fausse clef (cryptanalyse différentielle). Pour tout δ_0, tout $\delta_r \neq 0$, pour toute clef $\boldsymbol{k}^*$ et pour tout candidat $\boldsymbol{k}_{r+1} \neq \boldsymbol{k}_{r+1}^*$,*

$$\Pr_{\boldsymbol{M}}\left[F^{-1}_{\boldsymbol{k}_{r+1}}(E_{\boldsymbol{k}^*}(\boldsymbol{M})) \oplus F^{-1}_{\boldsymbol{k}_{r+1}}(E_{\boldsymbol{k}^*}(\boldsymbol{M} \oplus \delta_0)) = \delta_r\right] = \frac{1}{2^s - 1}.$$

Une cryptanalyse différentielle se base sur une différentielle ayant une probabilité suffisamment grande par rapport à 2^{-s} pour pouvoir être utilisée comme distingueur. L'attaquant récupère alors $N/2$ paires de couples clair/chiffré $(\boldsymbol{m}_1^i, \boldsymbol{m}_2^i, \boldsymbol{c}_1^i, \boldsymbol{c}_2^i)$ avec donc $\boldsymbol{m}_2^j = \boldsymbol{m}_1^j \oplus \delta_0$ et $\boldsymbol{c}_j^i = E_{\boldsymbol{k}^*}(\boldsymbol{m}_j^i)$. L'attaque se déroule alors comme le montre l'algorithme 10.

Algorithme 10 : Cryptanalyse différentielle d'un SPN.

```
Entrée : N couples clair/chiffré;
Sortie : La valeur devinée de k*_{r+1};
début
    pour toutes les valeurs possibles k pour k*_{r+1} faire
        t_k ⟵ 0;
        pour 1 ≤ i ≤ N faire
            si F^{-1}_k(c^i_1) ⊕ F^{-1}_k(c^i_2) = δ_r alors
                t_k ⟵ t_k + 1;
            fin
        fin
    fin
fin
retourner k_{r+1} = argmax_k t_k ;
```

Bien évidemment, et comme pour la cryptanalyse linéaire, une telle attaque ne fonctionne pas en tant que telle du fait du nombre élevé de déchiffrements à effectuer. Cependant, on ne peut pas utiliser directement la même remarque que pour la cryptanalyse linéaire où il suffisait de déchiffrer les bits impliqués dans l'approximation linéaire. En effet, pour vérifier que l'on a bien la différence δ_r après r tours on a besoin de tous les bits de l'état.

Comme pour la cryptanalyse différentielle d'un réseau de Feistel, on va utiliser un crible. Par exemple, avoir une différence δ_r après r tours implique des restrictions sur l'ensemble des différences possibles en sortie du $r+1$-ième tour. Par exemple, si les bits de δ_r correspondant à certaines boîtes-S du $r+1$-ième tour sont nuls, alors les bits correspondant à ces boîtes dans la différence $\boldsymbol{c}_1 \oplus \boldsymbol{c}_2$ seront aussi nuls. Parmi les paires passant le crible, on distingue de nouveau deux types de paires.

Bonne paire : Pour une bonne paire, la bonne clef sera incrémentée et on suppose que les $\alpha - 1$ autres seront incrémentées de façon aléatoire uniforme.

Mauvaise paire : Pour une mauvaise paire, n'importe quelle clef peut être incrémentée sauf la bonne car la paire étant une mauvaise paire, le déchiffrement avec le bon candidat ne donne pas δ_r. Les clefs incrémentées sont supposées être réparties de façon aléatoire uniforme sur l'ensemble des candidats.

On a alors

$$p = p^* \frac{\alpha - 1}{2^{n_k} - 1} + (\beta - p^*) \frac{\alpha}{2^{n_k} - 1} = \frac{\beta\,\alpha - p^*}{2^{n_k} - 1}, \tag{5.3}$$

$$p_0 = p^*. \tag{5.4}$$

On voit là la différence avec l'attaque sur les réseau de Feistel qui est principalement la valeur de p_0. Ici, p_0 vaut p^* alors que dans le cas précédent on avait $p_0 \approx p^* + p$. Maintenant que l'on a vu comment calculer les probabilités p_0 et p pour ces deux attaques, on va s'intéresser à l'analyse de la complexité en données et de la probabilité de succès de ces attaques en fonction des valeurs de p_0 et p.

5.1.3 Quelques travaux sur l'analyse d'une cryptanalyse différentielle

On a vu que, bien que différentes, les deux cryptanalyses différentielles présentées se modélisent de la même façon.

Modèle 5.3. *(Modèle général de la cryptanalyse différentielle).*
On a 2^{n_k} compteurs $T_{\boldsymbol{k}}$ suivant des lois binomiales indépendantes. Pour une clef $\boldsymbol{k}^*$ fixée ayant servi à générer les échantillons, on a

$$\boldsymbol{t}_{\boldsymbol{k}^*} \sim \mathcal{B}(N, p_0) \quad \textit{et} \quad \boldsymbol{t}_{\boldsymbol{k} \neq \boldsymbol{k}^*} \sim \mathcal{B}(N, p).$$

Les valeurs de p_0 et p dépendent du type d'attaque comme l'on a pu s'en apercevoir. Leur calcul dérive des valeurs de α et β. Ces calculs sont, d'ailleurs, le point le plus détaillé dans les papiers de cryptanalyse différentielle. On verra dans la sous-section 5.1.4 que pour le cas des attaques sur le dernier tour, cette analyse est souvent fausse car trop grossière. On ne rentrera pas, ici, dans le détail des calculs de α et β et l'on suppose donc que l'on a déjà les valeurs des probabilités p_0 et p.

Une grandeur qui tient une importance particulière dans les papiers de cryptanalyse différentielle est le *rapport signal sur bruit*.

Définition 5.4. Le *rapport signal sur bruit* d'une cryptanalyse différentielle dont la caractéristique a pour probabilité p^*, qui distingue une clef parmi 2^{n_k} candidats, dont le crible ne garde qu'une proportion β des paires et incrémente α compteurs pour chacun d'entre elles, est défini par,

$$S_N \stackrel{\text{def}}{=} \frac{2^{n_k} \cdot p^*}{\alpha \cdot \beta}.$$

On remarque que dans le cas de la cryptanalyse d'un schéma de Feistel, ce rapport vaut à peu près $(p_0 - p)/p$ tandis que pour les SPN, ce rapport est de l'ordre de p_0/p. Cette grandeur porte donc bien son nom : elle quantifie le rapport entre le signal (comportement différentiel du chiffrement) correspondant aux bonnes paires et le bruit dû aux mauvaises paires.

Calcul de N et de P_S. Il est bien connu et accepté qu'une cryptanalyse différentielle utilisant une caractéristique de probabilité p^* a une complexité en données de l'ordre de $1/p^*$. Dans [BS91b], ainsi que dans les nombreux papiers qui suivirent, la complexité en données est estimée à l'aide de l'argument empirique suivant. Si le rapport S_N est de l'ordre de 1 ou 2, alors une valeur moyenne de 40 pour le compteur de la bonne clef semble suffisant et pour des rapports signal sur bruit plus grands, 3 à 4 bonnes paires sont assez. En ce qui concerne la probabilité de succès, on rappelle qu'il existe deux approches pour une cryptanalyse statistique (comme expliqué section 1.5). Pour l'approche de type examen, on trouve, dans certains papiers, l'utilisation d'une loi de Poisson afin de calculer la probabilité de succès de l'attaque. Aucune explication n'est donnée sur le choix du seuil (on suppose qu'il est choisi suite à une recherche exhaustive sur les premiers entiers). Cette approche se retrouve aussi dans [Gil97] : pour un fort rapport S_N, une loi de Poisson est utilisée pour obtenir une complexité en données de $N = 4.6/p^*$ qui correspond à une probabilité de 0.01 de rejeter la bonne clef avec un seuil 1.

Dans le cas de l'approche concours, seul un article de Selçuk [Sel08] semble s'être penché sur le problème. L'ennui est qu'il dérive une formule pour la probabilité de succès en utilisant une approximation normale de la loi binomiale qui est, comme il le dit lui-même, très peu appropriée à la situation. On discutera de ce problème dans le chapitre 6.

$$N = \frac{\left(\sqrt{S_N + 1}\Phi^{-1}(P_S) + \Phi^{-1}(1 - 2^{-a})\right)^2}{S_N \cdot p^*}. \tag{5.5}$$

5.1.4 Une analyse simplifiée pour la différentielle sur le dernier tour

On va maintenant s'intéresser au calcul de p dans le cas de la cryptanalyse différentielle sur le dernier tour présentée ci-avant dans le cas des SPN. On utilise une différentielle $(\delta_0, \delta_r, p^*)$ sur r tours pour en attaquer $r + 1$. On rappelle que β est la proportion de paires passant le crible et α le nombre de compteurs incrémentés par une de ces paires. On rappelle aussi que

$$p = \frac{\beta\,\alpha - p^*}{2^{n_k} - 1}.$$

Intuitivement, on sent que $p \approx 2^{-s}$. En effet, l'hypothèse de répartition aléatoire par fausse clef nous dit que déchiffrer partiellement avec un mauvais candidat est équivalent à une sortie aléatoire uniforme sur les $2^s - 1$ différences possibles. On a donc envie de dire que pour un couple et un mauvais candidat, on aura une probabilité $\frac{1}{2^s-1}$ d'obtenir δ_r en déchiffrant partiellement les deux chiffrés. Afin de le prouver proprement, définissons formellement la notion de crible différentiel.

Définition 5.5. Lors d'une cryptanalyse différentielle, on peut éliminer certains échantillons en étant sûr que ceux-ci ne correspondent pas à une bonne paire. Cette élimination est faite à l'aide d'un *crible différentiel*. Soit la fonction suivante

$$\begin{array}{rccc} \Psi_{\delta_r} : & \mathbb{F}_2^s \times \mathbb{F}_2^{n_k} & \to & \mathbb{F}_2^s \times \mathbb{F}_2^s \\ & (\boldsymbol{x}, \boldsymbol{k}) & \mapsto & (\boldsymbol{y}_1, \boldsymbol{y}_2) = (F_{\boldsymbol{k}}(\boldsymbol{x}), F_{\boldsymbol{k}}(\boldsymbol{x} \oplus \delta_r)) \end{array}$$

Alors un *crible différentiel* est un sous-ensemble $\mathcal{C}_{\delta_r} \subset \mathbb{F}_2^s \times \mathbb{F}_2^s$ vérifiant la propriété :

$$\mathrm{Im}(\Psi_{\delta_r}) \subset \mathcal{C}_{\delta_r}.$$

Dans cette définition, $\boldsymbol{x}$ correspond aux valeurs possibles de l'état à la sortie du tour r et $\boldsymbol{k}$ aux sous-clefs possibles pour le déchiffrement partiel sur le dernier tour. Ne pas effectuer de crible revient donc à prendre un ensemble $\mathcal{C}_{\delta_r} = \mathbb{F}_2^s \times \mathbb{F}_2^s$ et un crible optimal serait $\mathcal{C}_{\delta_r} = \mathrm{Im}(\Psi_{\delta_r})$.

Le nombre de compteurs incrémentés par une paire $(\boldsymbol{y}_1, \boldsymbol{y}_2)$ est égal au cardinal $\#\Psi_{\delta_r}^{-1}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ de son image réciproque. Or,

$$\sum_{(\boldsymbol{y}_1,\boldsymbol{y}_2)\in\mathcal{C}_{\delta_r}} \#\Psi_{\delta_r}^{-1}(\boldsymbol{y}_1, \boldsymbol{y}_2) = \sum_{(\boldsymbol{y}_1,\boldsymbol{y}_2)\in\mathrm{Im}(\Psi_{\delta_r})} \#\Psi_{\delta_r}^{-1}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2^{s+n_k}.$$

Le nombre moyen de compteurs incrémentés par une paire est donc

$$\alpha = \frac{2^{s+n_k}}{\#\mathcal{C}_{\delta_r}}. \tag{5.6}$$

Pour ce qui est de β, la proportion de paires passant le crible, on va faire l'hypothèse que les paires $(\boldsymbol{y}_1, \boldsymbol{y}_2)$ obtenues après $r+1$ tours de chiffrement sont réparties uniformément sur $\mathbb{F}_2^s \times \mathbb{F}_2^s$ privé des couples de la forme $(\boldsymbol{y}_1, \boldsymbol{y}_1)$ qui ne peuvent être obtenus pour $\delta_0 \neq 0$ étant donné qu'un chiffrement est une bijection. Cette hypothèse n'est pas abusive car si l'on arrivait à détecter un biais sur cette distribution alors on pourrait effectuer une attaque sur un tour de plus. On a alors

$$\beta = \frac{\#\mathcal{C}_{\delta_r}}{2^s(2^s-1)}. \tag{5.7}$$

Cela donne, au final, une formule pour la probabilité p qui ne dépend pas du crible. On va écrire ce résultat sous forme de théorème.

Théorème 5.6. *La probabilité qu'un mauvais candidat soit incrémenté par une paire lors d'une cryptanalyse différentielle sur le dernier tour d'un chiffrement agissant sur des blocs de m bits est*

$$p = \frac{2^{n_k} - p^*(2^s-1)}{(2^s-1)(2^{n_k}-1)} = \frac{1}{2^s-1} - \mathcal{O}\left(p^* 2^{-n_k}\right).$$

Démonstration : On rappelle que $p = \frac{\beta\,\alpha - p^*}{2^{n_k}-1}$. On remplace alors par les valeurs pour α et β données par (5.6) et (5.7). On obtient

$$p = \frac{\frac{2^{n_k}}{2^s-1} - p^*}{2^{n_k}-1} = \frac{1}{2^s-1} - \frac{p^* - 1/(2^s-1)}{2^{n_k}-1}.$$

Vu que p est de l'ordre de 2^{-s}, alors il faut que $p^* > 2^{-s}$ pour pouvoir détecter la bonne clef et donc, on a

$$p = \frac{1}{2^s-1} - \mathcal{O}\left(p^* 2^{-n_k}\right).$$

□

5.2 Variantes de la cryptanalyse différentielle

Cette section a pour objectif de présenter brièvement les variantes de la cryptanalyse différentielle. Le but principal est de permettre au lecteur de se faire une idée de la diversité des cryptanalyses statistiques et en particulier de la diversité des paramètres (p_0, p) de ces attaques. Pour plus de détails sur certaines de ces attaques, on pourra se référer à [Vid05].

5.2.1 Cryptanalyse différentielle tronquée

La cryptanalyse différentielle tronquée est une extension de la cryptanalyse différentielle dans laquelle on utilise un ensemble Δ_0 de différences en entrée et un ensemble de différences Δ_r en sortie.

$$\Pr_{\boldsymbol{M},\boldsymbol{K}}\left[E_{\boldsymbol{K}}(\boldsymbol{M}) \oplus E_{\boldsymbol{K}}(\boldsymbol{M} \oplus \delta_0) \in \Delta_r | \delta_0 \in \Delta_0\right] = p^*.$$

En général, et comme dans le papier d'origine proposant cette attaque [Knu95], on découpe les différences en morceaux correspondant aux boîtes-S de la couche de substitution ou bien aux blocs manipulés par le chiffrement. On choisit ensuite les différences de Δ_0 en fonction des propriétés de diffusion du chiffrement attaqué. La plupart du temps cela consiste à prendre les différences ayant des 0 sur certains morceaux et des valeurs arbitraires sur les autres. On a alors une probabilité p^* bien plus grande qu'en différentielle mais étant donnée la taille Δ_r, la probabilité p qu'une sortie aléatoire appartienne à Δ_r est aussi bien plus élevée. Le rapport p_0/p est en général bien moins élevé que pour la cryptanalyse différentielle standard.

5.2.2 Cryptanalyse différentielle impossible

La cryptanalyse différentielle impossible, comme son nom l'indique, repose sur une différentielle avec une probabilité nulle. La première apparition d'une telle attaque date de 1999 [BBS99].

$$\Pr_{\boldsymbol{M},\boldsymbol{K}}\left[E_{\boldsymbol{K}}(\boldsymbol{M}) \oplus E_{\boldsymbol{K}}(\boldsymbol{M} \oplus \delta_0) = \delta_r\right] = 0.$$

Le concept est le même que pour la cryptanalyse différentielle standard. On filtre les paires afin de ne garder que des paires pour laquelle on peut espérer avoir la différence δ_r après r tours. On déchiffre alors les paires qui passent le crible avec tous les candidats possibles et on incrémente les compteurs des clefs qui donnent une différence δ_r. La seule différence avec la cryptanalyse différentielle est qu'on ne garde que les clefs ayant un compteur nul car les autres sont forcément de mauvais candidats.

Remarque. Il est difficile de trouver une différentielle impossible qui ne provienne pas de propriétés structurelles du chiffrement. C'est pourquoi, dans la plupart des attaques, on a plutôt une cryptanalyse tronquée impossible. En effet, les chiffrements attaqués sont des réseaux de Feistel ou le standard AES qui manipulent des blocs de bits.

5.2.3 Cryptanalyse différentielle d'ordre supérieur

La cryptanalyse différentielle d'ordre supérieur repose sur la propriété suivante tout d'abord évoquée par Lai [Lai94] puis reprise par Knudsen [Knu95].

Proposition 5.7. *[Lai94] Soit f une fonction vectorielle de $\mathbb{F}_2^n$ dans $\mathbb{F}_2^m$ de degré d. Alors, pour tout sous-espace $\mathcal{V} \subset \mathbb{F}_2^n$ de dimension supérieure à d,*

$$\sum_{\boldsymbol{v} \in \mathcal{V}} F(\boldsymbol{x} + \boldsymbol{v}) = 0,$$

et ce, pour tout $\boldsymbol{x} \in \mathbb{F}_2^n$.

Cette somme est appelée dérivée d'ordre $\dim(\mathcal{V})$ relativement au sous-espace $\mathcal{V}$. L'idée est que si la fonction vectorielle correspondant à un nombre r de tours de chiffrement a un degré suffisamment faible, alors on peut distinguer ces r tours en calculant la dérivée de cette fonction pour un sous-espace de dimension supérieure à ce degré. Dans ce cas, la probabilité de la caractéristique statistique est 1 mais la taille de l'échantillon nécessaire est $2^{\deg(F)+1}$.

5.2.4 Cryptanalyse différentielle-linéaire

La cryptanalyse différentielle-linéaire est présentée pour la première fois par Langford et Hellman dans [LH94]. Le principe, comme son nom l'indique, est de combiner une différentielle à une approximation linéaire afin de mener une attaque. Pour la première attaque la différentielle choisie a une probabilité 1 mais les attaques ultérieures [BDK02, BDK03] utilisent des différentielles ayant une probabilité différente de 1.

La cryptanalyse différentielle-linéaire est une cryptanalyse statistique sur le dernier tour d'ordre 2. Une attaque visant $r+1$ tours de chiffrement utilisera une différentielle sur les r_d premiers tours et une approximation linéaire sur les r_l tours restant avec $r_d + r_l = r$. On va donc décomposer le chiffrement E en trois parties : $E = F_{r+1} \circ E_l \circ E_d$. On notera $E_{l,d} = E_l \circ E_d$ pour ne pas surcharger les expressions. Avec ces notations, on a donc les deux caractéristiques statistiques suivantes

$$\begin{aligned} \Pr_{\boldsymbol{M},\boldsymbol{K}}\left[E_d(\boldsymbol{M}) \oplus E_d(\boldsymbol{M} \oplus \delta_0) = \delta_{r_d}\right] &= p^*, \\ \Pr_{\boldsymbol{M}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \gamma, E_l(\boldsymbol{M})\rangle = 0\right] &= \frac{1}{2} + (-1)^{\langle \kappa, \boldsymbol{K}\rangle}\varepsilon. \end{aligned}$$

Si on regarde maintenant des paires ayant une différence δ_0, on obtient le système

$$\begin{cases} \Pr_{\boldsymbol{M}}\left[\langle \pi, E_d(\boldsymbol{M})\rangle \oplus \langle \gamma, E_{l,d}(\boldsymbol{M})\rangle = 0\right] &= \frac{1}{2} + (-1)^{\langle \kappa, \boldsymbol{K}\rangle}\varepsilon, \\ \Pr_{\boldsymbol{M}}\left[\langle \pi, E_d(\boldsymbol{M} \oplus \delta_0)\rangle \oplus \langle \gamma, E_{l,d}(\boldsymbol{M} \oplus \delta_0)\rangle = 0\right] &= \frac{1}{2} + (-1)^{\langle \kappa, \boldsymbol{K}\rangle}\varepsilon. \end{cases}$$

En supposant que les événements $< \pi, E_d(\boldsymbol{M}) > \oplus < \gamma, E_{d,l}(\boldsymbol{M}) >= 0$ et $\langle \pi, E_d(\boldsymbol{M} \oplus \delta_0)\rangle \oplus \langle \gamma, E_{l,d}(\boldsymbol{M} \oplus \delta_0)\rangle = 0$ sont indépendants, on peut en déduire que

$$\Pr_{\boldsymbol{M}}\left[\langle \pi, E_d(\boldsymbol{M}) \oplus E_d(\boldsymbol{M} \oplus \delta_0)\rangle = \langle \gamma, E_{l,d}(\boldsymbol{M}) \oplus E_{l,d}(\boldsymbol{M} \oplus \delta_0)\rangle\right] = \frac{1}{2} + 2\varepsilon^2.$$

Le produit scalaire de droite est obtenu en déchiffrant partiellement les paires de chiffrés avec les candidats possibles pour la clef du tour $r+1$. En ce qui concerne le premier produit, on va utiliser la propriété différentielle des r_d premiers tours. Si l'on a bien $E_d(\boldsymbol{M}) \oplus E_d(\boldsymbol{M} \oplus \delta_0) = \delta_{r_d}$, alors le produit $\langle \pi, E_d(\boldsymbol{M}) \oplus E_d(\boldsymbol{M} \oplus \delta_0)\rangle$ est bien égal à $\langle \pi, \delta_{r_d}\rangle$. Si ce n'est pas le cas, alors la probabilité que les produits coïncident est $\frac{2^{s-1}-1}{2^s}$. On obtient donc

$$\begin{aligned} \Pr_{\boldsymbol{M},\boldsymbol{K}}\left[\langle \pi, E_d(\boldsymbol{M}) \oplus E_d(\boldsymbol{M} \oplus \delta_0)\rangle = \langle \pi, \delta_{r_d}\rangle\right] &= p^* + (1-p^*)\frac{2^{s-1}-1}{2^s} \\ &\approx \frac{1}{2} + \frac{p^*}{2}. \end{aligned}$$

On obtient finalement l'approximation linéaire suivante

$$\Pr_{\boldsymbol{M},\boldsymbol{K}}\left[\langle \pi, \delta_{r_d}\rangle = \langle \gamma, E_{l,d}(\boldsymbol{M}) \oplus E_{l,d}(\boldsymbol{M} \oplus \delta_0)\rangle\right] = \frac{1}{2} + 2\,\varepsilon^2\,p^*. \tag{5.8}$$

On se retrouve donc dans le cadre d'une cryptanalyse linéaire sur le dernier tour utilisant l'approximation (5.8) de biais $2\,\varepsilon^2\,p^*$ où ε est le biais d'une approximation linéaire sur r_l tours et p^* la probabilité d'une différentielle sur r_d tours.

Remarque. Les attaques présentées dans [LH94, BDK02, BDK03] n'utilisent pas, en fait, une différentielle mais une différentielle tronquée. En effet, notons Δ_0 et Δ_{r_d} deux ensembles de différences tels que pour tous $\delta, \delta' \in \Delta_{r_d}$, on a $\langle \pi, \delta \rangle = \langle \pi, \delta' \rangle$ et

$$\Pr_{\boldsymbol{M},\boldsymbol{K}} \left[E_d(\boldsymbol{M}) \oplus E_d(\boldsymbol{M} \oplus \delta_0) \in \Delta_{r_d} | \delta_0 \in \Delta_0 \right] = p^*.$$

Alors on peut effectuer l'attaque présentée précédemment. Ceci a deux avantages. Le premier est que la probabilité p^* est bien plus élevée dans ce cas. Le second est que l'on va pouvoir obtenir un même nombre d'échantillons avec moins de couples clair/chiffré, un même message pouvant appartenir à plusieurs paires avec des différences en entrée différentes. Il faudra alors prendre soin de vérifier que cela ne pose pas de problèmes de dépendances.

5.3 Estimation de la probabilité d'une différentielle

L'estimation de la probabilité d'une différentielle n'est pas chose aisée. De plus, on a un phénomène de dépendance par rapport à la clef utilisée pour générer les échantillons. Cette section a pour but de donner les clefs d'une bonne estimation de la probabilité d'une différentielle. On verra entre autre que la situation est bien plus agréable qu'en cryptanalyse linéaire.

5.3.1 Probabilité différentielle et chemins différentiels

La façon la plus standard de calculer la probabilité d'une caractéristique statistique, on l'a vu pour la cryptanalyse linéaire avec le *Piling-up lemma*, est de chaîner des caractéristiques sur un tour. On définit la notion de *chemin différentiel* qui est la caractéristique statistique obtenue en chaînant des différentielles sur 1 tour.

Définition 5.8. Un chemin différentiel sur r tours d'un chiffrement itératif $E = F^r$ agissant sur des blocs de s bits est un $(r+1)$-uplet $\beta = (\beta_0, \ldots, \beta_r)$ d'éléments de $\mathbb{F}_2^s$. Sa probabilité p_β est

$$p_\beta \stackrel{\text{def}}{=} \Pr_{\boldsymbol{M},\boldsymbol{K}} \left[\forall 1 \leq i \leq r, F^i(\boldsymbol{M}) \oplus F^i(\boldsymbol{M} \oplus \beta_0) = \beta_i \right].$$

La probabilité d'une différentielle (δ_0, δ_r) s'exprime en fonction des probabilités des chemins différentiels correspondant à celle-ci.

Proposition 5.9. *Soit (δ_0, δ_r) une différentielle, alors sa probabilité p^* est la somme des probabilités des chemins différentiels $\beta = (\beta_0, \ldots, \beta_r)$ avec $\beta_0 = \delta_0$ et $\beta_r = \delta_r$.*

$$p^* = \sum_{\beta = (\delta_0, \beta_1, \ldots, \beta_{r-1}, \delta_r)} p_\beta. \tag{5.9}$$

Concrètement il devient impossible de calculer les probabilités de tous les chemins composant une différentielle quand le nombre de tours augmente. Cependant, la valeur obtenue en prenant un sous-ensemble de tous ces chemins est une borne inférieure sur la probabilité de la différentielle. Le problème d'estimer la probabilité d'une différentielle revient donc à calculer les probabilités de chemins différentiels. On va se pencher sur le sujet dans la prochaine sous-section mais faisons au préalable la remarque suivante sur la recherche de chemins différentiels.

Remarque sur la recherche de chemins différentiels. Dans la plupart des articles de cryptanalyse différentielle, les auteurs estiment la probabilité d'une différentielle avec la probabilité du meilleur chemin différentiel correspondant. Il est pourtant rapide et facile de trouver un certain nombre de chemins différentiels correspondant à une différentielle en utilisant une version adaptée à la cryptanalyse différentielle de l'algorithme 7 de « Séparation et Évaluation » présenté pour la cryptanalyse linéaire.

Par exemple, la cryptanalyse de PRESENT par Wang [Wan08] utilise des différentielles pour lesquelles elle estime la probabilité à 2^{-62}. Le tableau 5.2 contient les bornes obtenues sur la probabilité de la première des 24 différentielles utilisées par Wang en fonction du temps de calcul de l'algorithme 7 implémenté par mes soins (et donc pas forcément optimisé au maximum comme pourrait le faire un spécialiste) et de la borne donnée sur la probabilité des chemins recherchés. On voit que l'on peut rapidement passer d'une attaque

Borne sur p_β	Temps de calcul algorithme 7	Borne sur p^* obtenue
meilleur chemin	-	$2^{-62.00}$
2^{-64}	10 s	$2^{-60.00}$
2^{-66}	2 m	$2^{-58.91}$
2^{-70}	1 h	$2^{-57.67}$
2^{-73}	16 h	$2^{-57.53}$

TABLE 5.2 – Bornes sur la probabilité d'une différentielle utilisée dans [Wan08].

nécessitant l'ensemble des paires possibles à une attaque avec une complexité en données bien plus raisonnable pour un coût de quelques heures de calcul.

5.3.2 Chiffrements de Markov

En utilisant l'égalité (5.9), le problème de calculer la probabilité d'une différentielle revient donc à calculer les probabilités de chemins différentiels. Ceux-ci sont, comme indiqué plus haut, obtenus en chaînant des différentielles sur 1 tour. Pour cela, on suppose que les tours sont indépendants. Nous allons préciser ce que l'on entend par tours indépendants. C'est la notion introduite dans [LMM91] de *chiffrement de Markov*.

Définition 5.10. *(Chiffrement de Markov).* Un chiffrement itératif $E = F^r$ est un *chiffrement de Markov* si, pour une distribution uniforme de la sous-clef $\boldsymbol{K}$, on a la propriété

$$\Pr_{\boldsymbol{K}}\left[F_{\boldsymbol{K}}(\boldsymbol{M}) \oplus F_{\boldsymbol{K}}(\boldsymbol{M} \oplus \delta_0) = \delta | \boldsymbol{M} = \boldsymbol{m}\right] = \Pr_{\boldsymbol{K}}\left[F_{\boldsymbol{K}}(\boldsymbol{M}) \oplus F_{\boldsymbol{K}}(\boldsymbol{M} \oplus \delta_0) = \delta\right].$$

On a alors le théorème suivant

Théorème 5.11. *[LMM91]. Si un chiffrement itératif de r tours est un chiffrement de Markov et que les r sous-clefs sont indépendantes et uniformément distribuées, alors la probabilité d'un chemin $\beta = (\beta_0, \beta_1, \ldots, \beta_r)$ vaut*

$$p_\beta = \prod_{i=1}^{r} \Pr_{\boldsymbol{M},\boldsymbol{K}}\left[F_{\boldsymbol{K}}(\boldsymbol{M}) \oplus F_{\boldsymbol{K}}(\boldsymbol{M}') = \beta_i | \boldsymbol{M} \oplus \boldsymbol{M}' = \beta_{i-1}\right].$$

Discussion. Beaucoup de chiffrements sont des chiffrements de Markov. Par exemple pour un SPN, la clef de tour est XORée à l'état et a la même taille que celui-ci. Cela implique, si celle-ci est uniformément distribuée sur l'ensemble des valeurs possibles, que connaître $\boldsymbol{m}$ n'apporte aucune information car après l'addition de la clef, on a une entropie maximale sur la valeur de l'état. Cependant, l'hypothèse sur la clef faite dans le théorème 5.11 est abusive car pour que celle-ci soit vraie, il faudrait que le nombre de bits de la clef maître soit égal à r fois le nombre de bits d'une clef de tour. On a, par exemple, mis en avant dans [BG10] des chemins différentiels ayant une probabilité différente de celle calculée par chaînage. Cela s'explique justement par le fait que les $i-1$ premières différences impliquent certaines relations entre les bits de l'état et les bits de la clef au tour i. Ce genre de phénomène s'accentue avec le nombre de tours (si l'on n'augmente pas la taille de la clef) et disparaît avec des clefs de tour indépendantes les unes des autres.

Cependant, si on somme les probabilités de tous les chemins possibles, on obtient 1 ce qui signifie que si certains chemins ont des probabilités sous-estimées, d'autres ont des probabilités surestimées. On a vu sur de petits exemples, dans [BG10], que lorsque l'on prend beaucoup de chemins correspondant à une différentielle, les différents comportements se compensent et on obtient, en général, une bonne estimation de la probabilité de la différentielle.

5.3.3 Probabilité à clef fixée

Jusqu'à présent, on a vu comment calculer les probabilités des différentielles et chemins différentiels qui sont définies en utilisant la notation $\Pr_{\boldsymbol{M},\boldsymbol{K}}[\cdot]$. Cependant, pour une clef $\boldsymbol{k}$ fixée, ce qui est le cas quand on cherche à effectuer une cryptanalyse, cette probabilité n'est pas forcément la même. L'hypothèse selon laquelle ces probabilités ne dépendent pas de la clef est appelée *hypothèse d'équivalence à clef fixée.*

Hypothèse 5.12. *Hypothèse d'équivalence à clef fixée (différentielle).*
Soit une différentielle (δ_0, δ_r) d'un chiffrement E. Pour toute clef $\boldsymbol{k} \in \mathbb{F}_2^{n_k}$,

$$\Pr_{\boldsymbol{M}}\left[E_{\boldsymbol{k}}(\boldsymbol{M}) \oplus E_{\boldsymbol{k}}(\boldsymbol{M} \oplus \delta_0) = \delta_r\right] \approx \Pr_{\boldsymbol{M},\boldsymbol{K}}\left[E_{\boldsymbol{K}}(\boldsymbol{M}) \oplus E_{\boldsymbol{K}}(\boldsymbol{M} \oplus \delta_0) = \delta_r\right].$$

Pour la cryptanalyse différentielle, [DR07] introduit un modèle qui permet de prévoir la distribution des probabilités à clef fixée. Des expérimentations menées avec Céline Blondeau sur PRESENT [BG10] montre que l'hypothèse 5.12 n'est effectivement pas vérifiée et que le modèle présenté dans [DR07] prédit bien le comportement de ces probabilités.

Pour préciser ce dernier point, l'on va introduire quelques notations.

Pour une différentielle $\delta = (\delta_0, \delta_r)$ fixée, soit Ω l'ensemble des couples $(\boldsymbol{m}, \boldsymbol{k}) \in \mathbb{F}_2^s \times \mathbb{F}_2^{n_k}$ quotienté par la relation d'équivalence suivante

$$(\boldsymbol{m}_1, \boldsymbol{k}_1) \sim (\boldsymbol{m}_2, \boldsymbol{k}_2) \Leftrightarrow \boldsymbol{m}_1 \oplus \boldsymbol{m}_2 = \delta_0 \text{ et } \boldsymbol{k}_1 = \boldsymbol{k}_2. \tag{5.10}$$

On note Ω_δ l'ensemble des couples $(\boldsymbol{m}, \boldsymbol{k}) \in \Omega$ tels que $E_{\boldsymbol{k}}(\boldsymbol{m}) \oplus E_{\boldsymbol{k}}(\boldsymbol{m} \oplus \delta_0) = \delta_r$. Une paire appartient à Ω_δ si et seulement si sa paire équivalente appartient aussi à Ω_δ. C'est pourquoi on a quotienté par la relation d'équivalence (5.10)

Pour une clef $\boldsymbol{k}^*$ fixée, la probabilité de la différentielle δ est

$$\begin{aligned} p^*(\boldsymbol{k}^*) &\overset{\text{def}}{=} \Pr_{\boldsymbol{M}}\left[E_{\boldsymbol{k}^*}(\boldsymbol{m}) \oplus E_{\boldsymbol{k}^*}(\boldsymbol{m} \oplus \delta_0) = \delta_r\right] \\ &= \frac{\#\{(\boldsymbol{m}, \boldsymbol{k}) \in \Omega_\delta | \boldsymbol{k} = \boldsymbol{k}^*\}}{2^{s-1}}. \end{aligned}$$

On sait que $\#\Omega_\delta = p^* \cdot 2^{s-1+n_k}$. Le modèle suivant proposé dans [DR07] est le *modèle d'échantillonnage*.

Modèle 5.13. *Dans le modèle d'échantillonnage, on voit Ω comme un ensemble de 2^{s-1+n_k} boules. Pour une différence δ, les boules de Ω_δ sont noires et les autres blanches. Fixer une clef est alors vu comme prendre 2^{s-1} boules simultanément, aléatoirement et uniformément sur Ω. Le nombre de boules noires dans les 2^{s-1} boules tirées suit alors une loi hypergéométrique. On a donc*

$$\Pr_{\boldsymbol{K}}\left[p^*(\boldsymbol{K}) = \frac{\omega}{2^{s-1}}\right] = \frac{\binom{p^*\,2^{s-1+n_k}}{\omega} \cdot \binom{(1-p^*)\,2^{s-1+n_k}}{2^{s-1}-\omega}}{\binom{2^{s-1+n_k}}{2^{s-1}}}. \tag{5.11}$$

Étant donnés les paramètres de la cryptanalyse différentielle, on peut approximer la distribution de $2^{s-1} \cdot p^*(\boldsymbol{K})$ par une binomiale $\mathcal{B}(2^{s-1}, p^*)$.

Si on note $P_s(p)$ la probabilité de succès d'une attaque pour une probabilité différentielle à clef fixée de p[3], alors, le calcul de la probabilité de succès globale d'une attaque se fait en utilisant la formule suivante

$$P_S \stackrel{\text{def}}{=} \sum_{0 \le \omega \le 2^{s-1}} P_s\left(\frac{\omega}{2^{s-1}}\right) \cdot \left[p^{*\omega}(1-p^*)^{2^{s-1}-\omega}\binom{2^{s-1}}{\omega}\right]. \tag{5.12}$$

3. Le lecteur trouvera une discussion sur la formule à utiliser pour $P_S(p)$ dans le chapitre 6.

Chapitre 6

Analyse des cryptanalyses statistiques

Sommaire

Ce chapitre est essentiellement composé des résultats des travaux effectués conjointement avec Céline Blondeau et Jean-Pierre Tillich sur le sujet [BG09b, BGT10]. Il s'agit de porter un regard global sur les cryptanalyses statistiques afin de donner des formules générales, c'est-à-dire valables pour toutes ces attaques, pour estimer les performances de celles-ci. Dans un premier temps l'on présentera le contexte et le modèle dans lequel les résultats sont valables (section 6.1). Nous discuterons ensuite des travaux existants dans la section 6.2 et mettrons en avant ce qui a été l'élément déclencheur de ce travail : l'absence de formules génériques

pour l'analyse des cryptanalyses statistiques. Nous présenterons ensuite l'outil principal utilisé qui est une approximation des queues de la distribution binomiale qui est pertinente quels que soient les paramètres de l'attaque (section 6.3). Nous nous intéresserons ensuite à deux paradigmes inhérents au problème. La première approche abordée à la section 6.4 nous donnera des résultats sur la complexité en données d'une attaque quand la seconde utilisera ces résultats pour dériver une formule pour la probabilité de succès d'une attaque, ce que nous explicitons dans la section 6.5.

6.1 Contexte

On s'arrêtera ici au cas des cryptanalyses statistiques que j'appellerais « simples » par opposition aux cryptanalyses multiples ou multidimensionnelles. Plus précisément, on se place dans le modèle défini ci-après.

Modèle 6.1. *Cryptanalyses statistiques simples*
L'information extraite lors de la cryptanalyse est une statistique Σ qui est composée de compteurs $\Sigma_{\boldsymbol{k}}$: un par candidat $\boldsymbol{k}$. Ces compteurs correspondent au nombre de fois qu'un phénomène est apparu pour un échantillon donné et ce candidat.

On note $X^i_{\boldsymbol{k}}$ la variable valant 1 si le phénomène est apparu pour l'échantillon numéro i et le candidat $\boldsymbol{k}$. Les variables $X^i_{\boldsymbol{k}}$ suivent donc une loi de Bernoulli. On suppose, comme dans la plupart des cryptanalyses statistiques, que l'hypothèse de répartition aléatoire par fausse clé (voir hypothèse 6.2) est vérifiée et donc, on définit deux probabilités p_0 et p comme

$$p_0 \stackrel{\text{def}}{=} \Pr\left[X^i_{\boldsymbol{k}} = 1 | \boldsymbol{k} = \boldsymbol{k}^*\right] \quad , \quad p \stackrel{\text{def}}{=} \Pr\left[X^i_{\boldsymbol{k}} = 1 | \boldsymbol{k} \neq \boldsymbol{k}^*\right].$$

On a alors $\Sigma = (\Sigma_0, \ldots, \Sigma_{2^{n_k}-1})$ la statistique extraite des N échantillons avec

$$\Sigma_k \stackrel{\text{def}}{=} \sum_{i=1}^{N} X^i_{\boldsymbol{k}}.$$

On se place dans le cas où les échantillons sont tirés de façon indépendante ce qui implique l'indépendance des variables $X^i_{\boldsymbol{k}}$. Les Σ_k suivent donc une loi binomiale ayant pour paramètres (N, p_0) ou (N, p) selon que $\boldsymbol{k}$ soit égal à $\boldsymbol{k}^*$ ou non.

Avec ces notations, on peut exprimer autrement l'hypothèse de répartition aléatoire par fausse clé (hypothèse 1.7) qui est faite lors d'une cryptanalyse sur le dernier tour pour pouvoir entrer dans le cadre du modèle 6.1.

Hypothèse 6.2. *Hypothèse de répartition aléatoire par fausse clé (générale).*

$$\forall i,\ \boldsymbol{k} \neq \boldsymbol{k}^*,\ \boldsymbol{k}' \neq \boldsymbol{k}^*, \quad \Pr\left[X^i_{\boldsymbol{k}} = 1\right] = \Pr\left[X^i_{\boldsymbol{k}'} = 1\right] \stackrel{\text{def}}{=} p.$$

Dans ce contexte, on s'intéresse à l'étude des performances d'une attaque. On rappelle les quatre caractéristiques permettant d'évaluer une cryptanalyse.

- la *complexité en données*,
- la *complexité en temps*,
- la *complexité en mémoire*,
- la *probabilité de succès*.

On s'intéresse ici à ces complexités sans prendre en compte les possibles astuces et améliorations pouvant être utilisées pour un type d'attaque ou un chiffrement particulier. Pour la plupart des cryptanalyses statistiques, la complexité en mémoire est essentiellement due au stockage de la statistique Σ et c'est pourquoi on n'abordera pas ce sujet dans ce document. Ce qui nous intéresse plus est le lien qu'il existe entre les trois caractéristiques restantes.

De la complexité en données N va dépendre la quantité d'information obtenue. Le choix de la taille ℓ de la liste $\mathcal{L}$ de candidats gardés pour la recherche exhaustive va donc permettre d'effectuer un compromis entre la complexité en temps et la probabilité de succès P_S. L'objectif est donc de réussir à quantifier le lien entre ces trois variables et d'avoir des formules permettant d'estimer l'une d'elles en fonction des deux autres.

Remarque sur la complexité en données. On veut avoir une approche globale du problème et donc éviter tous les détails spécifiques à telle ou telle attaque. Donc, quand on parle de la complexité en données, on ne se préoccupe que du nombre d'échantillons [1] et non du nombre de clairs nécessaires à une attaque. L'art de diminuer le nombre de couples clair/chiffré nécessaire à l'obtention d'un certain nombre d'échantillons est en effet très dépendant de la structure des chiffrements et des phénomènes statistiques considérés.

6.1.1 Deux paradigmes

Comme l'on a vu dans la section 1.5, il y a deux approches au problème que nous traitons l'une après l'autre dans ce chapitre. Voici un bref descriptif de ces deux approches dans le contexte du modèle 6.1.

Seuil fixé : ≪ examen ≫. Dans cette approche, on se fixe un seuil T sur les compteurs, à partir duquel, on considère le candidat correspondant comme valable. La liste des candidats est alors

$$\mathcal{L} = \{\boldsymbol{k},\ \Sigma_{\boldsymbol{k}} \geq T\}.$$

On a donc une expression simple de la probabilité de succès qui est

$$P_S = \Pr\left[\Sigma_{\boldsymbol{k}^*} \geq T\right].$$

La taille de la liste ℓ est non bornée (si ce n'est par le nombre total de candidats 2^{n_k}). On a, cependant, de l'information sur cette taille car la probabilité d'avoir $\ell = i$ est la probabilité qu'au moins i mauvais candidats aient leur statistique supérieure ou égale à T. La variable aléatoire L correspondant à ℓ suit donc une distribution binomiale ayant pour paramètres $2^{n_k} - 1$ et $\Pr\left[\Sigma_{\boldsymbol{k}\neq\boldsymbol{k}^*} \geq T\right]$.

La problématique, ici, est de trouver un couple (N, T) minimal permettant d'obtenir des probabilités P_S et $\Pr\left[\Sigma_{\boldsymbol{k}\neq\boldsymbol{k}^*} \geq T\right]$ inférieures à certaines bornes fixées par le cryptanalyste.

Taille de liste fixée : ≪ concours ≫. Ici on fixe la taille ℓ de la liste. Moralement, cela revient à fixer le seuil T a posteriori afin que le nombre de candidats ayant un compteur supérieur ou égal à T soit exactement ℓ. Cette approche est la plus souvent utilisée vu qu'elle permet de borner la complexité en temps. L'inconvénient, est que le calcul de la probabilité de succès va faire intervenir des statistiques d'ordre. Si l'on note Σ^*_ℓ la variable correspondant

1. On rappelle que pour une cryptanalyse statistique d'ordre d un échantillon contient d chiffrés.

au ℓ-ème meilleur compteur parmi les $2^{n_k} - 1$ mauvais candidats, alors la probabilité de succès est

$$P_S = \Pr\left[\Sigma_\ell^* < \Sigma_{\boldsymbol{k}^*}\right].$$

La problématique ici est le calcul précis de P_S et l'obtention d'une formule pour N quand ℓ et P_S sont fixés.

6.2 Travaux existants

L'analyse des cryptanalyses statistiques a été effectuée de façon ad-hoc à leur création. Nous rappelons ici les travaux effectués pour les deux cryptanalyses statistiques principales. Pour plus de détails on se reportera aux chapitres dédiés à ces cryptanalyses (chapitre 4 et chapitre 5).

6.2.1 Cryptanalyse différentielle

Le contexte est le suivant : on a deux probabilités p_0 et p telles que p_0 et p sont proches de 0 et $p_0 \gg p$. Le dernier point est le plus important : p_0 est supérieur p mais de combien ? On a vu que dans les papiers d'origine [BS91a, BS91b], ainsi que dans [Gil97], on nous dit que pour un fort rapport signal sur bruit il suffit que la différentielle apparaisse 3 à 4 fois pour le bon candidat alors que pour un signal plus faible une quarantaine de bonnes paires semble convenir. Dans les papiers plus récents, on s'intéresse plus à l'approche à taille de liste fixée. Dans ce cas, le calcul de P_S est effectué en tirant parti de la formule donnée par Selçuk [Sel08] qui utilise une approximation normale des compteurs.

6.2.2 Cryptanalyse linéaire

Dès le départ, Matsui utilise une approximation normale de la loi binomiale afin de calculer la probabilité de succès de son attaque [Mat94b]. C'est cette approximation qui sera utilisée dans tous les travaux ultérieurs car l'approximation est bonne dans ce contexte. La formule donnée par Matsui pour le succès des cryptanalyses linéaires de type 2 est assez complexe mais peut être évaluée numériquement.

Le premier travail théorique sur l'évaluation de N et P_S précisant les résultats de Matsui est dû à Pascal Junod [Jun01] qui propose une formule simplifiée pour la probabilité de succès en utilisant les statistiques d'ordre et la fonction bêta incomplète. Il effectue, de plus, des expérimentations de l'attaque de Matsui afin de vérifier son théorème. Sa formule coïncide avec les valeurs trouvées par Matsui mais il semble qu'elle soit un peu pessimiste par rapport aux 42 expériences. On verra une explication de ce phénomène dans le chapitre 8.

Le second travail de Junod a été d'optimiser l'utilisation de deux approximations comme proposé par Matsui [Mat94a] en utilisant la théorie des tests binaires d'hypothèses [Jun03, JV03]. Cela n'entre pas dans la problématique ici vu que l'on se concentre sur les cryptanalyses simples mais les outils utilisés sont semblables.

On peut noter un travail asymptotique sur le sujet [BJV04, BV08] dont on aura l'occasion de reparler vu qu'il apporte un point de vue plus général sur les attaques statistiques.

Enfin, on prendra soin de ne pas oublier l'article d'Ali Aydin Selçuk [Sel08] car celui-ci propose une formule fondée sur l'approximation normale des compteurs pour la cryptanalyse linéaire tout comme pour la cryptanalyse différentielle. On va d'ailleurs s'attarder tout de suite sur ce travail.

6.2.3 Discussion sur l'approximation normale

Comme le dit Selçuk dans son article sur le sujet [Sel08], l'utilisation de la loi normale comme estimation de la distribution binomiale n'est pas conseillée pour le cas de la cryptanalyse différentielle. Cependant, la formule donnée dans cet article pour le calcul de la probabilité de succès est simple à calculer et, de fait, plusieurs articles de cryptanalyse différentielle y font appel pour calculer le taux de réussite de leur attaque. Cette sous-section a pour objectif de persuader le lecteur que l'utilisation de la loi normale en lieu et place d'une distribution binomiale n'est pas toujours une chose pertinente.

Commençons par revenir sur le pourquoi de l'approximation de la distribution binomiale par une gaussienne. On le rappelle, une variable suivant une distribution binomiale est une somme de variables de Bernoulli indépendantes. On se trouve donc dans le cadre du théorème central limite qui nous dit que la distribution d'une telle somme converge vers une loi normale quand le nombre de termes de la somme tend vers l'infini. En cryptanalyse, le nombre de termes est le nombre d'échantillons disponibles. Autant dire que cela semble amplement suffisant pour appliquer le théorème. Ce raisonnement n'est malheureusement pas tout à fait exact. Regardons plus précisément ce que l'on sait sur ce théorème et intéressons-nous à la version du théorème central limite qui majore la vitesse de convergence : le théorème de Berry-Esséen (théorème A.11). On va regarder quelle est la valeur de la borne sur la vitesse de convergence pour les deux principales cryptanalyses statistiques.

Cryptanalyse linéaire. Dans le cas de la cryptanalyse linéaire, on a vu dans la sous-section 4.1.4 que la borne du théorème était de l'ordre de $C\,\frac{\rho}{\sigma^3\sqrt{N}} = \mathcal{O}(\varepsilon)$.

Cryptanalyse différentielle. Regardons maintenant ce qui se passe dans le cas de la cryptanalyse différentielle. Les variables X_i sont définies par

$$X_i \overset{\text{def}}{=} \begin{cases} 1-p & \text{avec probabilité } p, \\ -p & \text{avec probabilité } 1-p. \end{cases}$$

On a alors $\sigma^2 = p(1-p)$ et $\rho = p(1-p)(1-2p+2p^2)$.

$$\begin{aligned} \frac{\rho}{\sigma^3} &= \left[p(1-p)(1-2p+2p^2)\right] \cdot \left[p(1-p)\right]^{-3/2} \\ &= \frac{1-2p+2p^2}{\sqrt{p(1-p)}} \leq \frac{1-2p}{\sqrt{p(1-p)}} \end{aligned}$$

Dans le chapitre 4, on a vu que $N \geq 1/p$ et donc, $C\,\frac{\rho}{\sigma^3\sqrt{N}} \leq \frac{1-2p}{\sqrt{(1-p)}}\,C$.

Pour des valeurs de p proches de 0, comme c'est le cas pour la cryptanalyse différentielle, le coefficient devant la constante est de l'ordre de 1. On note donc qu'à la différence de la cryptanalyse linéaire, le théorème de Berry-Esséen ne nous dit pas que la convergence est bonne. Il ne nous dit pas non plus qu'elle est mauvaise vu que l'on n'a qu'une borne supérieure mais c'est déjà un premier indice.

Voici quelques éléments de réflexions. Les courbes de la figure 6.1 représentent la distribution binomiale et son estimation par une distribution normale pour des paramètres de cryptanalyse différentielle (i.e. $N \approx 1/p_0$) et de cryptanalyse linéaire (i.e. $N \approx 1/\varepsilon^2$ et $p_0 = 1/2+\varepsilon$). Plus précisément, les paramètres utilisés pour tracer ces courbes sont

– pour la cryptanalyse différentielle : $p_0 = 2^{-16}$ et $N = 2^{16}$;

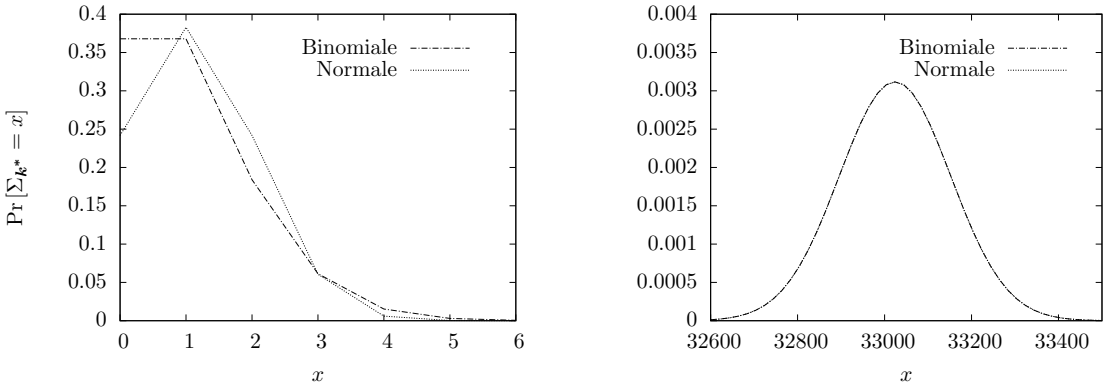

FIGURE 6.1 – Comparaison de la distribution de Σ_{k^*} et son approximation normale pour la cryptanalyse différentielle (gauche) et linéaire (droite).

– pour la cryptanalyse linéaire : $p_0 = 1/2 + 2^{-8}$ et $N = 2^{16}$.

On voit bien que dans le cas de la différentielle on est un peu loin du compte tandis que pour la cryptanalyse linéaire les courbes sont superposées. Mais cela n'est qu'un premier aperçu car ce qui nous intéresse quand on regarde la probabilité de succès d'une cryptanalyse statistique, c'est la probabilité que le compteur d'une mauvaise clef ait une valeur très largement supérieure à son espérance ou bien la probabilité que le compteur de la bonne clef soit trop petit. On veut donc utiliser les queues de la distribution binomiale. La figure 6.2 est un zoom sur la queue droite de la binomiale. Comme la décroissance de la probabilité est exponentielle, on doit passer en échelle logarithmique pour les ordonnées si l'on veut voir quelque chose. On

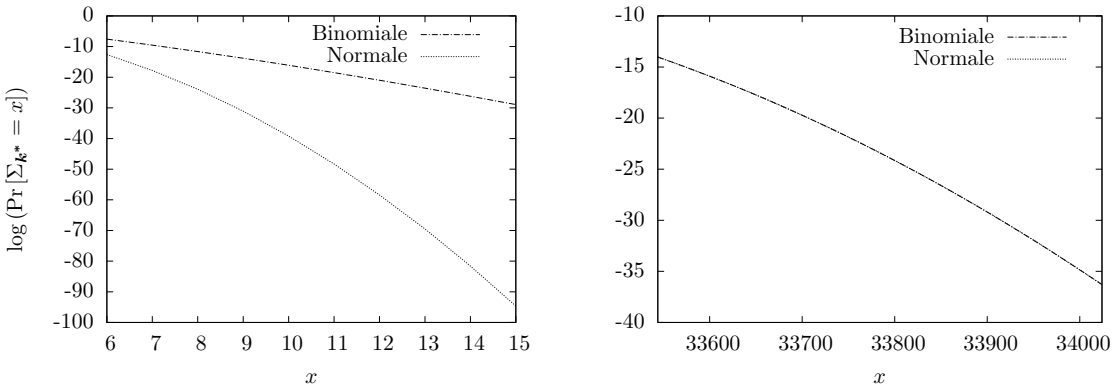

FIGURE 6.2 – Comparaison de la queue de la distribution de Σ_{k^*} et son approximation normale pour la cryptanalyse différentielle (gauche) et linéaire (droite).

voit bien mieux ici le fait que la loi normale ne convient pas à l'utilisation qui en est faite dans le cas de la cryptanalyse différentielle.

6.3 Estimation générale de la loi binomiale

On a vu que, selon les cryptanalyses, l'approximation de la distribution binomiale qui convient n'est pas la même. On peut même imaginer que cela va aussi dépendre des paramètres

de l'attaque. C'est ce que l'on va voir dans cette première sous-section.

6.3.1 Motivations

Prenons comme exemple la cryptanalyse différentielle tronquée, on a des probabilités plus élevées qu'en cryptanalyse différentielle mais plus proches. Le choix de l'approximation à utiliser est alors moins clair comme le montre la figure 6.3 qui compare les approximations normale et de Poisson pour deux jeux de paramètres pouvant provenir d'une cryptanalyse tronquée. Les paramètres correspondant aux courbes de gauche sont $p_0 = 1.01 \cdot 2^{-4}$, $p = 1 \cdot 2^{-4}$ et donc la valeur de N [2] est 2^{16}. On voit, dans ce cas, que l'approximation par la

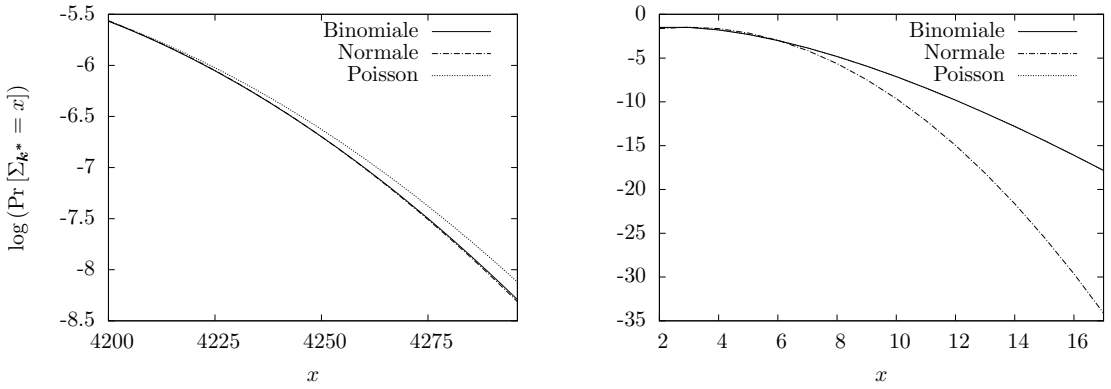

FIGURE 6.3 – Comparaison des approximations normale et de Poisson pour des paramètres de cryptanalyse différentielle tronquée.

loi normale convient mieux que l'utilisation de la distribution de Poisson. Cependant, si on regarde le second jeu de paramètres, $p_0 = 1.5 \cdot 2^{-19}$, $p = 1 \cdot 2^{-19}$ et $N = 2^{20}$, on s'aperçoit que c'est maintenant l'approximation par la loi normale qui est hors sujet. À noter que même si l'approximation de Poisson pour le premier jeu de paramètres semble moins mauvaise que la gaussienne pour le second jeu, elle n'en reste pas moins mauvaise (cette différence peut induire jusqu'à un facteur 2 d'erreur sur les probabilités du type $\Pr[\Sigma_{k^*} \leq x]$).

Mais on peut faire encore pire : la figure 6.4 montre que l'on peut trouver des paramètres de cryptanalyse tronquée pour lesquels aucune des deux approximations n'est valide. On a pris, pour cela les probabilités $p_0 = 1.05 \cdot 2^{-4}$, $p = 1 \cdot 2^{-4}$ et un nombre d'échantillons $N = 2^{12}$.

On voit donc qu'il serait très intéressant d'avoir une façon globale d'aborder le problème sans devoir chercher quelle approximation est valide pour tels paramètres. En fait, la queue de la fonction de répartition d'une variables binomiale provenant de N échantillons est le produit d'une composante polynomiale et d'une seconde exponentielle

$$\Pr[\Sigma_{k^*} \leq x] = Q(N)\, e^{-\Gamma N}.$$

Le facteur exponentiel est bien connu des statisticiens car largement utilisé dans l'étude des tests binaires d'hypothèses. Dans la communauté cryptographique, on le retrouve dans les travaux asymptotiques [Jun03, BJV04, BV08]. Mais pour une utilisation cryptographique, le

2. La façon dont a été calculé N est le sujet de la prochaine section. Le lecteur suspicieux pourra venir vérifier que les N choisis sont raisonnables une fois la section 6.4 lue.

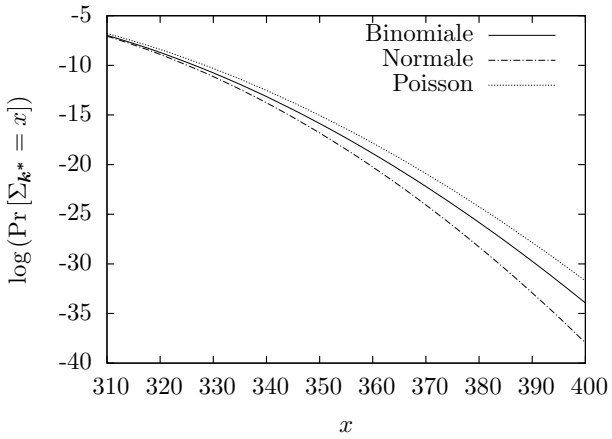

FIGURE 6.4 – Exemple de paramètres pour lesquels les deux approximations standards ne sont pas pertinentes.

coefficient polynômial a une grande importance et le négliger peut amener à des résultats totalement faussés. Une estimation de ce coefficient polynômial est donné dans [BJV04] mais seulement pour le cas particulier de la cryptanalyse linéaire. On va donc chercher à estimer, aussi, le facteur polynomial en gardant à l'esprit que la formule finale doit être valable pour tout type de paramètres cryptographiques. On utilisera ensuite ce résultat dans la section 6.4 pour obtenir une formule pour la valeur de N à prendre pour effectuer une cryptanalyse. On comparera alors les résultats obtenus avec notre formule à ceux obtenus avec les deux approximations classiques et les travaux suscités.

6.3.2 Estimation numérique de la fonction de répartition de la binomiale

Commençons tout d'abord par rappeler la définition de la divergence de Kullback-Leibler qui a été présentée à la section 3.1 (ici dans le cas d'une distribution discrète).

Définition 6.3. Soient $\mathcal{P}$ et $\mathcal{Q}$ deux distributions définies sur un même support $\mathcal{X}$. Alors, la *divergence de Kullback-Leibler* entre ces deux distributions est

$$D(\mathcal{P}||\mathcal{Q}) \stackrel{\text{def}}{=} \sum_{x\in\mathcal{X}} \Pr_{\mathcal{P}}[X=x] \ln\left(\frac{\Pr_{\mathcal{P}}[X=x]}{\Pr_{\mathcal{Q}}[X=x]}\right).$$

Dans le cas où l'une des probabilités est nulle, on utilise les conventions naturelles $0\ln\frac{0}{p}=0$ et $p\ln\frac{p}{0}=\infty$.

Cette divergence apparaît naturellement dans des problématiques d'hypothèses binaires où elle quantifie, d'une certaine manière, l'éloignement de deux distributions (d'où son appellation abusive de distance). Dans ce chapitre, nous appliquerons cette divergence à des distributions de Bernoulli.

Définition 6.4. On notera $D(p||q)$ la divergence de Kullback-Leibler entre deux distributions de Bernoulli de paramètres p et q.

$$D(p||q) = p\ln\left(\frac{p}{q}\right) + (1-p)\ln\left(\frac{1-p}{1-q}\right).$$

On va maintenant chercher à estimer les queues d'une distribution binomiale, c'est-à-dire à approximer les probabilités $\Pr[\Sigma_{\boldsymbol{k}^*} \leq T]$ et $\Pr[\Sigma_{\boldsymbol{k}^*} \geq T]$ avec $T \ll \mathbb{E}(\Sigma_{\boldsymbol{k}^*})$ pour la première et $T \gg \mathbb{E}(\Sigma_{\boldsymbol{k}^*})$pour la seconde.

La stratégie consiste à commencer par calculer le terme principal de la somme puis d'ajouter les autres termes par importance décroissante jusqu'à atteindre un certain degré de précision.

Soit Σ une variable aléatoire suivant une loi binomiale de paramètres N et p. Rappelons d'abord sa distribution :

$$\Pr[\Sigma = i] = \binom{N}{i} p^i (1-p)^{N-i}.$$

On a donc deux relations simples entre deux termes consécutifs :

$$\begin{aligned} \Pr[\Sigma = i+1] &= \binom{N}{i+1} p^{i+1}(1-p)^{N-i-1} \\ &= \frac{p(N-i)}{(1-p)(i+1)} \binom{N}{i} p^i (1-p)^{N-i} \\ &= \frac{p(N-i)}{(1-p)(i+1)} \Pr[\Sigma = i], \end{aligned} \tag{6.1}$$

et, en inversant la formule précédente,

$$\Pr[\Sigma = i-1] = \frac{(1-p)\,i}{p(N-i+1)} \Pr[\Sigma = i]. \tag{6.2}$$

On va donc définir deux fonctions γ^+ et γ^- qui donnent le rapport entre deux termes consécutifs

$$\gamma^+(i) \stackrel{\text{def}}{=} \frac{p(N-i)}{(1-p)(i+1)} \quad \text{et} \quad \gamma^-(i) \stackrel{\text{def}}{=} \frac{(1-p)\,i}{p(N-i+1)}.$$

L'algorithme de calcul numérique de la fonction de répartition est décrit par l'algorithme 11. Si $T > N\,p$ alors, pour pouvoir appliquer notre stratégie de descente, on calcule $\Pr[\Sigma > T]$ de façon à se placer sur la pente droite de la distribution.

L'algorithme est simple : on se place dans le cadre d'un calcul de queue de distribution et alors on effectue une descente jusqu'à atteindre une certaine précision. Pour cela on calcule le terme principal de la somme *reg* puis on le met à jour au fur et à mesure de la descente en utilisant (6.1) ou (6.2). L'algorithme est écrit de façon à ce qu'à chaque fin de boucle, $reg = \Pr[\Sigma = T]$, *coef* soit égal à $\gamma^{\pm}(T)$ et *res* soit la somme des termes jusqu'à T compris.

Il y a, toutefois, deux points à clarifier. Le premier est le calcul du terme principal $\Pr[\Sigma = T]$ car pour de grandes valeurs de N et T il devient difficile de calculer la valeur exacte de cette probabilité (ceci étant dû au coefficient binomial $\binom{N}{T}$). Le second point à clarifier à propos de cet algorithme est le calcul des bornes utilisées comme conditions d'arrêt des boucles.

Estimation du terme principal. La difficulté dans le calcul du terme principal est le coefficient binomial $\binom{N}{T} = \frac{N!}{T!(N-T)!}$. Le moyen classique d'estimer une factorielle est l'utilisation de l'approximation de Stirling (lemme 6.5).

Lemme 6.5. *Approximation de Stirling. Soit n un entier strictement positif, alors*

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\lambda_n},$$

Algorithme 11 : Calcul numérique de la fonction de répartition d'une distribution binomiale avec précision arbitraire.

Entrée : N et p paramètres de la distribution de Σ, le seuil T et ϵ l'erreur acceptée;
Sortie : y tel que $|\Pr[\Sigma \leq T] - y| \leq \epsilon$;
$reg \longleftarrow \Pr[\Sigma = T]$;
si $T \leq N\,p$ **alors**
  $res \longleftarrow reg$;
  $coef \longleftarrow \frac{(1-p)T}{p(N-T+1)}$;
  répéter
    $T \longleftarrow T - 1$;
    $reg \longleftarrow reg \cdot coef$;
    $res \longleftarrow res + reg$;
    $coef \longleftarrow \frac{(1-p)T}{p(N-T+1)}$;
  jusqu'à $T = 0$ *ou* $\frac{1-coef^{T-1}}{1-coef} \cdot coef \cdot reg \leq \epsilon$;
sinon
  $res \longleftarrow 0$;
  $coef \longleftarrow \frac{p(N-T)}{(1-p)T}$;
  répéter
    $T \longleftarrow T + 1$;
    $reg \longleftarrow coef \cdot reg$;
    $res \longleftarrow res + reg$;
    $coef \longleftarrow \frac{p(N-T)}{(1-p)(T+1)}$;
  jusqu'à $T = N$ *ou* $\frac{1-coef^{N-T}}{1-coef} \cdot coef \cdot reg \leq \epsilon$;
  $res \longleftarrow 1 - res$;
fin
retourner *res;*

avec $\frac{1}{12n+1} \leq \lambda_n \leq \frac{1}{12n}$.

On peut donc utiliser ce résultat pour estimer le coefficient binomial quand T et N sont trop grands. On obtient alors l'estimation donnée par le lemme 6.6. Attention, alors, à prendre en compte cette erreur d'estimation dans le calcul de l'erreur ϵ fournie à l'algorithme : pour une erreur globale de ϵ il faudra donner en paramètres $\frac{\epsilon}{1+\mathcal{O}(T^{-1})}$. Pour des raisons de lisibilité des formules, les résultats sont énoncés pour un seuil relatif τ correspondant à T/N.

Lemme 6.6. *Soit τ un seuil relatif, $0 \leq \tau \leq 1$. Soit Σ une variable aléatoire distribuée selon une loi binomiale de paramètres (N, p). Alors,*

$$\Pr[\Sigma = \lfloor \tau N \rfloor] = \sqrt{\frac{1}{2\pi N(1-\tau)\tau}}\, e^{-ND(\tau||p)} \left[1 + \mathcal{O}\left(\frac{1}{N(1-\tau)\tau}\right)\right]. \tag{6.3}$$

Plus précisément,

$$\Pr[\Sigma = \lfloor \tau N \rfloor] = \sqrt{\frac{1}{2\pi N(1-\tau)\tau}}\, e^{-ND(\tau||p)} (1 + E), \tag{6.4}$$

avec le terme d'erreur E tel que $|E| \leq \frac{1}{12N(1-\tau)\tau}$.

Démonstration : Commençons par utiliser le lemme 6.5 pour estimer $\binom{N}{T} = \frac{N!}{(N-T)!T!}$. On a,

$$\begin{aligned}\binom{N}{T} &= \frac{\sqrt{2\pi N}\left(\frac{N}{e}\right)^N e^{\lambda_N}}{\sqrt{2\pi T}\left(\frac{T}{e}\right)^T e^{\lambda_T}\sqrt{2\pi(N-T)}\left(\frac{N-T}{e}\right)^{N-T} e^{\lambda_{N-T}}} \\ &= \sqrt{\frac{N}{2\pi(N-T)T}}\cdot\frac{N^N}{T^T(N-T)^{N-T}}\cdot e^{\lambda_N-\lambda_T-\lambda_{N-T}},\end{aligned}$$

avec, pour rappel, $\frac{1}{12i+1} \leq \lambda_i \leq \frac{1}{12i}$. Avec $\tau = T/N$, on peut réécrire le second facteur

$$\begin{aligned}\frac{N^N}{(N-T)^{N-T}T^T} = \frac{N^{N-T}N^T}{(N-T)^{N-T}T^T} &= e^{-T\ln(\tau)-(N-T)\ln(1-\tau)} \\ &= e^{-N[\tau\ln(\tau)+(1-\tau)\ln(1-\tau)]}.\end{aligned}$$

On injecte ce résultat dans l'expression de $\Pr[\Sigma = \lfloor \tau N \rfloor]$.

$$\begin{aligned}\Pr[\Sigma = \lfloor \tau N \rfloor] &= \frac{e^{\lambda_N-\lambda_{N-T}-\lambda_T}}{\sqrt{2\pi N(1-\tau)\tau}}\cdot p^T(1-p)^{N-T}\cdot e^{-N[\tau\ln(\tau)+(1-\tau)\ln(1-\tau)]}. \\ &= \frac{e^{\lambda_N-\lambda_{N-T}-\lambda_T}}{\sqrt{2\pi N(1-\tau)\tau}}\cdot e^{-N\left[\tau\ln\left(\frac{\tau}{p}\right)+(1-\tau)\ln\left(\frac{1-\tau}{1-p}\right)\right]}.\end{aligned}$$

On s'occupe finalement du terme $e^{\lambda_N-\lambda_{N-T}-\lambda_T}$ qui peut être encadré par

$$e^{-\lambda_{N-T}-\lambda_T} \leq e^{\lambda_N-\lambda_{N-T}-\lambda_T} \leq e^{\lambda_N}.$$

La valeur de $\lambda_{N-T} + \lambda_T$ peut être majorée par $\frac{1}{12(N-T)} + \frac{1}{12T}$ et donc par $\frac{12N}{12^2(N-T)T} = \frac{1}{12N(1-\tau)\tau}$. En utilisant le développement en série de la fonction exponentielle on obtient l'encadrement

$$1 - \frac{1}{12N(1-\tau)\tau} \leq e^{\lambda_N-\lambda_{N-T}-\lambda_T} \leq 1 + \frac{1}{12N} + \sum_{i\geq 2}\frac{1}{i!(12N)^i}.$$

En minorant $i!$ par 1, on peut borner le terme de droite par $1 + \frac{1}{12N}\left(1 + \frac{1}{12N-1}\right)$. De plus, comme $(1-\tau)\tau \leq 0.25$, alors $\frac{1}{(1-\tau)\tau} \geq 4$ et donc le terme d'erreur $\frac{1}{12N} + \sum_{i\geq 2}\frac{1}{i!(12N)^i}$ est clairement supérieur à l'autre. On obtient donc $|e^{\lambda_N-\lambda_{N-T}-\lambda_T} - 1| \leq \frac{1}{12N(1-\tau)\tau}$ et donc

$$\Pr[\Sigma = \lfloor \tau N \rfloor] = \sqrt{\frac{1}{2\pi N(1-\tau)\tau}}\, e^{-ND(\tau||p)}\,[1+E],$$

avec $|E| \leq \frac{1}{12N(1-\tau)\tau}$. □

On a donc une bonne estimation de cette probabilité pour peu que T soit suffisamment grand. Et encore, pour $T = 1$, l'erreur relative est seulement de 1/12. Il reste cependant beaucoup plus facile et précis d'utiliser l'expression exacte de la probabilité pour les petites valeurs de T.

Conditions d'arrêt. Pour avoir une condition d'arrêt, il faut pouvoir majorer la somme des termes restant à un instant précis du déroulement de l'algorithme. On va pour cela majorer la suite des termes restant à calculer par une suite géométrique et en déduire une borne supérieure sur l'erreur commise en s'arrêtant tout de suite. Le lemme 6.7 donne le résultat de cette majoration.

Lemme 6.7. *Pour* $0 \leq A < Np < B \leq N$, *on définit* $\gamma^-(B) \stackrel{\text{def}}{=} \frac{(1-p)B}{p(N-B+1)}$ *et* $\gamma^+(A) \stackrel{\text{def}}{=} \frac{p(N-A)}{(1-p)(A+1)}$. *Alors,*

$$\sum_{i=0}^{B} \Pr[\Sigma = i] \leq \Pr[\Sigma = B] \left[1 + \gamma^-(B) \frac{1-\gamma^-(B)^B}{1-\gamma^-(B)}\right].$$

Et,

$$\sum_{i=A}^{N} \Pr[\Sigma = i] \leq \Pr[\Sigma = A] \left[1 + \gamma^+(A) \frac{1-\gamma^+(A)^{N-A}}{1-\gamma^+(A)}\right].$$

Démonstration : Pour la première borne,

$$\begin{aligned}
\sum_{i=0}^{B} \Pr[\Sigma = i] &= \Pr[\Sigma = B] \cdot \left[1 + \frac{(1-p)B}{p(N-B+1)} + \cdots + \frac{(1-p)^B\, B!}{p^{B-A}(N-B+1)\ldots N}\right] \\
&\leq \Pr[\Sigma = B] \cdot \left[1 + \sum_{i=1}^{B} \gamma^-(B)^i\right] \\
&\leq \Pr[\Sigma = B] \left[1 + \gamma^-(B) \frac{1-\gamma^-(B)^B}{1-\gamma^-(B)}\right].
\end{aligned}$$

Pour la seconde c'est aussi simple

$$\begin{aligned}
\sum_{i=A}^{N} \Pr[\Sigma = i] &= \Pr[\Sigma = A] \cdot \left[1 + \frac{p(N-A)}{(1-p)A} + \cdots + \frac{p^{N-A}(N-A)!}{(1-p)^{N-A}A\ldots(N-1)}\right] \\
&\leq \Pr[\Sigma = A] \cdot \left[1 + \sum_{i=1}^{N-A} \gamma^+(A)^i\right] \\
&\leq \Pr[\Sigma = A] \left[1 + \gamma^+(A) \frac{1-\gamma^+(A)^{N-A}}{1-\gamma^+(A)}\right].
\end{aligned}$$

□

Dans le cas où l'on est sur la pente gauche de la distribution on veut s'arrêter dès que t vérifie

$$\sum_{i=t}^{T} \Pr[\Sigma = i] \leq \Pr[\Sigma \leq T] \leq \sum_{i=t}^{T} \Pr[\Sigma = i] + \epsilon.$$

Il faut donc que $\sum_{i=0}^{t-1} \Pr[\Sigma = i] \leq \epsilon$. En utilisant le lemme 6.7, on borne cette erreur par

$$\sum_{i=0}^{t-1} \Pr[\Sigma = i] \leq \Pr[\Sigma = t] \left[\gamma^-(t) \frac{1-\gamma^-(t)^t}{1-\gamma^-(t)}\right].$$

Cette borne peut être traduite en terme de variables reg, $coef$ et T de l'algorithme 11 :

$$\sum_{i=0}^{t-1} \Pr[\Sigma = i] \leq reg \cdot \left[coef \cdot \frac{1 - coef^{T+1}}{1 - coef} \right].$$

Pour des questions de rapidité d'exécution, il peut être intéressant d'ignorer le terme $coef^{T+1}$ car il est souvent très petit et donc n'apporte pas beaucoup d'information alors qu'il est long à calculer (un produit, un logarithme et une exponentiation).

Pour la condition d'arrêt sur le second versant de la distribution, le raisonnement est le même. La différence ici est que l'erreur est dans l'autre sens : on veut s'arrêter dès que t vérifie

$$1 - \sum_{i=T+1}^{t} \Pr[\Sigma = i] - \epsilon \leq \Pr[\Sigma \leq T] \leq 1 - \sum_{i=T+1}^{t} \Pr[\Sigma = i].$$

En appliquant de nouveau le lemme 6.7 et on obtient

$$\sum_{i=t+1}^{N} \Pr[\Sigma = i] \leq \Pr[\Sigma = t] \cdot \gamma^+(t) \cdot \frac{1 - \gamma^+(t)^{N-t}}{1 - \gamma^+(t)}.$$

Et avec les notations de l'algorithme,

$$\sum_{i=t+1}^{N} \Pr[\Sigma = i] \leq reg \cdot coef \cdot \frac{1 - coef^{N-t}}{1 - coef}.$$

6.3.3 Approximation de la fonction de répartition de la binomiale

Maintenant que nous avons vu l'algorithme permettant de calculer précisément la fonction de répartition d'une binomiale, nous allons nous attaquer à l'estimation d'une queue de binomiale c'est-à-dire à trouver une formule qui donne une bonne approximation des probabilités $\Pr[\Sigma \leq T]$ et $\Pr[\Sigma \geq T]$ pour $T \ll \mathbb{E}(\Sigma)$ dans le premier cas et $T \gg \mathbb{E}(\Sigma)$ dans le second. On va pouvoir s'inspirer de la stratégie utilisée dans la sous-section 6.3.2 et obtenir le résultat suivant qui peut être trouvé, par exemple, dans un papier de biologie [AG89]. L'essentiel de cette sous-section est donc dévolu à la preuve du théorème 6.8 qui présente ces formules.

Théorème 6.8. *Soit Σ une variable aléatoire distribuée selon une loi binomiale de paramètres N et p, alors, pour $\tau < p$,*

$$\Pr[\Sigma \leq \tau N] \underset{N\to\infty}{\sim} \frac{p\sqrt{1-\tau}}{(p-\tau)\sqrt{2\pi N\tau}} e^{-ND(\tau||p)}, \tag{6.5}$$

et, pour $\tau > p$, on a

$$\Pr[\Sigma \geq \tau N] \underset{N\to\infty}{\sim} \frac{(1-p)\sqrt{\tau}}{(\tau-p)\sqrt{2\pi N(1-\tau)}} e^{-ND(\tau||p)}. \tag{6.6}$$

La notation $f(N) \underset{N\to\infty}{\sim} g(N)$ signifie $\lim_{N\to\infty} \frac{f(N)}{g(N)} = 1$.

Il est important de noter que ce résultat est asymptotique. Ces formules seront donc mauvaises pour les cas particuliers où le seuil T sera très proche de 0 ou N. Dans ces cas, le calcul exact de la probabilité se fait facilement car les coefficients binomiaux sont encore de taille raisonnable. D'ailleurs, les formules du théorème 6.8 deviennent précises bien avant que le fait de calculer la probabilité exacte soit difficile.

La stratégie ici est de trouver une estimation de la somme des termes importants. On pourra ainsi utiliser le lemme 6.7 pour majorer le reste. On va donc avoir besoin du lemme suivant.

Lemme 6.9. *On rappelle que pour* $0 \leq i \leq N$, $\gamma^-(i) \stackrel{\text{def}}{=} \frac{(1-p)i}{p(N-i+1)}$ *et* $\gamma^+(i) \stackrel{\text{def}}{=} \frac{p(N-i)}{(1-p)(i+1)}$. *Alors, pour* $0 \leq A < B < N\,p$,

$$\Pr[\Sigma = B]\,\frac{1-\gamma^-(A+1)^{B-A+1}}{1-\gamma^-(A+1)} \leq \sum_{i=A}^{B} \Pr[\Sigma = i] \leq \Pr[\Sigma = B]\,\frac{1-\gamma^-(B)^{B-A+1}}{1-\gamma^-(B)}. \tag{6.7}$$

De même, pour $N\,p < A < B \leq N$,

$$\Pr[\Sigma = A]\,\frac{1-\gamma^+(B-1)^{B-A+1}}{1-\gamma^+(B-1)} \leq \sum_{i=A}^{B} \Pr[\Sigma = i] \leq \Pr[\Sigma = A]\,\frac{1-\gamma^+(A)^{B-A+1}}{1-\gamma^+(A)}. \tag{6.8}$$

Démonstration : La preuve est simple, on reprend la preuve du lemme 6.7 avec A en lieu et place de 0 et l'on obtient

$$\sum_{i=A}^{B} \Pr[\Sigma = i] = \Pr[\Sigma = B] \cdot \left[1 + \frac{(1-p)B}{p(N-B+1)} + \cdots + \frac{(1-p)^B\,B\ldots(A+1)}{p^{B-A}(N-B+1)\ldots(N-A)}\right].$$

On borne alors le terme entre crochets par deux séries géométriques en prenant les coefficients minimum et maximum i.e. $\gamma^-(A+1)$ et $\gamma^-(B)$. La preuve est la même pour la seconde partie du lemme. □

On va maintenant s'intéresser à prouver le théorème 6.8.

Démonstration : On va maintenant appliquer la stratégie évoquée antérieurement pour prouver le théorème 6.8. On n'effectuera la preuve que pour le cas $\Pr[\Sigma \leq \tau N]$, le second cas étant similaire. Soit $\tau < p$ et $B \stackrel{\text{def}}{=} \lfloor N\,\tau \rfloor$, on cherche une estimation de $\sum_{i=0}^{B} \Pr[\Sigma = i]$. On va donc découper cette somme en deux comme prévu. Soit $0 < a < \tau$ et $A \stackrel{\text{def}}{=} \lfloor N\,a \rfloor$, on écrit

$$\Pr[\Sigma \leq \tau N] = \sum_{i=A}^{B} \Pr[\Sigma = i] + \sum_{i=0}^{A-1} \Pr[\Sigma = i]\,. \tag{6.9}$$

En utilisant les notations du lemme 6.9,

$$\begin{aligned}
\gamma^-(B) &= \frac{(1-p)B}{p(N-B+1)} \\
&= \frac{1-p}{p} \cdot \frac{B}{N-B} \cdot \frac{1}{1+\frac{1}{N-B}} \\
&= \frac{(1-p)\tau}{p(1-\tau)} \cdot \left[1 + \mathcal{O}\left(\frac{\epsilon - \tau}{N\,\tau(1-\tau)}\right)\right].
\end{aligned}$$

La dernière égalité est obtenue en remplaçant B par $N\tau - \epsilon$. En utilisant ce résultat dans le membre de droite de l'inégalité (6.7), on obtient

$$\begin{aligned}\frac{1-\gamma^-(B)^{B-A+1}}{1-\gamma^-(B)} &= \frac{1}{1-\gamma^-(B)} \cdot \left[1+\mathcal{O}\left(\gamma^-(B)^{B-A+1}\right)\right] \\ &\underset{N\to\infty}{\sim} \frac{1}{1-\frac{(1-p)\tau}{p(1-\tau)}} \\ &\underset{N\to\infty}{\sim} \frac{p(1-\tau)}{p-\tau}.\end{aligned}$$

On choisit A suffisamment proche de τ pour que $\frac{1-\gamma^-(A+1)^{B-A+1}}{1-\gamma^-(A+1)} \approx \frac{p(1-\tau)}{p-\tau}$.
On obtient alors, pour la première somme de (6.9), en utilisant le lemme 6.6,

$$\sum_{i=A}^{B} \Pr\left[\Sigma = i\right] \underset{N\to\infty}{\sim} \frac{p}{p-\tau}\sqrt{\frac{1-\tau}{2\pi N\tau}}\, e^{-ND(\tau||p)}. \tag{6.10}$$

Quant à la seconde somme, le lemme 6.7 nous dit que

$$\begin{aligned}\sum_{i=0}^{A-1} \Pr\left[\Sigma = i\right] &\leq \Pr\left[\Sigma = A-1\right] \frac{1-\gamma^-(A-1)^A}{1-\gamma^-(A-1)} \\ &\leq \frac{\Pr\left[\Sigma = A-1\right]}{1-\gamma^-(A-1)} \\ &\leq \frac{\Pr\left[\Sigma = B\right]\gamma^-(B)^{B-A+1}}{1-\gamma^-(B)}.\end{aligned}$$

Cette somme est donc majorée par $\gamma^-(B)^{B-A+1}$ fois la première somme et donc, quel que soit $A \leq \tau$, on a bien $\sum_{i=0}^{A-1} \Pr\left[\Sigma = i\right] \underset{N\to\infty}{\sim} 0$. □

6.3.4 Comparaison des différentes approximations

On va rapidement reprendre les exemples de cryptanalyse tronquée de la section 6.3 et comparer, cette fois-ci, les résultats obtenus pour des probabilités de la forme $\Pr\left[\Sigma \leq \tau N\right]$ ou $\Pr\left[\Sigma \geq \tau N\right]$ avec les formules du théorème 6.8 et les approximations normale et de Poisson.

On voit bien, sur la figure 6.5, que l'approximation donnée par le théorème 6.8 se comporte bien pour les deux types de paramètres même si pour un jeu donné, elle est moins précise qu'une des deux approximations classiques. Cependant, on voit dans la figure 6.6 que dans le cas où ni la loi de Poisson ni la distribution gaussienne ne conviennent, que la formule présentée ici est toujours aussi précise.

6.4 Complexité en données

On va maintenant s'attaquer au problème de l'estimation de la quantité de données nécessaires à la réussite d'une cryptanalyse statistique à seuil fixé (voir sous-section 6.1.1). On verra, dans la section 6.5 ce que l'on peut faire pour l'autre approche du problème.

Le problème ici est donc de trouver la meilleure règle de décision concernant le choix de garder un candidat ou non et de calculer la quantité de données correspondante. Cette problématique est abordée dans la section A.2. Voyons maintenant de quoi il retourne dans le cas des cryptanalyses statistiques simples.

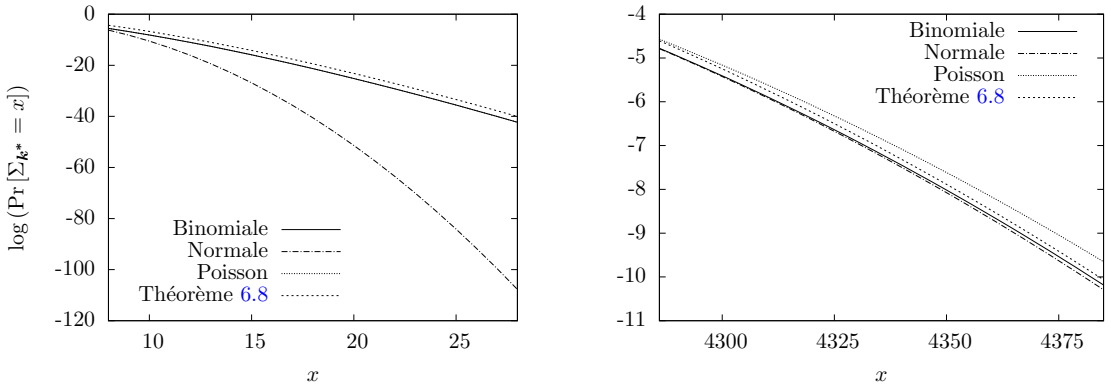

FIGURE 6.5 – Comparaison des approximations normale et de Poisson pour des paramètres de cryptanalyse différentielle tronquée.

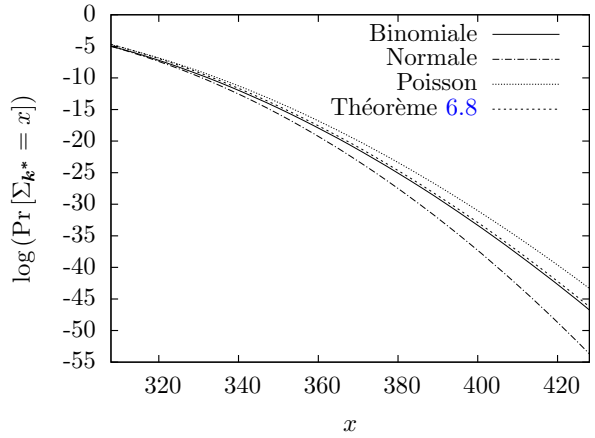

FIGURE 6.6 – Exemple de paramètres pour lesquels les deux approximations standards ne sont pas pertinentes.

6.4.1 Test binaire d'hypothèses

Les réalisations mises à notre disposition correspondent aux variables $X^i_{\boldsymbol{k}}$ définies dans le modèle 6.1. On sait que ces variables sont distribuées selon la distribution de Bernoulli ayant pour paramètre p_0 ou p selon la valeur de $\boldsymbol{k}$. On a donc deux hypothèses :

- H_0 : *le paramètre de la distribution est* p_0 ;
- H_1 : *le paramètre de la distribution est* p.

Ici, l'erreur de type 1 correspond au fait de refuser la bonne clef et celle de type 2 à garder un mauvais candidat. On va donc définir les deux probabilités d'erreur correspondantes en leur donnant des noms plus expressifs.

Définition 6.10. On se place dans le cadre d'un test entre deux hypothèses H_0 et H_1. Pour un ensemble de N réalisations donné, on valide une des deux hypothèses H_i. On note A_N la région d'acceptation qui détermine l'ensemble des réalisations pour lequel on va choisir l'hypothèse H_0. On peut alors définir les deux probabilités d'erreur suivantes.

– *probabilité de non-détection :*

$$\alpha(A_N) \stackrel{\text{def}}{=} \Pr\left[(X_1, \ldots, X_n) \notin A_N | H_0 \text{ vraie}\right];$$

– *probabilité de fausse alarme :*

$$\beta(A_N) \stackrel{\text{def}}{=} \Pr\left[(X_1, \ldots, X_n) \in A_N | H_1 \text{ vraie}\right].$$

La probabilité de succès d'une cryptanalyse est alors la probabilité que la bonne clef soit acceptée $P_S = 1 - \alpha(A_N)$. Dans cette approche à seuil fixé, l'objectif est donc de minimiser β vu que c'est cette probabilité qui détermine principalement la complexité en temps de l'attaque.

Regardons maintenant quelle est la forme d'une région d'acceptation optimale pour notre cas c'est-à-dire avec des variables qui suivent une loi de Bernoulli. On s'appuie sur le lemme de Neyman-Pearson (rappelé au théorème A.16)

Lemme 6.11. *La région d'acceptation optimale (au sens du lemme de Neyman-Pearson) pour un problème de test binaire d'hypothèse entre deux distributions de Bernoulli de probabilités p_0 et p est de la forme*

$$A_N(T) \stackrel{\text{def}}{=} \left\{ (x_1, \ldots, x_N), \sum_{i=1}^{N} x_i \geq T \right\}.$$

Démonstration : Les variables x_i étant indépendantes, on peut réécrire le rapport de vraisemblance de l'équation (A.2) comme :

$$\frac{\Pr_{\mathcal{P}_0}[x_1, \ldots, x_N]}{\Pr_{\mathcal{P}_1}[x1, \ldots, x_N]} = \frac{\prod_{i=1}^{N} \Pr_{\mathcal{P}_0}[x_i]}{\prod_{i=1}^{N} \Pr_{\mathcal{P}_1}[x_i]}.$$

Puis, on utilise le fait que ces variables soient des variables de Bernoulli

$$\frac{\prod_{i=1}^{N} \Pr_{\mathcal{P}_0}[x_i]}{\prod_{i=1}^{N} \Pr_{\mathcal{P}_1}[x_i]} = \frac{\prod_{i=1}^{N} \binom{N}{x_i} p_0^{x_i}(1-p_0)^{N-x_i}}{\prod_{i=1}^{N} \binom{N}{x_i} p^{x_i}(1-p)^{N-x_i}}.$$

Après simplification, on obtient

$$\left(\frac{1-p_0}{1-p}\right)^N \left(\frac{p_0(1-p)}{p(1-p_0)}\right)^{\sum x_i}.$$

Et donc, comme la fonction logarithme est croissante et le second quotient positif, on a bien

$$\frac{\Pr_{\mathcal{P}_0}[x_1, \ldots, x_N]}{\Pr_{\mathcal{P}_1}[x_1, \ldots, x_N]} \geq t \Leftrightarrow \sum_{i=1}^{N} x_i \geq T,$$

pour $T = \left[\log(t) - N \log\left((1-p_0)/(1-p)\right)\right] \cdot \log\left(\frac{p_0(1-p)}{p(1-p_0)}\right)^{-1}$. □

La règle de décision utilisant un seuil T sur les compteurs $\Sigma_{\boldsymbol{k}}$ est donc optimale. Il reste donc à trouver le nombre minimum N d'échantillons nécessaires pour atteindre des probabilités d'erreurs α et β fixées.

6.4.2 Algorithme de calcul numérique du couple (N, T) optimal

Dans cette sous-section nous allons présenter un algorithme permettant de calculer la paire (N, T) optimale pour une cryptanalyse statistique. Plus précisément, on souhaite trouver le N minimal tel qu'il existe un seuil T vérifiant le système suivant pour deux bornes α et β fixées

$$\begin{cases} \alpha\left(A_N(T)\right) \leq \alpha, \\ \beta\left(A_N(T)\right) \leq \beta. \end{cases} \tag{6.11}$$

Pour un nombre N d'échantillons donnés, le seuil T va permettre d'effectuer un compromis entre les deux probabilités. Plus T sera pris grand, moins on acceptera de candidats et donc plus la probabilité de fausse alarme diminuera. En contrepartie, la probabilité de non-détection, elle, augmentera. On verra évidemment le phénomène contraire se produire si l'on diminue T. Pour chaque T il existe donc un N tel que le couple (N, T) vérifie (6.11). Le but est de trouver le couple ayant le N minimum.

La pratique montre qu'utiliser des nombres entiers peut, dans certains cas, mener à des effets de seuil qui rendent difficile cette recherche (qui n'est rien d'autre qu'un problème simple d'optimisation). C'est pourquoi l'on va manipuler des valeurs réelles de façon à « lisser » l'espace de recherche ce qui facilitera la recherche du minimum. Le fait d'utiliser des nombres réels va nous permettre de manipuler le seuil relatif $\tau = T/N$ en lieu et place de T. Cela semble peut être anecdotique mais c'est en fait très important car l'espace de recherche sur T était dépendant de N. Ce ne sera plus le cas avec $0 \leq \tau \leq 1$. On peut d'ailleurs, dès à présent, préciser que pour avoir une probabilité de succès raisonnable et une complexité en temps meilleure que la recherche exhaustive, on prendra $p \leq \tau \leq p_0$.

Remarque. On fera, effectivement, dans le reste du chapitre, l'hypothèse que $p < p_0$. Ceci se fait sans perte de généralité.

Revenons à notre problème de recherche sur cet espace discret que l'on souhaite rendre continu. On va donc utiliser deux fonctions G_{nd} et G_{fa} à entrées réelles qui, d'une certaine manière, interpoleront les probabilités d'erreur. On prendra ces fonctions continues pour obtenir l'effet lissé souhaité. Le système (6.11) devient donc

$$\begin{cases} G_{\text{nd}}(N, \tau) \leq \alpha, \\ G_{\text{fa}}(N, \tau) \leq \beta. \end{cases} \tag{6.12}$$

L'algorithme consiste alors à faire une recherche dichotomique sur la valeur du seuil relatif τ de la façon suivante. Pour un τ fixé, on peut calculer les N minimaux permettant d'atteindre chacune des deux bornes sur les probabilités d'erreur

$$N_{\text{nd}}(\tau) \stackrel{\text{def}}{=} \underset{N}{\operatorname{argmin}}\, G_{\text{nd}}(N, \tau) \leq \alpha \quad , \quad N_{\text{fa}}(\tau) \stackrel{\text{def}}{=} \underset{N}{\operatorname{argmin}}\, G_{\text{fa}}(N, \tau) \leq \beta.$$

Si les deux valeurs sont égales alors le couple (N, τ) est celui recherché. En effet, pour $N' < N$, on aura les deux probabilités d'erreur supérieures aux bornes α et β. Un changement de seuil relatif τ ne pouvant diminuer qu'une de ces deux valeurs, on ne pourra jamais atteindre les bornes souhaitées. Dans le cas où une des deux valeurs ($N_{\text{nd}}(\tau)$ ou $N_{\text{fa}}(\tau)$) est supérieure à l'autre, on peut modifier le seuil τ afin de les rééquilibrer. Pour que cet algorithme fonctionne, il faut que nos fonctions G_{nd} et G_{fa} vérifient quelques conditions qui fassent d'elles des candidats raisonnables en tant qu'« interpolation » des probabilités d'erreur. Le lemme 6.12 liste les conditions nécessaires à l'exécution et à l'arrêt de l'algorithme.

Lemme 6.12. *Soient $G_{\rm nd}(N,\tau)$ et $G_{\rm fa}(N,\tau)$ deux fonctions continues étant définies sur $]0;+\infty[\times]p;p_0[$ et ayant les propriétés suivantes :*

- *Pour τ fixé, ce sont des fonctions décroissantes de N.*
- *Pour N fixé, $G_{\rm nd}(N,\tau)$ (resp. $G_{\rm fa}(N,\tau)$) est une fonction croissante (resp. décroissante) de τ.*
- *Pour τ fixé, $\lim\limits_{N\to\infty} G_{\rm nd}(N,\tau) \leq \alpha$ et $\lim\limits_{N\to\infty} G_{\rm fa}(N,\tau) \leq \beta$.*
- *Pour τ fixé, $\lim\limits_{N\to 0} G_{\rm nd}(N,\tau) \geq \alpha$ et $\lim\limits_{N\to 0} G_{\rm fa}(N,\tau) \geq \beta$.*

On rappelle que pour deux bornes α et β dans $[0;1]$ et un seuil relatif τ,

$$G_{\rm nd}(N_{\rm nd}(\tau),\tau) = \alpha \quad , \quad G_{\rm fa}(N_{\rm fa}(\tau),\tau) = \beta.$$

On introduit maintenant $N(\tau) \stackrel{\rm def}{=} \max\left(N_{\rm nd}(\tau), N_{\rm fa}(\tau)\right)$ qui est la valeur minimum de N telle que (N,τ) vérifie (6.12).

Alors, pour $p \leq \tau \leq p_0$,

- *si $N_{\rm nd}(\tau) > N_{\rm fa}(\tau)$, alors pour tout $\tau' > \tau$, $N(\tau') > N(\tau)$;*
- *si $N_{\rm nd}(\tau) < N_{\rm fa}(\tau)$, alors pour tout $\tau' < \tau$, $N(\tau') > N(\tau)$.*

Avant de donner la preuve de ce lemme, j'aimerais faire quelques remarques sur les conditions énoncées.

- Les deux dernières conditions ne sont là que pour assurer l'existence de $N_{\rm nd}(\tau)$ et $N_{\rm fa}(\tau)$ à chaque itération.
- On a donné les conditions permettant d'exécuter l'algorithme. Maintenant, si l'on veut que le résultat soit satisfaisant, il est bien entendu recommandé que pour des valeurs de N et τ telles que N et $\tau\,N$ soient des entiers, $G_{\rm nd}(N,\tau) \approx \alpha\left(A_N(\tau\,N)\right)$ et $G_{\rm fa}(N,\tau) \approx \beta\left(A_N(\tau\,N)\right)$.
- Les vraies probabilités d'erreur ne satisfont pas forcément la première condition pour des valeurs entières de τ et N, c'est ce que j'ai appelé effet de seuil et ce qui explique l'utilisation d'estimations continues de ces probabilités.

Démonstration : □

Résumons donc cette méthode de recherche sous la forme de l'algorithme 12.

Affiner les résultats. Comme l'on manipule des nombres réels, il va falloir traduire le résultat en nombres entiers. La façon la plus triviale de procéder est de regarder les quatre couples formés par les troncatures de N et $N\tau$ $((\lfloor N\rfloor, \lfloor N\,\tau\rfloor)$, $(\lfloor N\rfloor, \lceil N\,\tau\rceil)$,...), calculer les probabilités d'erreur correspondantes et choisir le couple minimal parmi ceux menant à des probabilités satisfaisantes. De nouveau, on peut se trouver gêné par les effets de seuils. Dans ce cas, passer de $\lfloor N\,\tau\rfloor$ à $\lceil N\,\tau\rceil$ fait faire un « bond » au niveau des probabilités. Il faudra alors effectuer une recherche dichotomique sur N pour chacun de ces deux seuils et regarder lequel des couples est optimal.

Choix des $G_{\rm nd}$ et $G_{\rm fa}$. La pertinence du résultat de l'algorithme dépend essentiellement du bon choix des estimations continues des probabilités. Notons que les estimations (6.5) et (6.6) peuvent être utilisées pour $G_{\rm nd}$ et $G_{\rm fa}$ car elles satisfont ces conditions. On peut cependant utiliser les estimations plus précises (mais plus longues à calculer).

$$G_{\rm nd}(N,\tau) = \Pr\left[\Sigma_{\boldsymbol{k}^*} < \lfloor N\tau\rfloor\right] + (N\tau - \lfloor N\tau\rfloor)\Pr\left[\Sigma_{\boldsymbol{k}^*} = \lfloor N\tau\rfloor\right] \tag{6.13}$$

$$G_{\rm fa}(N,\tau) = \Pr\left[\Sigma_{\boldsymbol{k}} \geq \lceil N\tau\rceil\right] + (\lceil N\tau\rceil - N\tau)\Pr\left[\Sigma_{\boldsymbol{k}^*} = \lfloor N\tau\rfloor\right] \tag{6.14}$$

Algorithme 12 : Calcul du nombre d'échantillons nécessaires à une attaque statistique.

Entrée : Deux bornes (α, β), les paramètres (p_0, p) de l'attaque et les estimations $G_{\text{nd}}(N, \tau)$ et $G_{\text{fa}}(N, \tau)$;
Sortie : Le couple (N, τ) minimal tel que les probabilités soient bornées par α et β;

$(\tau_{\min}, \tau_{\max}) \longleftarrow (p, p_0)$;
répéter
  $\tau \longleftarrow \frac{\tau_{\min}+\tau_{\max}}{2}$;
  $(N_1, N_2) \longleftarrow \big(N_{\text{nd}}(\tau), N_{\text{fa}}(\tau)\big)$;
  si $N_1 > N_2$ **alors**
    $\tau_{\max} \longleftarrow \tau$;
  sinon
    $\tau_{\min} \longleftarrow \tau$;
  fin
jusqu'à $N_1 = N_2$;
retourner $N = N_1 = N_2$ *et* τ;

Utilisation. Par la suite, nous allons donner des formules pour calculer une valeur approchée de N. On les comparera avec les valeurs obtenues en utilisant d'autres formules de la littérature ainsi qu'avec la « vraie » valeur de N. Cette « vraie » valeur de N sera calculée en utilisant l'algorithme 12 avec les estimations (6.13) et (6.14). On affinera, de plus, en utilisant l'algorithme 11 pour calculer précisément les probabilités d'erreur obtenues. C'est pourquoi ce qui sera appelé « vraie valeur de N » pourra, en effet, être considéré comme la vraie valeur de N.

6.4.3 Deux formules pour N

Maintenant que l'on a vu comment calculer précisément la valeur de N, on va essayer de trouver une expression simple qui soit une bonne estimation de N. Comme l'on a vu, la difficulté vient du fait que l'on a un système de deux équations à deux inconnues. De l'application de l'algorithme 12 on ne peut pas tirer une formule raisonnable, on va donc devoir contourner le problème en fixant un des deux paramètres. Comme l'on est dans le cadre de cryptanalyses statistiques, la borne β est la plus importante car elle est liée à la complexité en temps de l'attaque. En revanche, la seule chose que l'on demande à α est de ne pas être trop élevée. On va fixer le seuil relatif $\tau = p_0$. Ceci donne une probabilité de non-détection de l'ordre de 0.5 et donc une probabilité de succès pour l'attaque du même ordre (ce qui est raisonnable). On va alors chercher à exprimer N en fonction de la borne β et des probabilités p_0 et p.

Ce théorème diffère de celui que l'on trouve dans [BGT10] : l'approximation donnée par (6.15) est meilleure que dans l'article

Théorème 6.13. *Soit p_0 (resp. p) la probabilité d'apparition d'un phénomène lors d'une cryptanalyse statistique pour le candidat correspondant à la bonne clef (resp. un autre candidat). Alors, si on prend un seuil relatif $\tau = p_0$ pour définir la région d'acceptation, alors (6.15) et (6.16) sont de bonnes estimations du nombre d'échantillons N nécessaires pour distinguer ces deux distributions avec une probabilité de non-détection de l'ordre de* 0.5 *et une*

probabilité de fausse alarme inférieure ou égale à β.

$$N' \stackrel{\text{def}}{=} -\frac{1}{D\left(p_0 \middle| p\right)}\left[\ln\left(\frac{\lambda\beta}{\sqrt{D\left(p_0 \middle| p\right)}}\right) + \frac{1}{2}\ln\left(-\ln\left(\frac{\lambda\beta}{\sqrt{D\left(p_0 \middle| p\right)}}\right)\right)\right], \tag{6.15}$$

$$N'' \stackrel{\text{def}}{=} -\frac{1}{D\left(p_0 \middle| p\right)}\ln\left(\frac{\lambda\beta}{\sqrt{D\left(p_0 \middle| p\right)}}\right), \tag{6.16}$$

avec $\lambda \stackrel{\text{def}}{=} \frac{(p_0-p)\sqrt{2\pi(1-p_0)}}{(1-p)\sqrt{p_0}}$. *En effet, on a les encadrements*

$$N' \leq N_\infty \leq N'\left[1 + \frac{\ln(\theta)}{2\theta}\right],$$

et

$$N'' \leq N_\infty \leq N''\left[1 + \frac{\ln(N'/N'')}{2\theta}\right],$$

où $\theta = -\ln(\lambda\beta\sqrt{D\left(p_0 \middle| p\right)})$ *et où* N_∞ *est la valeur obtenue avec l'algorithme 12 en utilisant (6.5) et (6.6).*

Précision des estimations du théorème 6.13.

Avant de prouver ce théorème, regardons la précision des approximations N' et N''. La première chose à remarquer est qu'en utilisant le développement limité de $\ln(1-x)$, on obtient que $\frac{\ln(N'/N'')}{2\theta} \approx \frac{\ln(\theta)}{4\theta^2}$. De plus, on a $\theta > 1$ (ceci se voit clairement dans la preuve du théorème). Or, les fonctions $\frac{\ln(x)}{2x}$ et $\frac{\ln(x)}{4x^2}$ sont positives sur $[1;\infty[$ et respectivement majorées par $\frac{\ln(e)}{2e} \approx 0.18$ et $\frac{\ln(\sqrt{e})}{4e} \approx 0.046$. Ceci montre bien que les approximations sont très bonnes quelle que soit la valeur des paramètres. Cependant, celles-ci sont encore meilleures pour de petites valeurs de β. En effet, si on note $N_\infty(\beta), N'(\beta)$ et $N''(\beta)$ les valeurs de N_∞, N' et N'' pour un β fixé, alors, $\lim_{\beta\to 0}|N_\infty(\beta) - N'(\beta)| = \lim_{\beta\to 0}|N_\infty(\beta) - N''(\beta)| = 0$. Cela vient du fait que $\lim\limits_{\beta\to 0}\theta = \infty$ et $\lim\limits_{x\to\infty}\ln(x)x^{-a} = 0$ pour $a > 0$.

Démonstration :
On a $\tau = p_0$ et l'on fixe la probabilité de fausse alarme à β. On utilise alors l'équation (6.6) du théorème 6.8 qui nous donne l'approximation

$$\beta \approx \frac{(1-p)\sqrt{p_0}}{(p_0-p)\sqrt{2\pi N(1-p_0)}}e^{-ND(p_0||p)}. \tag{6.17}$$

La stratégie pour obtenir une approximation de N est d'utiliser la méthode du point fixe. On réécrit donc (6.17) en

$$N \approx -\frac{\ln(\lambda\beta\sqrt{N})}{D\left(p_0 \middle| p\right)}, \tag{6.18}$$

avec $\lambda \stackrel{\text{def}}{=} \frac{(p_0-p)\sqrt{2\pi(1-p_0)}}{(1-p)\sqrt{p_0}}$. On utilise alors la fonction f qui a la bonne propriété d'être contractante dans la zone qui nous intéresse.

$$f(x) \stackrel{\text{def}}{=} -\frac{\ln(\lambda\beta\sqrt{x})}{D\left(p_0 \middle| p\right)}.$$

En effet, la dérivée $f'(x)$ a une valeur absolue inférieure à 1 si $2D\left(p_0|\,|p\right)\cdot x$ est supérieur à 1. On va donc utiliser la technique du point fixe et appliquer f de façon itérative créant ainsi une suite de valeurs $(N_i)_{i\geq 0}$ avec $N_{i+1}=f(N_i)$. Si N_0 est pris tel que $2D\left(p_0|\,|p\right)\cdot N_0>1$, alors, grâce à la propriété de contraction de f sur $\left]\frac{1}{2D(p_0||p)};+\infty\right[$, les N_i appartiendront tous à cet intervalle et la suite N_i convergera vers une valeur N_∞. Comme f est décroissante, on a que pour tous i,j, $N_{2i}\leq N_\infty\leq N_{2j+1}$.

Prenons donc $N_0\stackrel{\text{def}}{=}\frac{1}{D(p_0||p)}$ qui est l'expression la plus simple appartenant à l'intervalle en question. On réécrit f en

$$\begin{aligned} f(x) &= -\frac{\ln(\lambda\beta)}{D\left(p_0|\,|p\right)}-\frac{\ln(x)}{2D\left(p_0|\,|p\right)} \\ &= -N_0\,\ln(\lambda\beta)-\frac{N_0}{2}\,\ln(x). \end{aligned}$$

On a alors

$$\begin{aligned} N_1 &= -N_0\,\ln(\lambda\beta)-\frac{N_0}{2}\,\ln(N_0) \\ N_2 &= -N_0\,\ln(\lambda\beta)-\frac{N_0}{2}\,\ln(N_1) \\ &= N_1+\frac{N_0}{2}\,\ln(N_0)-\frac{N_0}{2}\,\ln(N_1) \\ &= N_1+\frac{N_0}{2}\ln(N_0/N_1). \end{aligned}$$

Si on note $\theta\stackrel{\text{def}}{=}\frac{N_1}{N_0}$, on a alors l'encadrement suivant de N_∞

$$N_1\cdot\left[1-\frac{\ln(\theta)}{2\theta}\right]\leq N_\infty\leq N_1. \tag{6.19}$$

L'approximation donnée par (6.16) est l'expression suivante de N_1.

$$N_1=-\frac{1}{D\left(p_0|\,|p\right)}\cdot\ln\left(\frac{\lambda\beta}{\sqrt{D\left(p_0|\,|p\right)}}\right).$$

Pour ce qui est de l'approximation (6.15), elle est obtenue en prenant

$$N_2=-\frac{1}{D\left(p_0|\,|p\right)}\left[\ln\left(\frac{\lambda\beta}{\sqrt{D\left(p_0|\,|p\right)}}\right)+\frac{1}{2}\ln\left(-\ln\left(\frac{\lambda\beta}{\sqrt{D\left(p_0|\,|p\right)}}\right)\right)\right].$$

On a l'encadrement $N_2\leq N_\infty\leq N_3$. De la même façon que précédemment, on exprime N_3 en fonction de N_2 :

$$N_3=N_2\cdot\left[1+\frac{N_0}{2N_1}\cdot\ln\left(\frac{N_1}{N_2}\right)\right].$$

Comme la fonction f est contractante, $N_1/N_2<\theta$ et donc on obtient un encadrement de N_∞ plus précis que (6.19). □

Remarque. On remarque que N' est une fonction décroissante de $D(p_0||p)$. Comparer la quantité d'information apportée par deux phénomènes statistiques revient donc à comparer les divergences de Kullback-Leibler correspondantes. On peut ainsi, dans un premier temps, comparer le potentiel de deux attaques facilement. Il faudra ensuite prendre en compte les complexités liées à l'extraction et au traitement de l'information pour être plus précis.

On va simplifier l'expression (6.16) du théorème 6.13 qui utilise une valeur λ dont l'expression est assez complexe. Cette nouvelle expression pour N'' découle du lemme 6.14.

Lemme 6.14.

$$\ln(\lambda) \underset{p \to p_0}{\sim} \ln(2\sqrt{\pi D(p_0||p)}).$$

Démonstration : La preuve découle simplement du développement limité de la divergence $D(p_0||p)$ pour p proche de p_0 lemme 6.16. □

On obtient alors une nouvelle formule pour estimer N.

Corollaire 6.15. *Une bonne approximation de la formule (6.16) donnée dans le théorème 6.13 est la suivante :*

$$N'' = -\frac{\ln(2\sqrt{\pi}\beta)}{D(p_0||p)} \tag{6.20}$$

6.4.4 Comparaisons des formules

On va maintenant comparer les résultats obtenus en utilisant (6.15), (6.16) et les techniques présentes dans la littérature à la vraie valeur de N. Avant toute chose, commençons par détailler et commenter les résultats de la littérature avec lesquels nous effectueront les comparaisons. Comme nous souhaitons comparer ces travaux et nos formules (6.15) et (6.16), on prendra $\alpha = 0.5$ et l'on fera varier β.

[BJV04]. Dans ce travail, les auteurs donnent une formule liant N aux probabilités d'erreur α et β. Le résultat est obtenu en effectuant des approximations justifiées quand les deux distributions à distinguer sont proches.

$$N \approx \frac{2 \cdot \Phi^{-1}(\frac{\alpha+\beta}{2})^2}{D(p_0||p)}.$$

Le problème ici est que ce travail s'intéresse à la probabilité d'erreur globale $\frac{\alpha+\beta}{2}$. Or nos formules sont obtenues pour $\alpha \approx 0.5$ et β petit. Dans ce cas, les variations de β n'influencent pas beaucoup la probabilité d'erreur globale et donc les valeurs obtenues pour N sont très proches.

[Sel08]. On en a déjà beaucoup parlé, cet article donne une formule de calcul de la probabilité de succès pour des cryptanalyses linéaires et différentielles. On a insisté sur le fait que, contrairement à la cryptanalyse linéaire, la formule pour la différentielle est assez mauvaise et donc, on va donner un nouvel exemple de cela. Pour la cryptanalyse linéaire, on va devoir modifier un peu la formule car les formules données dans ce document et dans [BJV04] ne prennent pas en compte le fait que pour les attaques sur le dernier tour, on regarde la valeur

absolue d'un biais et non pas l'ordre des compteurs. Cela revient changer le 2^{-a-1} du papier en 2^{-a} (ce qui se comprend assez bien intuitivement). Avec nos notations, cela donne

$$N \approx \left(\frac{\Phi^{-1}(\alpha) + \Phi^{-1}(\beta)}{2|p_0 - p|} \right)^2 .$$

Pour la formule de la différentielle, on garde la même que dans l'article et donc

$$N \approx p \left(\frac{\sqrt{\frac{p_0}{p}}\Phi^{-1}(\alpha) + \Phi^{-1}(\beta)}{p_0 - p} \right)^2 .$$

Comparaisons. On peut maintenant regarder les résultats obtenus pour différents jeux de paramètres (avec α fixé à 0.5). Les graphes suivants représentent l'évolution du logarithme en base 2 du nombre d'échantillons nécessaires pour distinguer deux distributions binomiales de paramètres p_0 et p avec des probabilités d'erreur $\alpha \approx 0.5$ et β (β variant en abcisse).

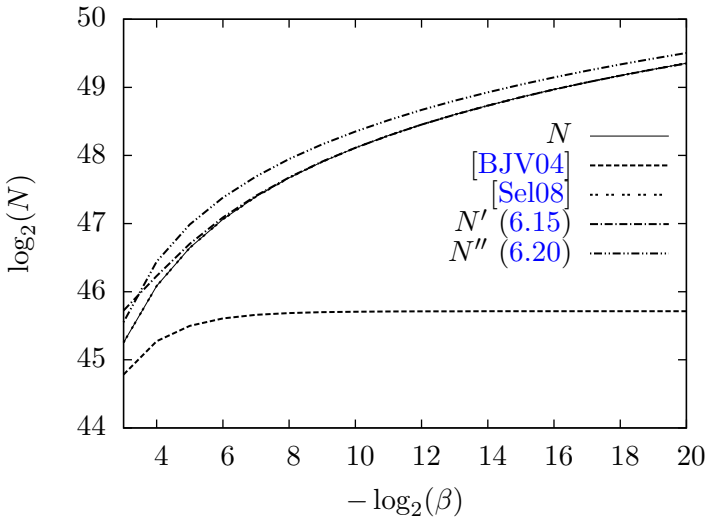

FIGURE 6.7 – Cryptanalyse linéaire avec $p_0 = \frac{1}{2} + 1.49 \cdot 2^{-24}$ et $p = \frac{1}{2}$.

Le calcul de N a été effectué en utilisant l'algorithme 11 pour calculer la probabilité de fausse alarme. On voit bien que les formules (6.15) et (6.20) se comportent bien dans tous les cas ce qui n'est pas le cas des autres qui ne sont bonnes que ponctuellement. On observe aussi sur ces courbes le fait (mentionné dans la preuve du théorème 6.13) que les bornes se bonifient quand β qui diminue.

6.4.5 Application à diverses cryptanalyses

On a vu que la complexité en données d'une cryptanalyse statistique est principalement déterminée par $D(p_0|\,|p)^{-1}$. Cela a déjà été mis en avant dans [BJV04, BV08] par exemple. Ce que nous proposons dans cette sous-section est d'effectuer le développement limité de la divergence de Kullback-Leibler pour les paramètres correspondant aux différentes cryptanalyses statistiques évoquées dans ce document. On retrouvera alors les comportements de la

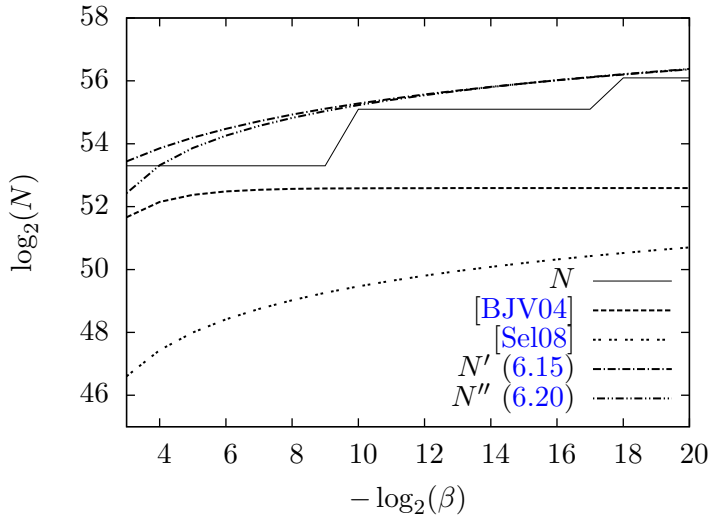

FIGURE 6.8 – Cryptanalyse différentielle avec $p_0 = 1.87 \cdot 2^{-56}$ et $p = 2^{-64}$.

complexité en données connus pour certaines cryptanalyses, pour d'autre, cela aura le mérite de préciser une intuition et enfin, pour la cryptanalyse différentielle tronquée, cela permettra de combler le manque d'une formule asymptotique pour cette complexité, absence qui a été l'élément déclencheur de ce travail.

Commençons d'abord par développer le plus possible la divergence de Kullback-Leibler. On utilisera ensuite ce résultat pour trouver les formules spécifiques correspondant aux différents paramètres des cryptanalyses étudiées. Le résultat suivant est donné par le lemme 6.16 en annexe B.

Lemme 6.16. *Soient* $0 < p < p_0 < 1$ *tels que* $\frac{p_0-p}{1-p} = \mathcal{O}(p_0 - p)$. *Alors,*

$$D(p_0||p) = p_0 \cdot \left[\ln\left(\frac{p_0}{p}\right) - \frac{p_0 - p}{p_0} + \frac{(p_0-p)^2}{2p_0(1-p_0)}\right] + \mathcal{O}(p_0 - p)^3 . \tag{6.21}$$

Démonstration : Voir annexe B.1. □

On va maintenant, comme promis, regarder ce que donne le développement de $D(p_0||p)^{-1}$ pour différentes attaques.

Cryptanalyse linéaire. On l'a vu en détail dans le chapitre 4, dans le cas de la cryptanalyse linéaire, p_0 est proche de $p = 1/2$. Le lemme 6.17 nous donne une formule dans un cas particulier qui nous intéressera plus tard et qui correspond, entre autre, à la cryptanalyse linéaire.

Lemme 6.17. *Soit un réel* $0 < a < 1$, *et soit* $\epsilon > 0$ *un nombre réel tel que* $\frac{\epsilon}{a} = \mathcal{O}(\epsilon)$ *et* $\frac{\epsilon}{1-a} = \mathcal{O}(\epsilon)$. *Alors,*

$$D(a + \epsilon||a) = \frac{\epsilon^2}{2a(1-a)} + \mathcal{O}\left(\epsilon^3\right).$$

Démonstration : Voir annexe B.1. □

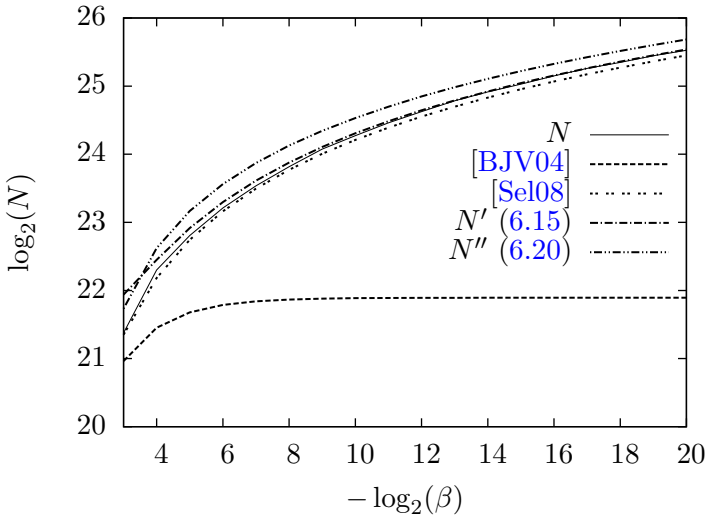

FIGURE 6.9 – Cryptanalyse différentielle tronquée avec $p_0 = 1.18 \cdot 2^{-16}$ et $p = 2^{-16}$.

On obtient $D(p_0||p) = 2\varepsilon^2 + \mathcal{O}(\varepsilon^3)$. On retrouve bien le « fameux » ε^{-2} donné par Matsui [Mat94b, Mat94a] et retrouvé ensuite dans de diverses analyses.

Cryptanalyse différentielle. Dans ce cas, p_0 et p sont proches de 0, comme l'on a vu au chapitre 5, mais $p_0 \gg p$ ce qui fait que l'on ne peut pas développer $\ln(p_0/p)$. On a, en revanche, $p_0 - p = p_0 + \mathcal{O}(p)$. Ce qui donne

$$\begin{aligned} D(p_0||p) &= p_0\left[\ln\left(\frac{p_0}{p}\right) - 1 - \frac{p}{p_0} + \frac{p_0 + \mathcal{O}(p)}{2(1-p_0)}\right] + \mathcal{O}(p_0 - p)^3 \\ &= p_0\left[\ln\left(\frac{p_0}{p}\right) - 1\right] + \mathcal{O}(p). \end{aligned}$$

Le résultat est donc légèrement différent du résultat standard i.e. $1/p_0$ que l'on trouve, par exemple, dans [LMM91] car il implique $\ln(p_0/p)$. Cette dépendance s'explique tout naturellement par le fait que $1/p_0$ est obtenu en supposant que le rapport signal sur bruit est suffisamment grand. Or, ce dernier vaut soit $\frac{p_0}{p}$ soit $\frac{p_0-p}{p}$. La formule précise donc ce que l'on entend par « rapport signal sur bruit suffisamment élevé ». Il s'agit du cas $\ln(p_0/p) \geq 2$.

Cryptanalyse différentielle-linéaire Cette attaque présentée dans la sous-section 5.2.4 a des paramètres qui correspondent à la cryptanalyse linéaire. En effet, $p = \frac{1}{2}$ et p_0 est proche de p et on peut donc l'écrire $p_0 = p + \varepsilon$. C'est pourquoi le résultat est le même que pour la cryptanalyse linéaire.

Cryptanalyse différentielle tronquée. Le cas de la cryptanalyse différentielle tronquée de la sous-section 5.2.4 est intéressant car il n'existait pas de formule générique pour la complexité en données d'une telle attaque. Cette absence de formule a motivé le travail présenté dans ce chapitre. Ici, $\frac{p_0-p}{p}$ n'est pas forcément un $\mathcal{O}(p_0 - p)$ et on ne peut donc pas directement appliquer le lemme 6.17. Cependant, comme p_0 et p sont proches, on peut utiliser

le même développement du logarithme (B.2) utilisé dans sa preuve. On obtient alors

$$\begin{aligned} D\left(p_0|\,|p\right) &= p_0(p_0-p)\left[\frac{1}{p}-\frac{1}{p_0}\right]+\frac{p_0(p_0-p)^2}{2}\left[\frac{1}{p_0(1-p_0)}-\frac{1}{p^2}\right]+\mathcal{O}\left(p_0-p\right)^3 \\ &= \frac{(p_0-p)^2}{p}+\frac{(p_0-p)^2}{2p^2(1-p_0)}\left[p^2-p_0+p_0^2\right]+\mathcal{O}\left(p_0-p\right)^3 \\ &= \frac{(p_0-p)^2}{p}\left[1-\frac{p_0}{2p}\right]+\mathcal{O}\left(p_0-p\right)^2 \\ &= \frac{(p_0-p)^2}{2p}+\mathcal{O}\left(p_0-p\right)^2. \end{aligned}$$

Cryptanalyses	Comportement du nombre d'échantillons	Comportement du nombre de clairs	Clair choisi ou connu (**CP/KP**)
Linéaire	$\frac{1}{2(p_0-p)^2}$	$\frac{1}{2(p_0-p)^2}$	**KP**
Différentielle	$\frac{1}{p_0\ln(p_0/p)-p_0}$	$\frac{2}{p_0\ln(p_0/p)-p_0}$	**CP**
Différentielle-linéaire	$\frac{1}{2(p_0-p)^2}$	$\frac{1}{(p_0-p)^2}$	**CP**
Différentielle tronquée	$\frac{2p}{(p_0-p)^2}$	$\frac{2p\cdot\gamma}{(p_0-p)^2}$, $1<\gamma<2$	**CP**
Différentielle d'ordre i	$-\frac{1}{\ln p}$	$-\frac{2^i}{\ln p}$	**CP**
Différentielle impossible	$\frac{1}{p}$	$\frac{2}{p}$	**CP**

TABLE 6.1 – Complexité en données asymptotique pour quelques cryptanalyses statistiques.

Cryptanalyse différentielle d'ordre supérieur. On a vu le principe de l'attaque dans la sous-section 5.2.3. On avait alors un phénomène qui se produisait avec probabilité 1 et qui nécessitait des échantillons de 2^i clairs où i était supérieur au degré de la fonction. On est dans un cas particulier où il ne sert à rien de faire un développement limité. En effet, $D\left(1|\,|p\right) = -\ln(p)$. Dans la littérature, on trouve qu'il suffit d'un échantillon pour mener l'attaque. Notre formule va dans ce sens car $-1/\ln(p) > 1 \Leftrightarrow p > e^{-1}$.

Cryptanalyse différentielle impossible. Pour finir, un cas encore plus particulier car ici, comme vu dans la sous-section 5.2.2, on a $p_0 < p$. On peut refaire tout le travail de ce chapitre en prenant cette convention et on obtient des résultats similaires. Ici encore on reprend la définition de la divergence de Kullback-Leibler $D\left(0|\,|p\right) = -\ln(1-p) = p+\mathcal{O}\left(p^2\right)$.

Ce résultat est en adéquation avec ce qu'il est d'usage de lire. En prenant de l'ordre de $1/p$ échantillons, les mauvais compteurs auront en moyenne une valeur de 1 ce qui permettra de les distinguer du bon compteur qui sera resté à 0.

6.5 Probabilité de succès

Dans cette section on s'intéresse au problème de la probabilité de succès d'une attaque dans le cadre de l'approche à taille de liste fixée. Celle-ci est la plus utilisée aujourd'hui pour analyser les cryptanalyses et nous nous devions d'adapter le travail présenté jusqu'ici à cette approche. Dans le cas de la cryptanalyse linéaire, on a les travaux de [Sel08] qui donnent une formule valide. L'idée est d'utiliser la même stratégie afin de dériver une formule similaire sans utiliser l'approximation normale de la loi binomiale mais en utilisant, si besoin est, les approximations données au théorème 6.8.

Modèle et notations. On rappelle ici le modèle 6.1 et on introduit quelques notations. On a un certain nombre n de variables $\Sigma_{\boldsymbol{k}}$ obtenues en traitant N échantillons provenant d'une clef $\boldsymbol{k}^*$. Ces compteurs suivent des lois binomiales :

$$\Sigma_{\boldsymbol{k}^*} \sim \mathcal{B}(N, p_0) \quad \text{et} \quad \Sigma_{\boldsymbol{k}\neq\boldsymbol{k}^*} \sim \mathcal{B}(N, p).$$

On garde la liste $\mathcal{L}$ des ℓ compteurs les plus élevés[3]. L'attaque est un succès si la bonne clef appartient à la liste.

$$P_s \stackrel{\text{def}}{=} \Pr\left[\boldsymbol{k}^* \in \mathcal{L}\right].$$

La probabilité étant calculée sur l'ensemble des groupes de N échantillons pouvant être obtenus. On note

$$\begin{aligned} G(x) &\stackrel{\text{def}}{=} \Pr\left[\Sigma_{\boldsymbol{k}\neq\boldsymbol{k}^*} \leq x\right], \\ G^{-1}(y) &\stackrel{\text{def}}{=} \min\{x \in \mathbb{N} | G(t) \geq y\}, \\ g_0(x) &\stackrel{\text{def}}{=} \Pr\left[\Sigma_{\boldsymbol{k}^*} = x\right]. \end{aligned}$$

La fonction G^{-1} est la réciproque généralisée de F au sens où pour G bijective, G^{-1} correspond bien à la réciproque de G.

Formule de [Sel08]. En notant $\tilde{G}^{-1}$ et $\tilde{g}_0$ les approximations des fonctions G^{-1} et g_0 obtenues en utilisant la loi normale pour estimer la loi binomiale, la formule donnée par Selçuk peut s'écrire comme

$$P_S \approx \int_{\tilde{G}^{-1}\left(1-\frac{\ell}{n}\right)}^{\infty} \tilde{g}_0(x)\, dx. \tag{6.22}$$

La suite de cette section est principalement dédiée à la preuve du théorème suivant qui donne une approximation de P_S de la forme

$$P_S \approx \sum_{i=G^{-1}\left(1-\frac{\ell-1}{n-2}\right)}^{N} g_0(i).$$

3. Ce n'est pas le cas pour la cryptanalyse différentielle impossible mais ce cas particulier est très facile à traiter précisément. Ce n'est pas le cas non plus pour la cryptanalyse linéaire de type 2 vu que l'on s'intéresse à la valeur absolue du biais. On verra que l'on peut tout de même utiliser ce résultat dans ce cas.

Le théorème suivant donne des précisions sur cette approximation dans le cas où $P_S > 0,5$.

Théorème 6.18. *Soit P_S la probabilité de succès d'une cryptanalyse statistique qui garde ℓ candidats parmi n. Soit N le nombre d'échantillons disponibles issus d'un chiffrement utilisant une clef $\boldsymbol{k}^*$. On note g_0 la distribution du compteur correspondant à la bonne clef, G la fonction de répartition des compteurs correspondant aux mauvaises clefs et G^{-1} sa réciproque. On définit*

$$\begin{aligned} \lambda &\stackrel{\text{def}}{=} \frac{\ell - 1}{n - 2} = 1 - t_0 \\ B &\stackrel{\text{def}}{=} G^{-1}(1 - \lambda) \\ \delta &\stackrel{\text{def}}{=} \sum_{i=0}^{B-1} g_0(i) \\ C_\lambda &\stackrel{\text{def}}{=} \frac{p}{p_0} \frac{p_0(N+1) - B}{B - p(N+1)} \end{aligned}$$

Si $\lambda \leq \frac{1}{4}$ alors

$$P_S = 1 - \delta + \mathcal{O}\left(\delta(1 + C_\lambda)\sqrt{\frac{\ln(\ell/\delta^2)}{\ell}} + \frac{1}{\ell^2} + \frac{1}{n}\right). \tag{6.23}$$

Conditions. Avant de rentrer dans le vif du sujet, discutons des conditions d'applicabilité du théorème et du terme d'erreur. Tout d'abord, il faut noter que dans le cas où $P_S < 0,5$ la même preuve sera valable en prenant $\delta = \sum_{i=B+1}^{N} g_0(i)$. On a donc une formule qui est meilleure pour des valeurs de P_S éloignées de $0,5$.

Il s'avère que pour les paramètres qui nous intéressent en cryptographie, la valeur de C_λ est petite. Afin de ne pas complexifier la chose, on ne donne pas de borne générale sur C_λ mais il s'avère que c'est dans le cas de la cryptanalyse linéaire qu'on obtient les plus grandes valeurs de C_λ car la variance des compteurs est très élevée. Dans ce cas, l'approximation gaussienne est valide et on peut vérifier que

$$B \approx pN + x\sqrt{Np(1-p)},$$

où $x \stackrel{\text{def}}{=} \Phi^{-1}(1-\lambda)$. On a $x \underset{\lambda \to 0^+}{\sim} \sqrt{-2\ln\lambda}$. De plus, par définition de δ, l'on a aussi

$$B \approx p_0 N - y\sqrt{Np_0(1-p_0)},$$

où $y \stackrel{\text{def}}{=} \Phi^{-1}(1-\delta)$. En remarquant que $y \underset{\lambda \to 0^+}{\sim} \sqrt{-2\ln\delta}$ et en utilisant le fait que $p_0 \approx p$, on obtient

$$C_\lambda \approx \frac{p}{p_0} \frac{y\sqrt{Np_0(1-p_0)}}{x\sqrt{Np(1-p)}} \approx \sqrt{\frac{-\ln\delta}{-\ln\lambda}}.$$

On peut voir δ comme une approximation de $1 - P_S$ et donc on s'attend à ce qu'il soit petit. De son côté, λ aussi doit être petit afin d'avoir une complexité en temps raisonnable. Dans le cas, par exemple, où $\delta \approx 0,1$ et $\lambda \approx 2^{-3}$, on obtient $C_\lambda \approx 1$. Cette valeur diminuera avec ℓ.

La preuve se base sur la théorie des statistiques d'ordre présentée dans la section A.3. On trie les $n-1$ variables $\Sigma_{\boldsymbol{k}}$ par ordre croissant, on note $\Sigma_{(i)}$ les variables triées. On peut alors

réécrire la probabilité de succès comme la probabilité que le compteur de la bonne clef soit supérieur au ℓ-ième plus grand des $n-1$ autres compteurs.

$$P_S = \Pr\left[\Sigma_{\boldsymbol{k}^*} \geq \Sigma_{(n-\ell)}\right] = \sum_{i=0}^{N} g_0(i) \cdot \Pr\left[\Sigma_{(n-\ell)} \leq i\right].$$

En appliquant le lemme A.21, on obtient

$$P_S = \sum_{i=0}^{N} g_0(i) \cdot (n-1) \cdot \binom{n-2}{\ell-1} \cdot B_{n-\ell,\ell}(G(i)). \tag{6.24}$$

On rappelle que

$$B_{n-\ell,\ell}(x) \stackrel{\text{def}}{=} \int_0^x t^{n-\ell-1}(1-t)^{\ell-1}\, dt.$$

On notera b l'intégrande

$$b(t) \stackrel{\text{def}}{=} (n-1) \cdot \binom{n-2}{\ell-1} \cdot t^{n-\ell-1}(1-t)^{\ell-1}.$$

La suite de la démonstration se base sur l'observation suivante : pour des paramètres cryptographiques (i.e. n et ℓ grands), la fonction b est concentrée au point $\frac{n-\ell-1}{n-2}$ et donc l'intégrale se comporte presque comme une fonction par paliers. La figure 6.10 donne une illustration de ce phénomène. On note t_0 le point de concentration de b

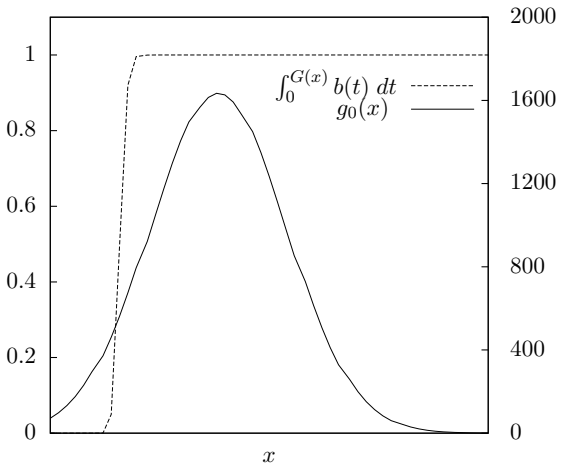

FIGURE 6.10 – Illustration de la concentration de l'intégrale de $b(t)$.

$$t_0 \stackrel{\text{def}}{=} \frac{n-\ell-1}{n-2}.$$

L'idée est alors de décomposer la somme en trois parties : une somme sur la zone où l'intégrale est très proche de 0, une sur l'intervalle où elle est proche de 1 et la troisième somme sur le

voisinage de t_0 sur lequel on ne sait pas dire grand chose mais qui contiendra peu de termes. Pour $\epsilon > 0$, on écrit donc

$$\begin{aligned} P_S &= \sum_{i=0}^{N} g_0(i) \int_0^{G(i)} b(t)\, dt \\ &= \sum_{i=0}^{G^{-1}(t_0-\epsilon)-1} g_0(i) \int_0^{G(i)} b(t)\, dt + \sum_{i=G^{-1}(t_0-\epsilon)}^{G^{-1}(t_0)-1} g_0(i) \int_0^{G(i)} b(t)\, dt \\ &+ \sum_{i=G^{-1}(t_0)}^{N} g_0(i) \int_0^{G(i)} b(t)\, dt \end{aligned}$$

Comme on l'a annoncé, la probabilité de succès est principalement influencée par la troisième somme. Sur cet intervalle, l'intégrale est proche de 1 ce qui nous fait écrire

$$\sum_{i=G^{-1}(t_0)}^{N} g_0(i) \int_0^{G(i)} b(t)\, dt = \sum_{i=G^{-1}(t_0)}^{N} g_0(i) - \sum_{i=G^{-1}(t_0)}^{N} g_0(i) \int_{G(i)}^{1} b(t)\, dt$$

La première des deux sommes est celle que l'on souhaite prendre comme estimation de la probabilité de succès. Le but est donc d'estimer l'erreur

$$\begin{aligned} P_S - \sum_{i=G^{-1}(t_0)}^{N} g_0(i) &= \underbrace{\sum_{i=0}^{G^{-1}(t_0-\epsilon)-1} g_0(i) \int_0^{G(i)} b(t)\, dt}_{S_1} \\ &+ \underbrace{\sum_{i=G^{-1}(t_0-\epsilon)}^{G^{-1}(t_0)-1} g_0(i) \int_0^{G(i)} b(t)\, dt}_{S_2} - \underbrace{\sum_{i=G^{-1}(t_0)}^{N} g_0(i) \int_{G(i)}^{1} b(t)\, dt}_{S_3} \end{aligned}$$

Pour la somme S_1, la majoration va venir du fait que l'intégrale a une valeur proche de 0. La même remarque peut s'appliquer à une partie de la somme S_3. C'est pourquoi on la décompose en

$$S_3 = \underbrace{\sum_{i=G^{-1}(t_0)}^{G^{-1}(t_0+\epsilon)-1} g_0(i) \int_{G(i)}^{1} b(t)\, dt}_{S_4} + \underbrace{\sum_{i=G^{-1}(t_0+\epsilon)}^{N} g_0(i) \int_{G(i)}^{1} b(t)\, dt}_{S_5}.$$

Pour résumer, on cherche à majorer l'erreur $|S_1 + S_2 - S_4 - S_5|$.

6.5.1 Majoration de $|S_1 - S_5|$.

Commençons par la partie la plus intuitive. Pour S_1 et S_5, on a une intégrale dont la valeur est très petite. Sur l'espace de sommation de S_1, on a $\int_{G(i)}^{1} b(t)\, dt \leq \int_{t_0-\epsilon}^{1} b(t)\, dt$. De plus, $\sum_{i=0}^{G^{-1}(t_0-\epsilon)-1} g_0(i) \leq 1$. On effectue la même chose pour S_5 et on obtient

$$S_1 \leq \int_0^{t_0-\epsilon} b(t)\, dt \quad , \quad S_5 \leq \int_{t_0+\epsilon}^{1} b(t)\, dt.$$

Ce qui donne

$$|S_1 - S_5| \leq S_1 + S_5 = 1 - \int_{t_0-\epsilon}^{t_0+\epsilon} b(t)\ dt.$$

Comme on l'a dit, l'intégrale est concentrée autour de t_0 et donc $\int_{t_0-\epsilon}^{t_0+\epsilon} b(t)\ dt$ est proche de 1. On calcule, dans le lemme 6.19 l'ordre de $1 - \int_{t_0-\epsilon}^{t_0+\epsilon} b(t)\ dt$.

Lemme 6.19. *Soit g la densité de la distribution bêta ayant pour paramètres $(n-\ell-1, \ell-1)$:*

$$b(t) \stackrel{\text{def}}{=} (n-1) \cdot \binom{n-2}{\ell-1} \cdot t^{n-\ell-1}(1-t)^{\ell-1}.$$

Son maximum est atteint en $t_0 \stackrel{\text{def}}{=} \frac{n-\ell-1}{n-2}$. Soit $\epsilon \stackrel{\text{def}}{=} z \cdot \frac{\sqrt{\ell-1}}{n-2}$.Alors, si $z = o\left(\sqrt{\ell}\right)$ et $\ell \in [0; \frac{n}{2}]$, alors,

$$\int_{t_0-\epsilon}^{t_0+\epsilon} b(t)\ dt = 1 + \mathcal{O}\left(\frac{1}{\ell^2} + \frac{1}{n} + \frac{e^{-z^2/2}}{z}\right).$$

Démonstration : Voir annexe B.2. □

On a donc

$$|S_1 - S_5| = \mathcal{O}\left(\frac{1}{\ell^2} + \frac{1}{n} + \frac{e^{-z^2/2}}{z}\right).$$

6.5.2 Majoration de $|S_2 - S_4|$.

Pour S_2 et S_4, on l'a déjà dit, la majoration va venir du fait qu'il y a peu de termes dans la somme. On peut donc majorer l'intégrale par 1 et obtenir $S_2 \leq \sum_{i=G^{-1}(t_0-\epsilon)}^{G^{-1}(t_0)-1} g_0(i)$. Pour simplifier les formules, on note $B \stackrel{\text{def}}{=} G^{-1}(t_0)$ et $B_\epsilon \stackrel{\text{def}}{=} G^{-1}(t_0 - \epsilon)$.

Lemme 6.20. *On note $\delta \stackrel{\text{def}}{=} \sum_{i=0}^{B-1} g_0(i)$. Soit $\epsilon \stackrel{\text{def}}{=} z \cdot \frac{\sqrt{\ell-1}}{n}$ pour un certain réel $z = o\left(\sqrt{\ell}\right)$. Alors, si $Np < B < Np_0$, ,*

$$S_2 \stackrel{\text{def}}{=} \sum_{i=B_\epsilon}^{B-1} g_0(i) = \mathcal{O}\left(z \cdot \frac{C_\lambda \delta}{\sqrt{\ell-1}}\right).$$

Tout d'abord, notons que si $B = B_\epsilon$ alors le lemme est prouvé. On va donc supposer que $B \geq B_\epsilon + 1$. La preuve de ce lemme est fondée sur le lemme 6.9. On va donc utiliser les coefficients $\gamma_0 \stackrel{\text{def}}{=} \frac{(1-p_0)\cdot B}{p_0\cdot(N-B+1)}$ et $\gamma \stackrel{\text{def}}{=} \frac{(1-p)\cdot B}{p\cdot(N-B+1)}$. Avant d'entrer dans la preuve du lemme 6.20, l'on va s'intéresser à prouver le lemme technique suivant qui sera utilisé dans cette preuve.

Lemme 6.21. *Si $\frac{\ell-1}{n-2} \leq \frac{1}{4}$ et ϵ est choisit de la forme $\epsilon = z\frac{\sqrt{\ell-1}}{n}$, avec $z = o\left(\sqrt{\ell}\right)$ pour ℓ tendant vers l'infini, alors*

$$(B - B_\epsilon)(\gamma - 1) = \mathcal{O}\left(\frac{z}{\sqrt{\ell-1}}\right).$$

Démonstration : Premièrement, en utilisant le lemme 6.9, on peut facilement montrer que

$$\sum_{i=B+1}^{N} g(i) = \Theta\left(\frac{g(B)}{1-1/\gamma}\right) = \Theta\left(\gamma\,\frac{g(B)}{\gamma-1}\right). \tag{6.25}$$

D'autre part,

$$(\gamma_-^{B-B_\epsilon} - 1)\,\frac{g(B)}{\gamma_- - 1} \leq \sum_{B_\epsilon+1}^{B} g(i), \tag{6.26}$$

avec $\gamma_- \stackrel{\text{def}}{=} \frac{1-p}{p} \min\left(\frac{B}{N-B+1}, \frac{B_\epsilon+2}{N-B_\epsilon-1}\right)$. De l'hypothèse sur ϵ, on obtient que $\sum_{i=B_\epsilon+1}^{B} g(i) = o\left(\sum_{i=B+1}^{N} g(i)\right)$ pour ℓ qui tend vers l'infini. Cela n'est possible que si $\gamma_-^{B-B_\epsilon} - 1$ tend vers 0 avec ℓ qui tend vers l'infini. Ce qui implique que $\gamma_-^{B-B_\epsilon} - 1 \sim (B-B_\epsilon)(\gamma_- - 1)$. Comme pour ℓ suffisamment grand, γ et γ_- coïncident, on peut remplacer γ_- par γ ce qui, avec (6.25) et (6.26), implique que

$$\begin{aligned}(B-B_\epsilon)(\gamma-1) &\underset{\ell\to\infty}{\sim} \gamma_-^{B-B_\epsilon} - 1 \\ &= \mathcal{O}\left(\sum_{i=B_\epsilon+1}^{B} g(i)\frac{\gamma-1}{g(B)}\right) \\ &= \mathcal{O}\left(\frac{\sum_{i=B_\epsilon+1}^{B} g(i)}{\sum_{i=B+1}^{N} g(i)}\right).\end{aligned}$$

On exprime alors les deux sommes apparaissant dans la fraction en fonction de ϵ et t_0

$$\sum_{i=B_\epsilon+1}^{B} g(i) = G(G^{-1}(t_0)) - G(G^{-1}(t_0-\epsilon)) = \epsilon\left[1 + \mathcal{O}\left(\frac{g(B_\epsilon)}{\epsilon}\right)\right],$$

et

$$\sum_{i=B+1}^{N} g(i) = 1 - G(G^{-1}(t_0)) = (1-t_0)\left[1 + \mathcal{O}\left(\frac{g(B)}{1-t_0}\right)\right].$$

On obtient enfin,

$$(B-B_\epsilon)(\gamma-1) = \mathcal{O}\left(\frac{\epsilon}{1-t_0}\left[1 + \mathcal{O}\left(\frac{g(B_\epsilon)}{\epsilon}\right)\right]\right).$$

On voit facilement que $\frac{g(B_\epsilon)}{\epsilon} = \mathcal{O}(1)$. On peut alors substituer les valeurs prises pour ϵ et t_0 et l'on obtient

$$(B-B_\epsilon)(\gamma-1) = \mathcal{O}\left(\frac{z}{\sqrt{\ell-1}}\right).$$

□

On peut maintenant prouver le lemme 6.20.

Démonstration : En utilisant le lemme 6.9, on peut obtenir

$$\sum_{i=0}^{B-1} g_0(i) = \Theta\left(\frac{g_0(i)}{1-\gamma_0}\right), \tag{6.27}$$

$$\sum_{i=B_\epsilon}^{B} g_0(i) = \Theta\left(\frac{(1-\gamma_0^{B-B_\epsilon})g_0(i)}{1-\gamma_0}\right). \tag{6.28}$$

Observons aussi que

$$\begin{aligned} 1-\gamma_0^{B-B_\epsilon} &= O\left((B-B_\epsilon)(1-\gamma_0)\right) \\ &= \mathcal{O}\left(C_\lambda(\gamma-1)(B-B_\epsilon)\right) \\ &= \mathcal{O}\left(C_\lambda \frac{z}{\sqrt{\ell-1}}\right). \end{aligned}$$

En assemblant (6.27), (6.28) et le fait que $\delta \stackrel{\text{def}}{=} \sum_{i=0}^{B-1} g_0(i)$, on obtient le lemme 6.20. □

6.5.3 Résultats expérimentaux et observations

Au tableau 6.2, on s'intéresse à comparer la formule donnée par le théorème 6.18 pour la probabilité de succès

$$P_s \approx \sum_{i=G^{-1}(1-\frac{\ell-1}{n-2})}^{N} g_0(i) \tag{6.29}$$

à la formule de Selçuk (6.22) et à la vraie probabilité de succès obtenue en effectuant un long et minutieux calcul numérique pour évaluer (6.24) avec une précision d'au moins 4 digits. Le but est de bien montrer que l'utilisation de l'approximation normale pour calculer la probabilité de succès n'est pas forcément pertinente ce qui, bien qu'expliqué dans [Sel08], n'a apparemment pas été pris en compte dans les papiers de cryptanalyse différentielle. Dans le

Type de cryptanalyse	Probabilités	Paramètres $N = 2^{48}$ $n = 2^{20}$	P_S	Estimation (6.29)	Estimation (6.22)
Linéaire	$p = 0.5$ $p_0 = p + 1.49 \cdot 2^{-24}$	$\ell = 2^{15}$	0.8681	0.8681	0.8681
Linéaire	$p = 0.5$ $p_0 = p + 1.49 \cdot 2^{-24}$	$\ell = 2^{10}$	0.4533	0.4533	0.4533
Différentielle	$p = 2^{-64}$ $p_0 = 2^{-47.2}$	$\ell = 2^{15}$	0.8257	0.8247	0.9050
Différentielle	$p = 2^{-64}$ $p_0 = 2^{-47.2}$	$\ell = 2^{10}$	0.8250	0.8247	0.9050

TABLE 6.2 – Comparaisons entre l'estimation présentée dans ce document (6.29), l'estimation (6.22) donnée dans [Sel08] et la vraie valeur de P_S.

cas de la cryptanalyse linéaire, l'approximation gaussienne est valide et donc l'expression de Selçuk est égale à la vraie valeur tout comme notre formule. Cependant, dans le cas de la

cryptanalyse différentielle, on voit bien que ce n'est plus le cas. On observe, d'ailleurs, un effet de seuil inhérent à la cryptanalyse différentielle : l'augmentation de la taille de la liste n'a pas autant d'effets sur la probabilité de succès qu'en cryptanalyse linéaire.

Avant de conclure, nous allons nous attarder sur une observation en forme de question ouverte. On a bien vu que les valeurs ℓ, N et P_S étaient liées et on pourrait donc vouloir avoir une formule pour N en fonction des deux autres paramètres. Bien que nous n'ayons pas (encore) de résultats théoriques sur le sujet, une formule semble particulièrement appropriée. Il s'agit de reprendre l'estimation N'' donnée dans le corollaire 6.15 en remplaçant la probabilité de fausse alarme β par son pendant dans l'approche taille de liste fixée : $\frac{\ell-1}{n}$. On peut alors écrire N comme

$$N = -c \cdot \frac{\ln(2\sqrt{\pi}\frac{\ell-1}{n})}{D\left(p_0 \middle| p\right)}. \tag{6.30}$$

On remarque alors que pour des valeurs de N de cette forme, la probabilité de succès semble dépendre uniquement de c. Plus précisément, le tableau 6.3 contient les valeurs de P_S obtenues avec (6.29) pour différents paramètres de cryptanalyses et des valeurs de N de la forme donnée par (6.30). On remarque que dans chacune des deux colonnes correspondant à $c = 1$ et $c = 1.5$, les probabilités de succès des différentes cryptanalyses avec différentes valeurs de ℓ sont toutes du même ordre. On peut donc se risquer à la conjecture suivante.

	$c = 1$			$c = 1.5$		
Paramètres	ℓ			ℓ		
	2^{10}	2^{20}	2^{30}	2^{10}	2^{20}	2^{30}
$p = 0.5$ $p_0 = p + 1.49 \cdot 2^{-24}$	0.5855	0.5898	0.5949	0.9799	0.9687	0.9500
$p = 0.5$ $p_0 = p + 1.23 \cdot 2^{-11}$	0.5856	0.5899	0.5950	0.9799	0.9687	0.9500
$p = 2^{-30}$ $p_0 = 1.2 \cdot 2^{-30}$	0.5802	0.5921	0.5875	0.9766	0.9650	0.9446
$p = 2^{-40}$ $p_0 = 1.2 \cdot 2^{-40}$	0.5802	0.5827	0.5875	0.9766	0.9650	0.9446
$p = 2^{-64}$ $p_0 = 2^{-60}$	0.5801	0.5257	0.5544	0.9249	0.9070	0.8844
$p = 2^{-32}$ $p_0 = 2^{-29}$	0.5993	0.6179	0.6443	0.9605	0.9375	0.9241

TABLE 6.3 – Probabilités de succès pour de paramètres correspondant à plusieurs cryptanalyses avec $n = 2^{60}$ et $N = -c \cdot \frac{\ln(2\sqrt{\pi}\frac{\ell-1}{n})}{D(p_0||p)}$.

Conjecture 6.22. *Une complexité en données de* $N = -\frac{\ln(2\sqrt{\pi}\frac{\ell-1}{n})}{D(p_0||p)}$ *garantit une probabilité de succès d'au moins* 0.5 *pour une cryptanalyse statistique cherchant à distinguer une variable suivant une loi binomiale* (N, p_0) *de* $n-1$ *variables suivant une loi binomiale* (N, p) *et renvoyant les* ℓ *candidats les plus vraisemblables.*

De même, une complexité en données de $N = -1.5 \cdot \frac{\ln(2\sqrt{\pi}\frac{\ell-1}{n})}{D(p_0||p)}$ *garantit une probabilité de succès supérieure à* 0.9.

Bien évidemment, on aimerait surtout avoir une formule pour c en fonction de P_S et c'est la question qui, à cette heure, reste ouverte.

6.6 Conclusion

Nous allons ici résumer ce chapitre assez technique qui contient des résultats importants en ce qui concerne l'analyse des cryptanalyses statistiques simples.

Tout d'abord, nous avons vu que l'utilisation de l'approximation gaussienne de la loi binomiale n'est pas pertinente pour tous les types de cryptanalyses statistiques ce qui est le point de départ de ces travaux. Les premiers résultats sur la complexité en données proviennent de l'approche à seuil fixé des cryptanalyses statistique et nous a permis d'obtenir deux estimations de la complexité en données la seconde étant une simplification de la première avec une petite perte de précision.

$$\begin{aligned} N' &\stackrel{\text{def}}{=} -\frac{1}{D\left(p_0 \| p\right)}\left[\ln\left(\frac{\lambda\beta}{\sqrt{D\left(p_0 \| p\right)}}\right) + \frac{1}{2}\ln\left(-\ln\left(\frac{\lambda\beta}{\sqrt{D\left(p_0 \| p\right)}}\right)\right)\right], \\ N'' &\stackrel{\text{def}}{=} -\frac{\ln(2\sqrt{\pi}\beta)}{D\left(p_0 \| p\right)}, \end{aligned}$$

où $\lambda \stackrel{\text{def}}{=} \frac{(p_0-p)\sqrt{2\pi(1-p_0)}}{(1-p)\sqrt{p_0}}$.

On a ensuite utilisé l'approche taille de liste fixée où l'on garde une liste des ℓ candidats les plus vraisemblables. On définit alors la probabilité de succès P_S qui est la probabilité de garder le bon candidat dans la liste. Sous certaines conditions on peut prouver (théorème 6.18) que

$$P_S \approx \sum_{i=G^{-1}\left(1-\frac{\ell-1}{n-2}\right)}^{N} g_0(i).$$

Avec un terme d'erreur qui diminue quand ℓ augmente et quand P_S s'éloigne de $0,5$.

Enfin, on a remarqué, en utilisant cette formule, que pour une complexité en données de la forme

$$N = -c \cdot \frac{\ln(2\sqrt{\pi}\frac{\ell-1}{n})}{D\left(p_0 \| p\right)},$$

l'ordre de grandeur de la probabilité de succès d'une attaque semble dépendre uniquement de la valeur de c. Tous ces résultats sont importants car ces formules sont valables quelle que soit la cryptanalyse statistique considérée.

Chapitre 7

Retrouver la clef lors d'une cryptanalyse linéaire multiple

Sommaire

Lors d'une cryptanalyse linéaire utilisant plusieurs approximations, les deux questions principales sont

- Comment traiter les informations fournies par les différentes approximations ?
- Quelle est la quantité de données nécessaires à l'attaque ?

Dans le chapitre 4, on a vu que lorsque l'on choisit un espace vectoriel d'approximations, les outils venus des fonctions booléennes peuvent être utilisés et permettent d'analyser l'attaque sans faire l'hypothèse d'indépendance statistique entre les approximations. Dans ce document, nous nous intéressons au cas où l'on a intérêt à choisir des approximations de façon non structurée. La question de la quantité de données est traitée dans le chapitre 8. Quant à la première des deux problématiques, elle fait l'objet de ce chapitre.

Nous verrons tout d'abord qu'une telle attaque peut être modélisée par un problème de décodage en liste d'un code linéaire binaire de faible rendement sur un canal gaussien, puis, nous analyserons un algorithme de décodage par syndrome proposée par Antoine Valembois [Val00a, Val00b] pour le décodage de tels codes.

7.1 Cryptanalyse linéaire et canal gaussien

On a présenté le canal binaire à bruit blanc additif gaussien dans la sous-section 3.1.3. On va voir, maintenant, que la cryptanalyse linéaire multiple de type 1 peut être vue comme un

problème de décodage d'un code linéaire aléatoire sur un tel canal.

7.1.1 Cryptanalyse linéaire simple

L'on a vu dans le chapitre 4, que pour une approximation linéaire

$$\Pr_{\boldsymbol{M},\boldsymbol{K}}\left[\langle \pi, \boldsymbol{M}\rangle \oplus \langle \kappa, \boldsymbol{K}\rangle = \langle \gamma, E_{\boldsymbol{K}}(\boldsymbol{M})\rangle\right] = \frac{1}{2} + \varepsilon,$$

l'information provenant de N couples clair/chiffré était extraite dans le compteur défini par $t \stackrel{\text{def}}{=} \sum_{i=1}^{N} < \langle \pi, \boldsymbol{m}^i\rangle \oplus \langle \gamma, E_{\boldsymbol{k}}\left(\boldsymbol{m}^i\right)\rangle$. On notera $\boldsymbol{c} \stackrel{\text{def}}{=} E_{\boldsymbol{k}}(\boldsymbol{m})$ de façon à alléger les notations. Sous l'hypothèse d'équivalence de clé fixée, et si les $\boldsymbol{m}^i$ ont été tirés de façon indépendante sur l'ensemble des messages, alors les variables $\langle \pi, \boldsymbol{M}^i\rangle \oplus \langle \gamma, \boldsymbol{C}^i\rangle$ sont indépendantes et identiquement distribuées : elles suivent une loi de Bernoulli de paramètre $\frac{1}{2} + (-1)^{\langle \kappa, \boldsymbol{K}\rangle}\varepsilon$. On peut donc appliquer le théorème central limite afin de montrer que la distribution de la variable T correspondant au compteur est proche d'une distribution gaussienne. Cependant, on ne va pas regarder T mais

$$Y \stackrel{\text{def}}{=} \frac{N - 2\,T}{2N\varepsilon}.$$

En effet, il s'avère que travailler avec ce compteur va nous permettre de modéliser l'attaque comme un problème de décodage sur un canal gaussien.

Modèle 7.1. *Cryptanalyse linéaire simple (type 1).*
L'attaquant reçoit un compteur Y tel que

$$Y = (-1)^{\langle \kappa, \boldsymbol{K}\rangle} + B,$$

où B est un bruit qui suit une loi normale de paramètres $\left(0, \frac{1}{4N\varepsilon^2}\right)$.

Tâchons maintenant de voir pourquoi ce modèle est pertinent. On va pour cela utiliser le théorème de Berry-Esséen (théorème A.11) qui est une version quantitative du théorème central limite. Dans notre cas, les variables X_i sont définies comme suit :

$$X_i \stackrel{\text{def}}{=} \langle \pi, \boldsymbol{M}^i\rangle \oplus \langle \gamma, \boldsymbol{C}^i\rangle - \left(\frac{1}{2} + \tilde{\varepsilon}\right),$$

où $\tilde{\varepsilon} \stackrel{\text{def}}{=} (-1)^{\langle \kappa, \boldsymbol{K}\rangle}\varepsilon$. On a alors

$$\begin{aligned} \mathbb{E}\left(X_i\right) &= 0, \\ \sigma^2 = \mathbb{E}\left(X_i^2\right) &= \frac{1}{4} - \varepsilon^2, \\ \rho = \mathbb{E}\left(|X_i|^3\right) &= \frac{1}{8} - 2\,\varepsilon^4. \end{aligned}$$

Les puissances de $\tilde{\varepsilon}$ étant paires, elles ont été remplacées par ε. La variable T correspondant au compteur $T \stackrel{\text{def}}{=} \sum_{i=1}^{N} \langle \pi, \boldsymbol{M}^i\rangle \oplus \langle \gamma, \boldsymbol{C}^i\rangle$ peut donc s'écrire

$$T = \sigma\sqrt{N}\left(\frac{\sum_{i=1}^{N} X_i}{\sigma\sqrt{N}} + \frac{\sqrt{N}}{\sigma}(1/2 + \tilde{\varepsilon})\right).$$

Si $\frac{\sum_{i=1}^{N} X_i}{\sigma\sqrt{N}}$ suit une loi normale de paramètres $(0,1)$ (c'est presque le cas étant donné le théorème central limite), alors T suit une loi normale de paramètres $\left((1/2+\tilde{\varepsilon})\,N, \sigma^2 N\right)$ et $Y \sim \mathcal{N}\left((-1)^{\langle\kappa,\boldsymbol{K}\rangle}, \frac{4\,\sigma^2}{4\,N\varepsilon^2}\right)$. Or, $4\,\sigma^2 = 1 + O(\varepsilon^2)$ donc on peut effectivement dire que Y a une distribution proche de la distribution gaussienne de paramètres $\left((-1)^{\langle\kappa,\boldsymbol{K}\rangle}, \frac{1}{4N\varepsilon^2}\right)$.

7.1.2 Cryptanalyse linéaire multiple de type 1

Intéressons nous maintenant au cas de la cryptanalyse multiple. Dans ce cas on a à notre disposition n approximations

$$\Pr_{\boldsymbol{M},\boldsymbol{K}}\left[\langle\pi_j,\boldsymbol{M}\rangle \oplus \langle\kappa_j,\boldsymbol{K}\rangle = \langle\gamma_j, E_{\boldsymbol{K}}(\boldsymbol{M})\rangle\right] = \frac{1}{2} + \varepsilon_j.$$

On définit donc un vecteur $\boldsymbol{Y} = (Y_j)_{1\leq j\leq n}$ avec

$$Y_j \stackrel{\text{def}}{=} \frac{N - 2\sum_{i=1}^{N}\langle\pi_j,\boldsymbol{m}^i\rangle \oplus \langle\gamma_j, E_{\boldsymbol{k}}\left(\boldsymbol{m}^i\right)\rangle}{2N\varepsilon_j}.$$

Dans ce chapitre ainsi que le suivant on fera l'hypothèse que les approximations linéaires sont statistiquement indépendantes. On discutera de la validité d'une telle hypothèse.

Modèle 7.2. *Cryptanalyse linéaire multiple (type 1).*
L'attaquant reçoit un vecteur de compteurs $\boldsymbol{Y}$ *tel que*

$$Y_j = (-1)^{\tilde{\boldsymbol{K}}_j} + B_j \quad , \quad \tilde{\boldsymbol{K}} \stackrel{\text{def}}{=} \left(\langle\kappa_j,\boldsymbol{K}\rangle\right)_{1\leq j\leq n}$$

où B_j *est un bruit qui suit une loi normale de paramètres* $\left(0, \frac{1}{4N\varepsilon_j^2}\right)$.

De plus, si on note $f(\boldsymbol{Y}|\tilde{\boldsymbol{K}})$ *la densité de la variable* $\boldsymbol{Y}$ *conditionnée par* $\tilde{\boldsymbol{K}}$ *et* $f(Y_j|\tilde{K}_j)$ *la densité de la variable* Y_j *conditionnée par* $\tilde{K}_j$*, alors*

$$f(\boldsymbol{Y}|\tilde{\boldsymbol{K}}) = \prod_{j=1}^{n} f(Y_j|\tilde{K}_j). \tag{7.1}$$

La relation (7.1) est l'expression de l'indépendance supposée des approximations.

Regardons maintenant en quoi on retrouve une problématique de décodage. La figure 7.1 illustre la situation. À la différence du schéma présenté au chapitre 3, on n'effectue pas de démodulation afin d'avoir à notre disposition toute l'information possible. En effet, deux positions à 0.5 et 3 indiquent un bit à 0 car elles sont positives mais l'on a plus confiance en la seconde car elle est plus éloignée de 0 et donc la probabilité d'erreur est plus faible. L'utilisation de cette information lors du décodage est appelée *décodage souple.*

La matrice génératrice utilisée pour encoder l'information est donc celle formée par les masques de clef κ_j. On ne récupère donc pas vraiment la clef complète mais un nombre de bits d égal à la dimension du sous-espace vectoriel engendré par les masques κ_j. Le code en question a donc une dimension $d \leq n_k$ et une longueur n et n'a, a priori, pas de structure particulière. Le problème de décoder un tel code linéaire aléatoire est difficile : il n'existe pas d'algorithme polynômial pour le décoder. De plus, on est dans un cas assez particulier car on va se placer dans une zone de bruit relativement fort. En effet, le bruit est inversement proportionnel à la

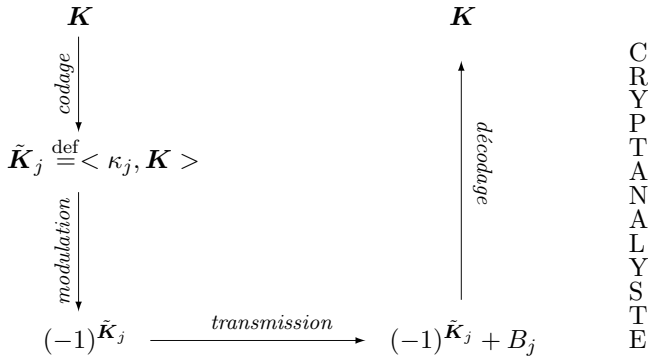

FIGURE 7.1 – Cryptanalyse linéaire multiple de type 1 et décodage.

complexité en données de l'attaque et on a donc intérêt à se placer dans la zone de bruit la plus forte permettant le décodage. De plus, si le nombre de bits intervenant dans les masques de clef est important, on pourra se permettre de renvoyer plusieurs candidats plutôt qu'un seul et donc effectuer un décodage en liste. Enfin, la dimension du code est bornée par le nombre de bits de la clef maître n_k alors que le nombre d'approximations que l'on peut utiliser est énorme et donc on aura tendance à regarder un code avec un très faible rendement (ce qui permettra d'augmenter encore un peu le bruit et donc réduire la complexité en données). Comme l'on a déjà dit, le but est de garder les candidats les plus vraisemblables c'est-à-dire les candidats $\boldsymbol{k}$ qui maximisent $\Pr[\boldsymbol{K}^* = \boldsymbol{k} | \boldsymbol{Y} = \boldsymbol{y}]$. On va voir qu'il existe une formule simple permettant de calculer cette probabilité ou, plus précisément, de calculer une valeur liée à cette probabilité et qui ordonne les candidats de façon identique. Comme on connaît la distribution de $\boldsymbol{y}$ sachant la valeur du bit émis, on utilise la formule de Bayes qui donne $\Pr[A|B] = \frac{\Pr[B|A]\Pr[A]}{\Pr[B]}$. Comme on n'a pas d'information a priori sur la clef, alors $\Pr[\boldsymbol{K}^* = \boldsymbol{k}] = 2^{-d}$ et donc

$$\Pr[\boldsymbol{K}^* = \boldsymbol{k} | \boldsymbol{Y} = \boldsymbol{y}] = 2^{-d} \frac{f(\boldsymbol{Y}|\boldsymbol{K})}{f(\boldsymbol{Y})}.$$

De plus, le modèle dans lequel on se place suppose qu'il y a indépendance entre les approximations. On peut donc décomposer $f(\boldsymbol{Y}|\boldsymbol{K})$ comme un produit de densités pour chaque coefficient de $\boldsymbol{Y}$ en fonction des coefficients de $\tilde{\boldsymbol{K}}$. La fonction logarithme étant croissante, on va en profiter pour passer au logarithme (tout en préservant l'ordre donc) afin d'obtenir une somme plutôt qu'un produit. De plus, la valeur $f(\boldsymbol{Y})$ ne dépend pas du candidat. Les valeurs de $\Pr[\boldsymbol{K}^* = \boldsymbol{k} | \boldsymbol{Y} = \boldsymbol{y}]$ sont donc ordonnées dans le même ordre que les valeurs de $\sum_{j=1}^{n} \ln f_j(Y_j | \tilde{K_j})$. C'est bien mais on peut faire encore plus simple (et donc plus efficace d'un point de vue de l'implémentation). En effet,

$$\begin{aligned}
\ln f_j(Y_j|\tilde{K_j}) &= \ln \left(\frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(y_j - (-1)^{\tilde{k_j}})^2}{2\sigma_j^2}} \right) \\
&= -\frac{(y_j - (-1)^{\tilde{k_j}})^2}{2\sigma_j^2} - \ln \sqrt{2\pi}\sigma_j \\
&= (-1)^{\tilde{k_j}} \frac{y_j}{\sigma_j^2} - \frac{y_j^2 + 1}{2\sigma_j^2} - \ln \sqrt{2\pi}\sigma_j.
\end{aligned}$$

On a donc finalement le critère suivant pour ordonner les candidats par vraisemblance.

Lemme 7.3. *Soit $\boldsymbol{k}$ et $\boldsymbol{k}'$ deux candidats. Alors les équations (7.2) et (7.3) sont équivalentes.*

$$\Pr\left[\boldsymbol{K}^* = \boldsymbol{k} | \boldsymbol{Y} = \boldsymbol{y}\right] > \Pr\left[\boldsymbol{K}^* = \boldsymbol{k}' | \boldsymbol{Y} = \boldsymbol{y}\right] \tag{7.2}$$

$$\sum_{j=1}^{n} (-1)^{\tilde{k}_j} \frac{y_j}{\sigma_j^2} > \sum_{j=1}^{n} (-1)^{\tilde{k}'_j} \frac{y_j}{\sigma_j^2} \tag{7.3}$$

On peut donc normaliser les valeurs y_j une fois pour toute et le calcul de la vraisemblance d'un mot revient juste à $n-1$ additions/soustractions. On présente, dans la section suivante, un algorithme particulièrement adapté au décodage en liste de codes linéaires aléatoires de faible rendement en milieu fortement bruité qui, bien qu'ayant peu d'intérêt pour des applications purement orientées correction d'erreurs, convient parfaitement au cas de la cryptanalyse linéaire.

7.2 Décodage d'un code linéaire de faible rendement en milieu fortement bruité

On cherche donc à décoder, en liste et au maximum de vraisemblance, un code linéaire binaire de faible rendement sur un canal gaussien fortement bruité. La différence avec le cas standard est qu'ici, la puissance du bruit du canal en question dépend de la position. L'algorithme le plus simple est de regarder tous les mots de codes, de calculer leur vraisemblance et de renvoyer les ℓ plus vraisemblables.

Cet algorithme a une probabilité d'erreur de 0 pour un coût de :
- *génération des mots* : 2^d produits matrice vecteur,
- *nombre de mots regardés* : 2^d calculs de vraisemblances.

Le but d'un algorithme de décodage efficace est de permettre de trouver le mot de code le plus vraisemblable sans avoir, justement, à regarder tous les mots de code. L'idée est donc de regarder moins de mots quitte à prendre un faible risque de rater le bon.

Comparaison des complexités. Disons ici un petit mot sur les complexités d'un produit matrice vecteur et d'un calcul de vraisemblance. Le produit matrice vecteur a, dans ce cas, une complexité en $O(n \cdot d)$ et le calcul de vraisemblances une complexité $O(n)$. Cependant, ce sont les calculs de vraisemblances qui vont prendre le plus de temps. En effet, les opérations d'algèbre linéaire s'effectuent sur $\mathbb{F}_2$, on peut donc effectuer autant d'additions ou de multiplications simultanées que le nombre de bits des registres du processeur (32 ou 64 typiquement). De plus, ces opérations correspondent à des opérateurs logiques bit à bit alors que les additions pour le calcul d'une vraisemblance s'effectuent sur des nombres flottants et sont donc plus gourmandes en temps. C'est pourquoi, on va plus chercher à diminuer le nombre de mots regardés que le temps passé à générer ces mots.

7.2.1 Regarder les mots les plus proches

On définit $\tilde{\boldsymbol{y}}$ la quantification du vecteur reçu $\boldsymbol{y}$, $\tilde{\boldsymbol{y}}$ correspond au vecteur reçu dans le cas d'un décodage dur i.e.

$$\tilde{\boldsymbol{y}} = (\tilde{y}_j)_{1 \leq j \leq n}, \quad \tilde{y}_j = \begin{cases} 0 \text{ si } y_j > 0, \\ 1 \text{ sinon} \end{cases}.$$

La première chose qui vient à l'esprit est que, bien que le canal soit très bruité, les mots les plus proches du vecteur $\tilde{\boldsymbol{y}}$ (au sens de la distance de Hamming) seront les plus vraisemblables. Attardons nous donc sur le nombre d'erreurs du vecteur $\tilde{\boldsymbol{Y}}$. Supposons que l'on ait un canal gaussien de puissance σ^2. La probabilité d'erreur sur une position correspondant à un bit 0 du mot de code est la probabilité que $\tilde{Y}_j$ soit négatif et donc que le bruit B_j ait une valeur inférieure à -1. Pour une position 1, cela revient à ce que le bruit soit supérieur à 1 ce qui donne le même résultat vu que pour une loi normale, $\Phi(x) = 1 - \Phi(-x)$.

$$\Pr\left[\tilde{Y}_j \neq \tilde{K}_j\right] = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{-1} e^{-\frac{1}{2}\frac{t^2}{\sigma^2}}\, d\,t.$$

En effectuant un changement de variable, on obtient :

$$\Pr\left[\tilde{Y}_j \neq \tilde{K}_j\right] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-\sigma^{-1}} e^{-\frac{u^2}{2}}\, d\,u = \Phi(-\sigma^{-1}).$$

Le canal étant sans mémoire, le nombre d'erreurs suit donc une loi binomiale de paramètres n et $\Phi(-\sigma^{-1})$. On va donc se fixer un nombre w d'erreurs au-delà duquel on considère qu'il est inutile de chercher. On induit donc une probabilité d'erreur : celle de ne pas regarder le bon candidat. Cette probabilité est la probabilité que le nombre de positions erronées soit strictement supérieur à w et donc

$$P_{\text{err}} = \sum_{i=w+1}^{n} \binom{n}{i} \Phi(-\sigma^{-1})^i \left(1 - \Phi(-\sigma^{-1})\right)^{n-i}.$$

Le problème est qu'ici la puissance du bruit dépend de la position. On a donc une probabilité d'erreur de $\Phi(-\sigma_j^{-1})$ à la position j et donc le nombre d'erreur est une somme de variables de Bernoulli de paramètres $\Phi(-\sigma_j^{-1})$ différents et qui donc n'est pas une loi binomiale. On peut cependant estimer la loi de probabilité du nombre d'erreurs par une loi de Poisson de paramètre $\lambda = \sum_{j=1}^{n} \Phi(-\sigma_j^{-1})$ en utilisant l'inégalité de Le Cam (théorème A.8). Si on applique ce théorème à notre cas, on obtient

$$\left| P_{\text{err}} - \sum_{j=w+1}^{n} \frac{\lambda^j e^{-\lambda}}{j!} \right| \leq \sum_{j=1}^{n} \Phi(-\sigma_j^{-1})^2.$$

Il est alors raisonnable de prendre $w \approx \lambda$.

On a alors les complexités suivantes :

- *syndromes* : $\sum_{i=0}^{w} \binom{n}{i}$ produits matrice vecteur,
- *vraisemblance* : $2^{d-(1-H(w/n))n}$ calculs de vraisemblances.

On va donc regarder moins de mots de code mais par contre le temps pour les générer sera plus élevé. En effet, il faudra, en moyenne, calculer le syndrome de 2^{n-k} motifs d'erreurs avant d'obtenir un motif menant à un mot de code (c'est-à-dire ayant le même syndrome que $\tilde{\boldsymbol{y}}$).

7.2.2 Utilisation de codes poinçonnés

Avec la méthode précédente, on a un paramètre w qui permet de diminuer le nombre de mots regardés au prix d'un plus grand nombre de calcul de syndromes. Le problème est que w permet juste d'effectuer le compromis entre probabilité d'erreur et complexité totale. On

n'a aucun moyen de modifier le nombre de syndromes à générer pour obtenir un mot de code qui est fixé à 2^{n-d} par les paramètres du code (et donc de l'attaque). Or, comme on est en rendement faible, $n \gg d$ et 2^{n-d} calculs de syndromes auront un coût bien plus important que le calcul de la vraisemblance du mot de code généré. Le seul moyen pour arranger cela est diminuer n la longueur du code et donc de poinçonner le code comme expliqué dans la sous-section 3.2.2. Cela réduira le nombre d'équations de parité et donc diminuera le nombre de syndromes à calculer pour obtenir un mot de code. Notons h le nombre de bits de redondance conservés. On va donc travailler sur un code poinçonné de longueur $d + h$. On choisira donc h de façon à ce que le coup de calcul de 2^h syndromes soit équivalent au coût du calcul d'une vraisemblance. On pourra ensuite regarder le compromis temps d'exécution/probabilité d'erreur en ajustant la valeur de w.

7.2.3 Utilisation de l'information souple

Une question importante que l'on n'a pas encore évoquée est le choix des positions à poinçonner. Étant donné que l'on a accès à de l'information souple, il paraît évident qu'il faut en profiter. Sachant que le signal a été transmis via un canal gaussien, on peut mesurer la probabilité d'erreur de transmission connaissant Y. Si Y est positif alors elle vaut $\Phi(-Y)$ et si Y est négatif, elle vaut $1 - \Phi(-Y) = \Phi(Y)$. On voit bien que dans les deux cas, on a une probabilité que la position soit erronée qui vaut $\Phi(-|Y|)$. On en conclut donc que $|Y|$ est une bonne mesure de la fiabilité d'une position.

Définition 7.4. On appelle *fiabilité* de la j-ième position la valeur $|Y_j|$.

L'idée est donc de trier les positions par fiabilité et de regarder les $d + h$ meilleures en espérant qu'elles contiennent un ensemble d'information. L'utilisation de cette information va donc faire diminuer la probabilité d'erreur du décodage sans faire augmenter la complexité de l'algorithme (le coût du tri étant négligeable devant le reste). Cependant, le calcul de cette nouvelle probabilité d'erreur va être encore un peu plus complexe.

7.2.4 Paradoxe des anniversaires

Afin d'accélérer encore plus la phase de génération des motifs d'erreur, on peut utiliser un algorithme qui utilise le *paradoxe des anniversaires*. L'idée est assez simple et va considérablement réduire la complexité de génération des motifs d'erreur mais en contrepartie on ne regardera pas non plus certains mots pourtant intéressants. Le principe est de séparer le support des $d + h$ positions regardées en deux et on va supposer que les w erreurs sont équitablement réparties entre les deux moitiés (on suppose pour le moment que w est pair). On perd alors un certain nombre de motifs d'erreurs : ceux qui ont plus de $w/2$ erreurs sur une des deux moitiés. Le gain en temps d'exécution vient du fait que l'on va utiliser un compromis temps/mémoire afin de générer les motifs d'erreur ayant le syndrome voulu. Les choses commençant à devenir plus compliquées, il est grand temps de fixer quelques notations de façon à clarifier la suite.

Notations. Le code linéaire binaire considéré est noté $\mathcal{C}$. Soit G une de ses matrices génératrices et H une de ses matrices de parité. On note $\boldsymbol{y} \in \mathbb{R}^n$ le vecteur reçu et $\tilde{\boldsymbol{y}} \in \mathbb{F}_2^n$ sa quantification. Le support du poinçonnage I est partitionné en deux ensembles de même cardinalité I_1 et I_2. On définit, pour un ensemble de positions A, les matrices G_A et H_A qui

sont les matrices formées des colonnes dont les indices appartiennent à A. Il en va de même pour les vecteurs $\boldsymbol{y}$ et $\tilde{\boldsymbol{y}}$. On pourra alors parler de $S_I(\cdot)$, syndrome d'un vecteur relativement à la matrice de parité H_I ainsi que de $S_1(\cdot)$ et $S_2(\cdot)$. Bien sûr, $S_I(\cdot) = S_{I_1}(\cdot) \oplus S_{I_2}(\cdot)$. Pour finir, on note $s \stackrel{\text{def}}{=} S_I(\tilde{\boldsymbol{y}}_I)$ Nous pouvons maintenant revenir à notre algorithme et expliquer comment générer les motifs d'erreurs ayant le syndrome s voulu. On génère deux tableaux dans lesquels on stocke les demi motifs d'erreurs en fonction de leur syndrome. Il suffit ensuite de regarder les cases dont le XOR des indices donne $\boldsymbol{s}$ et former les couples de motifs. L'algorithme ne nécessitant pas que w soit pair ou que les deux ensembles I_1 et I_2 soient de même cardinalité, il sera présenté de façon générale. On note alors w_1 et w_2 les poids autorisés sur I_1 et I_2 ainsi que h_1 et h_2 la cardinalité de ces deux ensembles.

Algorithme 13 : Génération de motifs d'erreurs ayant un syndrome s.

Entrée : $\boldsymbol{s} \in \mathbb{F}_2^h$ syndrome souhaité;
Sortie : $\mathcal{M}$ liste de motifs ayant $\boldsymbol{s}$ pour syndrome;
Préparer deux tables vides T_1 et T_2 de taille 2^h;
$\mathcal{M} \longleftarrow \emptyset$;
pour tous les $\boldsymbol{m}_1 \in \mathbb{F}_2^{h_1}$ *tel que* $\mathrm{w}_H(\boldsymbol{m}_1) \leq w_1$*;*
faire
  $T_1[S_1(\boldsymbol{m}_1)] \longleftarrow T_1[S_1(\boldsymbol{m}_1)] \cup \{\boldsymbol{m}_1\}$;
fin
pour tous les $\boldsymbol{m}_2 \in \mathbb{F}_2^{h_2}$ *tel que* $\mathrm{w}_H(\boldsymbol{m}_2) \leq w_2$*;*
faire
  $T_2[S_2(\boldsymbol{m}_2)] \longleftarrow T_2[S_2(\boldsymbol{m}_2)] \cup \{\boldsymbol{m}_2\}$;
fin
pour $\boldsymbol{i} \in \mathbb{F}_2^h$ **faire**
  pour tous les $(\boldsymbol{m}_1, \boldsymbol{m}_2) \in T_1[\boldsymbol{s} \oplus \boldsymbol{i}] \times T_2[i]$ **faire**
    $\mathcal{M} \longleftarrow \mathcal{M} \cup \{(\boldsymbol{m}_1, \boldsymbol{m}_2)\}$;
  fin
fin
retourner $\mathcal{M}$*;*

7.2.5 Résonance stochastique

Cette idée est assez originale vu qu'elle consiste à ajouter du bruit. En effet, une fois un tour effectué avec les $d+h$ positions les plus fiables, on ajoute un peu de bruit gaussien (mais moins puissant que celui du canal) au vecteur $\boldsymbol{y}$ (on note $\boldsymbol{y}'$ le nouveau vecteur) et on choisit les positions les plus fiables de $\boldsymbol{y}'$. Le but était de faire varier les ensembles en essayant de garder une bonne proportion de positions fiables. Il fallait donc que des positions très fiables puissent être retirées mais pas trop souvent. Utiliser cette technique permet de renouveler suffisamment les positions tout en ayant une bonne proportion de positions fiables et semble donc plutôt bien adaptée.

Voilà donc l'algorithme de décodage par résonance stochastique tel que proposé dans [Val00a, Val00b]. On notera $v(\cdot)$ la fonction qui calcule « la vraisemblance » d'un mot par rapport à $\boldsymbol{y}$. En fait $v(\cdot)$ n'est pas la vraisemblance du mot mais la fonction v conserve l'ordre relatif à cette vraisemblance (comme prouvé section 7.1) $v(\boldsymbol{c}) \stackrel{\text{def}}{=} \sum_{j=1}^{n} (-1)^{\boldsymbol{c}_j} \frac{\boldsymbol{y}_j}{\sigma_j^2}$.

Algorithme 14 : Algorithme de décodage par résonance stochastique.

Entrée : $\boldsymbol{y}$ vecteur reçu;
Sortie : $\mathcal{L}$ liste de ℓ mots de code (les plus vraisemblables parcourus);
$\mathcal{L} \longleftarrow \emptyset$;
pour $1 \leq j \leq t$ **faire**
  si $j \neq 1$ **alors**
    Ajouter un bruit gaussien aux $\boldsymbol{y}_i$;
  fin
  Trier les $\boldsymbol{y}'_i$ selon leur fiabilité ($|\boldsymbol{y}'_i|$);
  $I \longleftarrow d + h$ positions les plus fiables ;
  Quantifier $\boldsymbol{y}_I$ en $\tilde{\boldsymbol{y}}_I$;
  $\boldsymbol{s} \longleftarrow S_I(\tilde{\boldsymbol{y}}_I)$;
  pour tous les *motifs* $\boldsymbol{m}$ *obtenus avec l'algorithme 13* **faire**
    $\boldsymbol{c}_I \longleftarrow \tilde{\boldsymbol{y}}_I \oplus \boldsymbol{m}$;
    Réencoder $\boldsymbol{c}_I$ en $\boldsymbol{c}$;
    si $\#\mathcal{L} < \ell$ *ou* $\exists\, \boldsymbol{c}' \in \mathcal{L}, v(\boldsymbol{c}') < v(\boldsymbol{c})$ **alors**
      $\mathcal{L} \longleftarrow \mathcal{L} \cup \{\boldsymbol{c}\} \backslash \{\boldsymbol{c}'\}$;
    fin
  fin
fin
retourner $\mathcal{L}$;

7.2.6 Quelques résultats

Les résultats que nous allons présenter ici ne sont pas publiés. Ils ont été obtenus dans le prolongement de ceux qui figurent dans [Gér07]. Il s'agit de compter le nombre de candidats regardés lors d'une cryptanalyse linéaire multiple utilisant l'algorithme 14.

Des tests on été effectués sur un des groupes d'approximations utilisées dans le chapitre 8. Ce groupe contient 12384 approximations linéaires qui forment un code de dimension 19.

On utilise deux types d'algorithmes pour générer les motifs d'erreur lors de cette attaque. Le premier est l'algorithme 13 et le second est une version légèrement différente. En effet, comme on va se trouver dans des zones de bruit très élevé et que l'on a des restrictions fortes sur la complexité en temps de l'attaque (il faut pouvoir l'effectuer un bon nombre de fois), on va utiliser une variante qui scinde le support en 4 morceaux sur lesquels on va autoriser au plus une erreur. À performances égales en terme de mots regardés et probabilité de succès, cet algorithme est plus rapide que le premier dès que le bruit dépasse une certaine borne.

Les tableaux 7.1 et 7.2 contiennent les résultats obtenus pour $N = 2^{46}$ et $N = 2^{43}$. La valeur 2^{46} correspond à la quantité de messages nécessaires pour que la bonne clef se retrouve avec probabilité proche de 1 au début de la liste. La valeur de h donnée est, on le rappelle, le paramètre qui donne le nombre de positions de redondance du code poinçonné. L'expression $w(a/b|c/d)$ signifie que l'on a découpé le support du poinçonnage en deux parties de taille respectives b et d, que l'on autorise a erreurs sur la première partie et c sur la seconde. On utilise une notation similaire pour un support découpé en quatre.

Pour chacune des deux zones de bruit, on donne les deux meilleurs jeux de paramètres obtenus lors des expériences. On donne la probabilité de succès de l'algorithme de décodage par rapport au nombre de mots regardés et de la taille ℓ de la liste des candidats gardés. La

Paramètres : $h = 7$ et $w = (3/15, 3/11)$.

P_S	2^{10}	2^{12}	2^{14}	2^{16}	2^{17}	2^{18}	2^{19}
$\ell = 2^0$	0.020	0.077	0.226	0.548	0.714	0.828	0.883
$\ell = 2^6$	0.020	0.078	0.230	0.576	0.775	0.922	0.994
$\ell = 2^{10}$	0.020	0.078	0.230	0.577	0.776	0.926	1.000

Paramètres : $h = 9$ et $w = (2/7, 2/7, 2/7, 2/7)$.

P_S	2^{10}	2^{12}	2^{14}	2^{16}	2^{17}	2^{18}	2^{19}
$\ell = 2^0$	0.018	0.052	0.211	0.515	0.695	0.821	0.869
$\ell = 2^6$	0.018	0.052	0.219	0.547	0.753	0.917	0.994
$\ell = 2^{10}$	0.018	0.052	0.219	0.547	0.755	0.921	1.000

TABLE 7.1 – Performances de l'algorithme 14 pour $N = 2^{46}$.

Paramètres : $h = 7$ et $w = (3/14, 3/12)$.

P_S	2^{10}	2^{12}	2^{14}	2^{16}	2^{17}	2^{18}	2^{19}
$\ell = 2^2$	0.000	0.003	0.002	0.010	0.016	0.017	0.021
$\ell = 2^8$	0.004	0.014	0.037	0.083	0.103	0.129	0.144
$\ell = 2^{14}$	-	-	0.075	0.261	0.382	0.521	0.634
$\ell = 2^{18}$	-	-	-	-	-	0.706	0.990

Paramètres : $h = 9$ et $w = (2/7, 2/7, 3/7, 3/7)$.

P_S	2^{10}	2^{12}	2^{14}	2^{16}	2^{17}	2^{18}	2^{19}
$\ell = 2^2$	0.001	0.001	0.001	0.007	0.012	0.017	0.018
$\ell = 2^8$	0.003	0.011	0.028	0.066	0.097	0.127	0.148
$\ell = 2^{14}$	-	-	0.058	0.207	0.356	0.503	0.630
$\ell = 2^{18}$	-	-	-	-	-	0.680	0.991

TABLE 7.2 – Performances de l'algorithme 14 pour $N = 2^{43}$.

dernière colonne correspond à un décodage complet *i.e.* au calcul des vraisemblances des 2^{19} mots.

On voit que le découpage en deux ou en quatre donnent des résultats similaires au niveau de la probabilité de succès de l'algorithme. Cependant, l'algorithme utilisé avec un découpage en quatre est bien plus rapide du fait de sa plus grande efficacité dans la génération des motifs d'erreur.

Chapitre 8

Entropie et estimation de la complexité en données : application à la cryptanalyse linéaire

Sommaire

On a vu dans le chapitre précédent que l'on pouvait modéliser une cryptanalyse linéaire multiple de type 1 par un problème de décodage sur un canal gaussien. On a ensuite utilisé des outils de théorie des codes correcteurs, ici un algorithme de décodage de codes binaires aléatoires, pour traiter le sujet.

De la même façon, on va utiliser, dans ce chapitre, une approche codage au problème de la quantification de la complexité en données d'une attaque. Ce chapitre traite essentiellement du travail présenté dans [GT09]. Comme on l'a dit dans la sous-section 4.2.4, l'estimation donnée dans [BDCQ04] pour la complexité en données est pessimiste du fait que ce que regardent les auteurs est l'espérance du rang de la bonne clef. Nous proposons dans la section 8.1 une autre approche au problème qui permet d'estimer la quantité d'échantillons nécessaires pour une probabilité de succès raisonnable. On présentera, ensuite, dans la section 8.2, une cryptanalyse linéaire multiple de type 1 du DES utilisant 32968 approximations. On a ainsi pu comparer la formule donnée section 8.1 aux résultats expérimentaux.

8.1 Entropie et complexité en données

Comme expliqué en introduction, il s'agit ici de reprendre le modèle 7.2 et d'étudier la cryptanalyse multiple d'un point de vue théorie de l'information. Le problème, rappelons-le, est que le rang de la bonne clef parmi les candidats est une variable qui peut prendre de très grandes valeurs dans des cas extrêmement rares. Ceci influence l'espérance de cette variable bien que ces cas arrivent peu souvent. Pour éviter cet écueil, on va utiliser une grandeur bien moins sensible à ce genre de comportements : l'entropie. Celle-ci est utilisée en théorie des codes correcteurs où ce genre de phénomènes n'est pas rare dans certaines zones de bruit. L'entropie d'une variable X est une grandeur moins sensible à de grands écarts rares que l'espérance de cette variable. On verra aussi que l'information mutuelle entre la clef et les compteurs est une grandeur qui se comporte de façon similaire à *l'avantage* d'une cryptanalyse statistique.

On va, dans un premier temps, présenter la borne générale qui peut être obtenue sur N (sous-section 8.1.1). On appliquera ensuite, dans la sous-section 8.1.2, cette borne aux trois types d'attaque multiple afin d'en donner une formule un peu plus explicite dans chacun de ces cas. On terminera enfin, dans la sous-section 8.1.3, par une discussion sur les résultats obtenus. On commencera par comparer la formule obtenue pour la complexité en données à celle donnée dans [BDCQ04]. Cette dernière est basée sur une approximation de l'espérance du rang de la bonne clef et donc, on le verra, fournit un résultat pessimiste. On effectuera aussi un parallèle avec la remarque de Pascal Junod dans [Jun01] à propos du pessimisme présumé de son théorème.

8.1.1 Borne générale sur l'entropie et la complexité en données

Lors d'une cryptanalyse statistique, on cherche à retrouver certains bits de clef ou certaines fonctions de bits de clef à partir d'échantillons de données chiffrées avec cette même clef $\boldsymbol{K}$. Dans le cas d'une cryptanalyse multiple, pour chaque caractéristique utilisée, on récupère une statistique Y_j. Celle-ci va apporter de l'information sur une partie de $\boldsymbol{K}$ que l'on note $\boldsymbol{K'}$. La borne que l'on propose repose sur l'hypothèse d'indépendance de ces caractéristiques statistiques (approximations linéaires dans le cadre de la cryptanalyse linéaire). Voici le modèle dans lequel on se place, celui-ci est donné de façon très générale car cette borne peut être utilisée pour n'importe quelle cryptanalyse statistique.

Modèle 8.1.

1. *La variable* $\boldsymbol{K'} = (K'_j)_{1\leq j\leq n}$ *peut prendre* $2^{k'}$ *valeurs. Cette variable est distribuée uniformément sur ces* $2^{k'}$ *valeurs.*
2. *On suppose que les variables* $\boldsymbol{K'}$ *et* $\boldsymbol{Y} = (Y_j)_{1\leq j\leq n}$ *vérifient la condition suivante.*

$$f(\boldsymbol{Y}|\boldsymbol{K'}) = \prod_{j=1}^{n} f(Y_j|K'_j), \tag{8.1}$$

où $f(A)$ *désigne la densité de la variable aléatoire* A*.*

La borne que l'on va maintenant dériver repose essentiellement sur l'égalité (8.1) qui découle directement de l'indépendance statistique supposée des caractéristiques [1].

Lemme 8.2. *[CT91]*

$$I\left(\boldsymbol{K}';\boldsymbol{Y}\right) \leq \sum_{j=1}^{n} I\left(K'_j;Y_j\right) \tag{8.2}$$

$$H\left(\boldsymbol{K}'|\boldsymbol{Y}\right) \geq k' - \sum_{j=1}^{n} I\left(K'_j;Y_j\right). \tag{8.3}$$

Avant de voir la preuve de ce lemme, regardons quelle information celle-ci apporte sur la complexité en données. Avoir $H\left(\boldsymbol{K}'|\boldsymbol{Y}\right) = 0$ signifie avoir une connaissance totale de la variable $\boldsymbol{K}'$ en regardant $\boldsymbol{Y}$. Dans le cas du *top ranking*, pour avoir une cryptanalyse avec une probabilité de succès proche de 1, il faut donc une entropie $H\left(\boldsymbol{K}'|\boldsymbol{Y}\right)$ proche de 0. Le fait d'avoir une borne inférieure sur l'entropie nous donne donc une borne inférieure sur la complexité en données N. En effet, il faut au moins que N soit tel que $\sum_{j=1}^{n} I\left(K'_j;Y_j\right) \geq k'$ si l'on veut pouvoir espérer avoir l'entropie $H\left(\boldsymbol{K}'|\boldsymbol{Y}\right)$ proche de 0 et donc une cryptanalyse avec probabilité de succès proche de 1.

On donnera dans la sous-section 8.1.3 une preuve que cette borne est précise dans le cas particulier de la cryptanalyse linéaire multiple de type 1 qui est l'attaque qui a été implémentée et qui est présentée dans la section 8.2. Pour le moment, revenons la preuve du lemme 8.2.

Démonstration : Afin de prouver ce lemme, rappelons d'abord que, par définition, $H\left(\boldsymbol{K}'|\boldsymbol{Y}\right) = H\left(\boldsymbol{K}'\right) - I\left(\boldsymbol{K}';\boldsymbol{Y}\right)$. Le premier point du modèle 8.1 implique que $H\left(\boldsymbol{K}'\right) = k'$. Reste donc à majorer l'information $I\left(\boldsymbol{K}';\boldsymbol{Y}\right)$ sur la clef contenue dans le vecteur $\boldsymbol{Y}$ extrait des échantillons. On utilise pour cela le fait que l'information mutuelle est symétrique et on obtient

$$I\left(\boldsymbol{K}';\boldsymbol{Y}\right) = I\left(\boldsymbol{Y};\boldsymbol{K}'\right) = H\left(\boldsymbol{Y}\right) - H\left(\boldsymbol{Y}|\boldsymbol{K}'\right).$$

On utilise le corollaire 3.6 pour décomposer chacun des deux termes selon les composantes des vecteurs considérés.

$$H\left(\boldsymbol{Y}\right) = \sum_{j=1}^{n} H\left(Y_j|Y_{j-1}\dots Y_1\right), \tag{8.4}$$

$$H\left(\boldsymbol{Y}|\boldsymbol{K}'\right) = \sum_{j=1}^{n} H\left(Y_j|Y_{j-1}\dots Y_1,\boldsymbol{K}'\right). \tag{8.5}$$

On va voir que l'égalité (8.1) du modèle utilisé nous permet de simplifier chacun des termes de la somme de (8.5) en remarquant que l'on peut écrire l'entropie conditionnelle comme

$$\begin{aligned} H\left(Y_j|Y_{j-1}\dots Y_1,\boldsymbol{K}'\right) &= \sum_{\boldsymbol{k}'} \int_{\mathbb{R}^{j-1}} H\left(Y_j|Y_{j-1}=y_{j-1}\dots Y_1=y_1,\boldsymbol{K}'=\boldsymbol{k}'\right) \\ &\times f(y_1,\dots,y_{j-1}|\boldsymbol{K}'=\boldsymbol{k}')\Pr\left[\boldsymbol{K}'=\boldsymbol{k}'\right] d\,y_1 \ \dots \ d\,y_{j-1}, \end{aligned}$$

1. On a discuté de cette hypothèse pour le cas particulier de la cryptanalyse linéaire dans la sous-section 4.4.1.

avec la notation évidente qui consiste à noter $f(y_1,\dots,y_{j-1}|\boldsymbol{K}'=\boldsymbol{k}')$ la densité de probabilité conditionnelle du vecteur $(Y_1,\dots,Y_{j-1})$ pour une valeur $\boldsymbol{k}'$ de la variable $\boldsymbol{K}'$ et au point $(y_1,\dots,y_{j-1})$.

On va maintenant pouvoir appliquer l'égalité (8.1) à $H\left(Y_j|Y_{j-1}=y_{j-1}\dots Y_1=y_1,\boldsymbol{K}'=\boldsymbol{k}'\right)$ et obtenir

$$H\left(Y_j|Y_{j-1}=y_{j-1}\dots Y_1=y_1,\boldsymbol{K}'=\boldsymbol{k}'\right)=H\left(Y_j|K'_j=k'_j\right).$$

On décompose alors la somme sur $\boldsymbol{k}'$ en une somme sur k'_j et une somme sur les $k'_{j'<j}$.

$$\begin{aligned}H\left(Y_j|Y_{j-1}\dots Y_1,\boldsymbol{K}'\right) &= \sum_{k'_j}H\left(Y_j|K'_j=k'_j\right)\sum_{k'_1,\dots,k'_{j-1}}\int_{\mathbb{R}^{j-1}}\Pr\left[\boldsymbol{K}'=\boldsymbol{k}'\right]\\ &\times\ f(y_1,\dots,y_{j-1}|\boldsymbol{K}'=\boldsymbol{k}')d\,y_1\ \dots\ d\,y_{j-1},\end{aligned}$$

La somme combinée à l'intégrale donne $\Pr\left[K'_j=k'_j\right]$ et l'on obtient finalement

$$\begin{aligned}H\left(Y_j|Y_{j-1}\dots Y_1,\boldsymbol{K}'\right) &= \sum_{k'_j}H\left(Y_j|K'_j=k'_j\right)\Pr\left[K'_j=k'_j\right]\\ &= H\left(Y_j|K'_j\right).\end{aligned}$$

On revient à l'équation (8.5) qui devient alors

$$H\left(\boldsymbol{Y}|\boldsymbol{K}'\right)=\sum_{j=1}^{n}H\left(Y_j|K'_j\right).$$

En ce qui concerne (8.4), comme la connaissance d'une variable ne peut que diminuer l'entropie, on peut borner $H(\boldsymbol{Y})$ en bornant chaque terme de la somme par $H(Y_j)$. On peut trouver ce résultat dans [CT91] sous le nom de *borne d'indépendance sur l'entropie*. Ce nom provient du fait que cette borne est atteinte dans le cas de variables indépendantes.

$$H(\boldsymbol{Y})\leq\sum_{j=1}^{n}H(Y_j). \tag{8.6}$$

On obtient donc

$$\begin{aligned}I\left(\boldsymbol{K}';\boldsymbol{Y}\right) &\leq \sum_{j=1}^{n}H(Y_j)-\sum_{j=1}^{n}H\left(Y_j|K'_j\right)\\ &\leq \sum_{j=1}^{n}H(Y_j)-H\left(Y_j|K'_j\right)\\ &\leq I\left(K'_j;Y_j\right).\end{aligned}$$

Ce qui donne le résultat souhaité vu que l'on a commencé cette preuve par l'égalité $H\left(\boldsymbol{K}'|\boldsymbol{Y}\right)=k'-I\left(\boldsymbol{K}';\boldsymbol{Y}\right)$. □

8.1.2 Applications aux trois types de cryptanalyse multiple

On a vu une borne générale sur l'entropie de la clef connaissant les statistiques extraites de N échantillons. Pour pouvoir calculer le N nécessaire à une attaque statistique, il faut donc avoir une formule pour $I\left(K'_j; Y_j\right)$. On propose dans cette sous-section de donner les formules de $I\left(K'_j; Y_j\right)$ correspondant aux trois types de cryptanalyse linéaire vues dans le chapitre 4. On rappelle les modèles vus dans le chapitre 4 et le chapitre 7. Pour ce qui est de la clef interne, on rappelle le modèle 7.2.

Modèle 7.2
L'attaquant reçoit des compteurs $\tilde{Y}_j$ tels que

$$\tilde{Y}_j = (-1)^{<\kappa_j, \tilde{\boldsymbol{K}}_j>} + B_j,$$

où B_j est un bruit qui suit une loi normale de paramètres $\left(0, \sigma_j^2\right)$, $\sigma_j^2 \stackrel{\text{def}}{=} \frac{1}{4N\varepsilon_j^2}$. De plus,

$$f(\tilde{\boldsymbol{Y}}|\tilde{\boldsymbol{K}}) = \prod_{j=1}^{n} f(\tilde{Y}_j|\tilde{K}_j).$$

Si on effectue une attaque sur le dernier tour, on obtient autant de vecteurs de n compteurs qu'il existe de candidats pour la clef externe $\boldsymbol{K}_{\mathrm{O}}$. On les note $\hat{\boldsymbol{Y}}^{\boldsymbol{k}_{\mathrm{O}}}$ et alors, l'hypothèse de répartition aléatoire par fausse clef hypothèse 4.3 nous dit que, si l'on note $\boldsymbol{k}^*_{\mathrm{O}}$ la bonne clef externe,

$$\hat{Y}_j^{\boldsymbol{k}_{\mathrm{O}}} = \begin{cases} \tilde{Y}_j \text{ si } \boldsymbol{k}_{\mathrm{O}} = \boldsymbol{k}^*_{\mathrm{O}}, \\ B_j^{\boldsymbol{k}_{\mathrm{O}}} \text{ sinon.} \end{cases},$$

où $B_j^{\boldsymbol{k}_{\mathrm{O}}}$ est un bruit gaussien suivant une loi normale $\mathcal{N}\left(0, \sigma_j^2\right)$.

Cryptanalyse de type 1.
Dans le cas de la cryptanalyse de type 1, on a $\boldsymbol{K}' = \tilde{\boldsymbol{K}}$ et $\boldsymbol{Y} = \tilde{\boldsymbol{Y}}$. On a donc $H\left(\boldsymbol{K}'\right) = d$ avec d la dimension de l'espace vectoriel engendré par les κ_j. Comme l'on a vu dans le chapitre 7, on peut voir Y_j comme le signal reçu en ayant transmis K'_j sur un canal à bruit blanc additif gaussien de puissance σ_j^2. On cherche à calculer $I\left(K'_j; Y_j\right)$. Or, si on reprend ce qui a été vu dans la sous-section 3.1.3, on sait que la capacité du canal gaussien de puissance σ_j^2 est définie par

$$\mathrm{Cap}\left(\sigma_j^2\right) \stackrel{\text{def}}{=} \max_{p(K'_j)} I\left(K'_j; Y_j\right),$$

où le maximum est pris sur l'ensemble des distributions possibles pour la variable K'_j. De plus, le théorème 3.14 nous dit que le maximum est atteint pour la distribution uniforme, or, la distribution de K'_j est a priori uniforme puisqu'avant l'attaque, on n'a pas d'information sur la clef. On en déduit que

$$I\left(K'_j; Y_j\right) = \mathrm{Cap}\left(\sigma_j^2\right) = 1 - \frac{\sigma_j}{\sqrt{8\pi}} \int_{-\infty}^{\infty} e^{-\frac{(\sigma_j^2 t - 2)^2}{8\sigma_j^2}} \log_2(1 + e^{-t})\, dt.$$

Cryptanalyse de type 2.

Dans le cas de la cryptanalyse linéaire de type 2, on ne s'intéresse pas à la clef interne. Les variables $\boldsymbol{K}'_j$ correspondent alors aux bits de clef externe nécessaires au déchiffrement partiel des chiffrés pour évaluer la j-ème approximation linéaire. On suppose que ces bits sont les mêmes pour chaque approximation et on note $\hat{\boldsymbol{K}}$ cette valeur : pour tout j, $\boldsymbol{K}'_j = \hat{\boldsymbol{K}}$. Comme on ne se soucie pas de la clef interne, on regarde le vecteur $\boldsymbol{Y}$ défini par

$$Y_j^{\boldsymbol{k}_O} \stackrel{\text{def}}{=} |\hat{Y}_j^{\boldsymbol{k}_O}|.$$

On note $\hat{k}$ le nombre de bits nécessaires au déchiffrement partiel d'un couple pour toutes les approximations. On a donc $H\left(\boldsymbol{K}'\right) = \hat{k}$.

On va ici avoir besoin d'utiliser à nouveau l'inégalité (8.2) afin de décomposer l'information mutuelle $I\left(\boldsymbol{K}'_j; \boldsymbol{Y}_j\right)$ selon les clefs externes utilisées

$$I\left(\boldsymbol{K}'_j; \boldsymbol{Y}_j\right) \leq \sum_{\boldsymbol{k}_O} I\left(\boldsymbol{K}'_j; Y_j^{\boldsymbol{k}_O}\right). \tag{8.7}$$

On utilise ensuite l'égalité $I\left(\boldsymbol{K}'_j; Y_j^{\boldsymbol{k}_O}\right) = H\left(Y_j^{\boldsymbol{k}_O}\right) - H\left(Y_j^{\boldsymbol{k}_O} | \boldsymbol{K}'_j\right)$. On connaît seulement la distribution de $Y_j^{\boldsymbol{k}_O}$ sachant si $\boldsymbol{K}'_j$ est ou non le bon candidat. On décomposera donc la densité de $Y_j^{\boldsymbol{k}_O}$ en une somme de deux densités selon que k_O est ou non le bon candidat. De même, on décompose la seconde entropie en

$$H\left(Y_j^{\boldsymbol{k}_O} | \boldsymbol{K}'_j\right) = \sum_{k'_r} H\left(Y_j^{\boldsymbol{k}_O} | \boldsymbol{K}'_j = \boldsymbol{k}'_r\right) \Pr\left[\boldsymbol{K}'_j = \boldsymbol{k}'_r\right].$$

Cette somme contient le terme pour lequel $\boldsymbol{k}'_r = \boldsymbol{k}_O$ et les $2^{\hat{k}} - 1$ termes correspondant aux candidats $\boldsymbol{k}'_r \neq \boldsymbol{k}_O$. La distribution des $Y_j^{\boldsymbol{k}_O}$ étant la même quelle que soit la valeur de $\boldsymbol{K}'_j$ différente de k_O, on obtient :

$$H\left(Y_j^{\boldsymbol{k}_O} | \boldsymbol{K}'_j\right) = 2^{-\hat{k}} H\left(Y_j^{\boldsymbol{k}_O} | \boldsymbol{K}'_j = \boldsymbol{k}_O\right) + (1 - 2^{-\hat{k}}) H\left(Y_j^{\boldsymbol{k}_O} | \boldsymbol{K}'_j \neq \boldsymbol{k}_O\right).$$

Les $Y_j^{\boldsymbol{k}_O}$ sont des valeurs absolues de lois normales. On définit la notation suivante afin d'obtenir des formules plus claires pour les densités en jeu.

$$\varphi_j^{\alpha}(t) \stackrel{\text{def}}{=} \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(t-\alpha)^2}{2\sigma_j^2}}.$$

Ainsi, la densité $f_j^0(y)$ de $Y_j^{\boldsymbol{k}_O}$ sachant $\boldsymbol{K}'_j \neq \boldsymbol{k}_O$ est $f_j^0(y) = 2\varphi_j^0(y)$. La densité $f_j^1(y)$ de $Y_j^{\boldsymbol{k}_O}$ sachant $\boldsymbol{K}'_j = \boldsymbol{k}_O$ est $f_j^1(y) = \varphi_j^{-1}(y) + \varphi_j^1(y)$. Et la densité $f_j(y)$ de $Y_j^{\boldsymbol{k}_O}$ est

$$f_j(y) = 2^{-\hat{k}} f_j^1(y) + (1 - 2^{-\hat{k}}) f_j^0(y). \tag{8.8}$$

On obtient donc finalement

$$\begin{aligned} I\left(K'_j; Y_j^{\boldsymbol{k}_O}\right) &= H\left(Y_j^{\boldsymbol{k}_O}\right) - H\left(Y_j^{\boldsymbol{k}_O} | K'_j\right) \\ &= -\int_{\mathbb{R}^+} (2^{-\hat{k}} f_j^1(y) + (1-2^{-\hat{k}}) f_j^0(y)) \log_2 f_j(y)\, dy \\ &+ 2^{-\hat{k}} \int_{\mathbb{R}^+} f_j^1(y) \log_2 f_j^1(y)\, dy \\ &+ (1-2^{-\hat{k}}) \int_{\mathbb{R}^+} f_j^0(y) \log_2 f_j^0(y)\, dy \\ &= 2^{-\hat{k}} \int_{\mathbb{R}^+} f_j^1(y) \log_2 \left(\frac{f_j^1(y)}{f_j(y)}\right) dy \\ &+ (1-2^{-\hat{k}}) \int_{\mathbb{R}^+} f_j^0(y) \log_2 \left(\frac{f_j^0(y)}{f_j(y)}\right) dy \end{aligned}$$

Si on injecte cela dans l'équation (8.7), on obtient

$$I\left(\boldsymbol{K}'_j; \boldsymbol{Y}_j\right) \leq 2^{\hat{k}} I_j,$$

avec

$$I_j \stackrel{\text{def}}{=} 2^{-\hat{k}} \int_{\mathbb{R}^+} f_j^1(y) \log_2 \left(\frac{f_j^1(y)}{f_j(y)}\right) + (1-2^{-\hat{k}}) f_j^0(y) \log_2 \left(\frac{f_j^0(y)}{f_j(y)}\right) dy. \tag{8.9}$$

Cryptanalyse de type 3.

La seule différence avec le cas de la cryptanalyse multiple de type 2 est qu'ici on s'intéresse aussi à la clef interne. Ici, on a donc $\boldsymbol{K}'_j = (\hat{\boldsymbol{K}}, \tilde{K}_j)$ et les $Y_j^{\boldsymbol{k}_O}$ sont donc exactement les $\hat{Y}_j^{\boldsymbol{k}_O}$. L'entropie $H(\boldsymbol{K}')$ est donc égale à $\hat{k} + d$ où, on le rappelle, d est la dimension de l'espace engendré par les masques de clef. Le raisonnement effectué pour la cryptanalyse de type 2 reste vrai pour la cryptanalyse de type 3 vu que c'est aussi une attaque sur le dernier tour. La différence apparaît dans les densités de $Y_j^{\boldsymbol{k}_O}$. Ainsi, la densité $f_j^0(y)$ de $Y_j^{\boldsymbol{k}_O}$ sachant $\boldsymbol{K}'_j \neq \boldsymbol{k}_O$ est

$$f_j^0(y) = \varphi_j^0(y). \tag{8.10}$$

La densité $f_j^1(y)$ de $Y_j^{\boldsymbol{k}_O}$ sachant $\boldsymbol{K}'_j = \boldsymbol{k}_O$ est

$$f_j^1(y) = \frac{1}{2}\varphi_j^{-1}(y) + \frac{1}{2}\varphi_j^1(y). \tag{8.11}$$

En effet, cela dépend de $\tilde{\boldsymbol{K}}_j$. Et la densité $f_j(y)$ de $Y_j^{\boldsymbol{k}_O}$ est toujours définie par (8.8). La formule finale est toujours

$$I\left(\boldsymbol{K}'_j; \boldsymbol{Y}_j\right) \leq 2^{\hat{k}} I_j,$$

avec I_j définie dans (8.9) mais avec les densités définies par (8.10) et (8.11).

8.1.3 Discussion

Nous avons donné une formule générale et son expression pour les trois types de cryptanalyse linéaire. On va maintenant discuter les résultats obtenus. Tout d'abord, nous commencerons par montrer que la borne obtenue est fine. Nous présenterons pour cela, d'une part,

des expérimentations montrant que l'entropie observée est proche de cette borne et, d'autre part, une preuve théorique pour les cas particuliers de la cryptanalyse de type 1 et pour une taille de liste $\ell = 1$ que la probabilité de succès de l'attaque est proche de 1 si notre borne est négative. On en profitera pour comparer la borne obtenue sur N pour ce cas particulier à la formule dérivée dans [BDCQ04] afin de mettre en avant le pessimisme de cette dernière. On parlera ensuite du lien entre l'entropie $H\left(\boldsymbol{K}'|\boldsymbol{Y}\right)$ et la taille de liste ℓ et l'on appliquera cela au cas particulier de la cryptanalyse de type 3 de Matsui sur laquelle Pascal Junod a basé ses expérimentations dans [Jun01] et qui met en évidence le même phénomène de surestimation de la complexité.

Validation expérimentale. Nous avons testé la borne obtenue pour une cryptanalyse multiple de type 1 utilisant 76 approximations linéaires dont les masques de clef forment un espace de dimension 13. Le choix de cette attaque a été effectué essentiellement pour des raisons de coût de calcul. Il était en effet bien plus long d'effectuer l'expérience pour une attaque de type 2 ou 3.

La figure 8.1 montre l'évolution des entropies sur la clef estimées et observées pour différentes valeurs de N. On voit que la borne est très fine pour de faibles valeurs de N et donc de fortes valeurs de bruit. Ceci est plutôt agréable vu que l'on se place généralement dans cette zone pour une cryptanalyse.

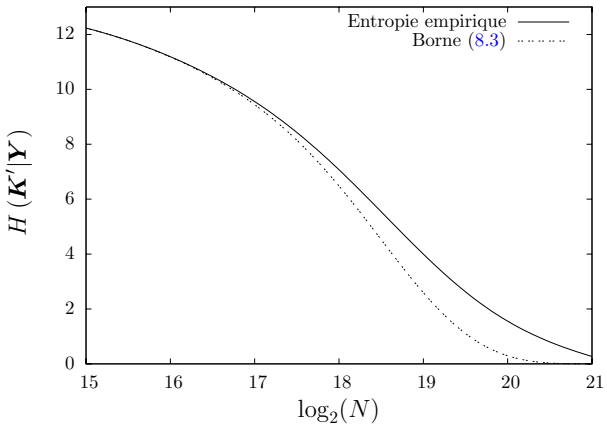

FIGURE 8.1 – Comparaison entre l'entropie observée et la borne (8.3).

Bien que la borne soit moins fine quand l'entropie s'approche de 0, elle reste néanmoins bonne : on peut prouver que quand la borne devient négative, alors la probabilité d'échec de l'attaque tends vers 0 avec les nombre d'approximations. La preuve de cette assertion est le sujet du paragraphe suivant.

Pertinence de la borne pour le type 1. On a dit que pour $H\left(\boldsymbol{K}'|\boldsymbol{Y}\right)$ proche de 0 on avait une probabilité de succès proche de 1 pour une attaque avec $\ell = 1$. La borne donnée dans le lemme 8.2 nous fait donc dire qu'il faut que

$$\sum_{j=1}^{n} \mathrm{Cap}\left(\sigma_j^2\right) \approx d.$$

On va donc montrer que si $\sum_{j=1}^{n} \mathrm{Cap}\left(\sigma_j^2\right) > d$, alors la probabilité d'erreur de l'attaque décroît. Le théorème 8.3 donne ce résultat de façon formelle.

Théorème 8.3. *Supposons que les bits des κ_j soient choisis de façon aléatoire uniforme, et que $\sum_{j=1}^{n} \mathrm{Cap}\left(\sigma_j^2\right) = d + \delta n$ pour un $\delta > 0$.*

Alors, si on note P_{err} la probabilité que la bonne clef $\boldsymbol{K}$ ne soit pas la plus vraisemblable connaissant les statistiques contenues dans $\boldsymbol{Y}$, il existe une constante A telle que

$$P_{\mathrm{err}} \leq \frac{A}{\delta^2 n} + 2^{-\delta n/2}.$$

Éléments de preuve : La preuve de ce théorème est assez semblable à la preuve de la partie directe du théorème de Shannon du codage canal (théorème 3.22) que l'on peut trouver dans [CT91]. Cependant, ne nous intéressant pas aux résultats asymptotiques, on ne peut reprendre intégralement cette preuve. Les preuves non-asymptotiques existantes étant, elles, prévues pour le cas particulier où les σ_j sont égaux. C'est pourquoi on ne peut que s'inspirer des preuves précédentes afin d'en fournir une qui corresponde à notre cas particulier.

La preuve repose sur l'utilisation des séquences typiques et se trouve à l'annexe B.3. □

Il faut noter que la probabilité d'erreur P_{err} est calculée sur $\boldsymbol{Y}$, sur $\boldsymbol{k}$ mais surtout sur tous les choix de masques κ_j. Il se peut donc que pour un choix particulier de masques, la probabilité d'erreur soit bien plus élevée. Les masques κ_j forment, comme on l'a déjà dit chapitre 7, un code linéaire. On va donc noter $\mathcal{C}$ un tel code et définir $P_{\mathrm{err}}(\mathcal{C})$ la probabilité d'erreur pour ce choix spécifique des masques. Supposons, que pour $\epsilon > 0$, $P_{\mathrm{err}} \leq \epsilon$. Comme $P_{\mathrm{err}} = \sum_{\mathcal{C}} P_{\mathrm{err}}(\mathcal{C})\Pr[\mathcal{C}]$, on s'attend à ce que la plupart des choix $\mathcal{C}$ aient une probabilité d'erreur proche de ϵ. Si on note $P \stackrel{\mathrm{def}}{=} \Pr[P_{\mathrm{err}}(\mathcal{C}) \geq t\epsilon]$, alors on peut prouver que cette probabilité décroît proportionnellement à t.

Pour cela, on regarde $P_{\mathrm{err}} = \sum_{\mathcal{C}} P_{\mathrm{err}}(\mathcal{C})\Pr[\mathcal{C}]$ que l'on peut brutalement minorer par la somme sur les codes $\mathcal{C}$ menant à une probabilité d'erreur supérieure à $t\epsilon$. On obtient $P_{\mathrm{err}} \geq Pt\epsilon$ ce qui donne $P \leq \frac{1}{t}$. Ceci permet de prouver le corollaire suivant.

Corollaire 8.4. *Supposons que $\sum_{j=1}^{n} \mathrm{Cap}\left(\sigma_j^2\right) = d + \delta n$ pour un $\delta > 0$. On note P_{err} la probabilité d'échec d'une attaque de type top ranking. Alors, il existe une constante A telle que pour tout $t > 0$,*

$$\Pr\left[P_{\mathrm{err}} \geq t\left(\frac{A}{\delta^2 n} + 2^{-\delta n/2}\right)\right] \leq \frac{1}{t}.$$

Démonstration : La preuve est très simple. On définit $P_{t\epsilon} \stackrel{\mathrm{def}}{=} \Pr_{\mathcal{C}}[P_{\mathrm{err}}(\mathcal{C}) \geq t\epsilon]$. Alors, on peut minorer $P_{\mathrm{err}} = \sum_{\mathcal{C}} P_{\mathrm{err}}(\mathcal{C})\Pr[\mathcal{C}]$ en utilisant le fait que, par définition, il y a une proportion $P_{t\epsilon}$ des codes pour lesquels $P_{\mathrm{err}}(\mathcal{C}) \geq t\epsilon$:

$$P_{\mathrm{err}} \geq P_{t\epsilon}\, t\, \epsilon.$$

Comme on a pour hypothèse que $P_{\mathrm{err}} \leq \epsilon$ on en conclut que $P_{t\epsilon} \leq \frac{1}{t}$. □

Comparaison avec [BDCQ04] pour le type 1. La motivation initiale de l'utilisation de l'entropie dans l'évaluation de la complexité en données était d'éviter d'avoir à passer par le calcul de l'espérance du rang de la bonne clef parmi les candidats car cette grandeur était trop influencée par des cas rares. On va donc, maintenant que l'on a dérivé notre borne, comparer les deux formules obtenues. Pour cela, nous utilisons la formule du gain donnée dans [BDCQ04] et rappelée dans le corollaire 4.13 [2].

$$\Gamma \approx -\log_2 \left(\frac{2(2^{n_k}-1)\Phi\left(-\sqrt{2\,N\sum_j \varepsilon_j^2}\right)+1}{2^{n_k}} \right).$$

Pour des paramètres de cryptanalyse, $2^{n_k} \gg 1$ et donc on peut utiliser une version simplifiée de cette formule

$$\Gamma \approx -\log_2 \left(2\,\Phi\left(-\sqrt{2N\sum_j \varepsilon_j^2}\right)\right).$$

On peut trouver, dans [Fel68] par exemple, l'estimation suivante du logarithme de la fonction de répartition de la loi normale : $\ln(\Phi(-x)) = -x^2/2(1+o\,(1))$. En appliquant cela, on obtient une formule du gain $\Gamma \approx \frac{N\sum_j \varepsilon_j^2}{\ln(2)} - 1$, et donc une formule pour une attaque cherchant à classer la bonne clef première parmi 2^d candidats qui demande une complexité en données

$$N_{\text{[BDCQ04]}} = \frac{(d+1)\ln(2)}{\sum_j \varepsilon_j^2}.$$

Pour ce qui est de notre formule, on a vu dans la sous-section 3.1.3 que la capacité du canal gaussien était de l'ordre de $\mathrm{Cap}\left(\sigma_j^2\right) \approx \frac{1}{2\sigma_j^2 \ln(2)}$. Notre borne sur l'entropie se traduit donc par la condition $\sum_{j=1}^{n} \frac{1}{2\sigma_j^2} \geq d$. Or, on rappelle que $\sigma_j^2 \stackrel{\text{def}}{=} 1/4N\varepsilon_j^2$ et donc on obtient la condition suivante sur N

$$N \geq \frac{d\ln(2)}{2\sum_j \varepsilon_j^2}.$$

En conclusion, on voit qu'il y a un facteur 2 entre l'estimation de N donnée dans [BDCQ04] et notre borne qui, on l'a vu dans la paragraphe précédent, est une bonne estimation de la complexité en données.

Entropie et complexité en temps. On a vu que si $H\left(\boldsymbol{K}'|\boldsymbol{Y}\right)$ est proche de 0, alors on a une attaque avec une taille de liste $\ell = 1$ qui a une probabilité de succès proche de 1. L'entropie quantifiant l'incertitude que l'on a sur une variable, il semble naturel, pour obtenir une bonne probabilité de succès, de prendre $\ell = 2^{H(\boldsymbol{K}'|\boldsymbol{Y})}$.

Conjecture. *Si, lors d'une cryptanalyse, on garde les $\ell = 2^{H(\boldsymbol{K}'|\boldsymbol{Y})}$ meilleurs candidats, alors la probabilité de succès de l'attaque est proche de 1.*

Je n'ai, pour le moment, pas de preuve que cela donne bien une probabilité de succès proche de 1 si ce n'est les expériences présentées dans la section 8.2 et les expériences de Junod dont on va parler maintenant.

2. Cette formule, on l'a vu, est optimiste du fait de l'utilisation de l'inégalité de Jensen comme une approximation. Cependant, nous l'utilisons quand même pour cette comparaison car notre but est de montrer que les formules dérivées en utilisant l'espérance du rang sont pessimistes par rapport à notre formule basée sur l'entropie. Utiliser cette formule erronée ne joue donc pas en notre faveur.

Comparaison aux résultats de [Jun01] pour le type 3. On va s'intéresser ici à la cryptanalyse proposée dans [Mat94a] et reprise par Junod afin d'en effectuer une étude théorique précise. L'analyse de l'attaque est effectuée en utilisant les statistiques d'ordre. Junod donne alors une formule précise pour calculer la distribution du rang de la bonne clef externe. Il en dérive un théorème pour calculer l'espérance du rang de la bonne clef. À la différence de [BDCQ04], on a là un théorème totalement fondé qui donne une complexité en temps de 2^{43} pour 2^{43} couples, ce qui correspond à la valeur avancée par Matsui. Cependant, suite à 42 expériences, Junod dit la chose suivante

We observe that Theorem 1 seems to give a pessimistic rank expected value. It is difficult to explain this fact because of the small statistical sample size.

En effet, l'espérance est influencée par les évènements rares pour lesquels la cryptanalyse se passe très mal. Sur 42 expériences, on n'aura rencontré aucun de ces évènements ce qui explique que la moyenne des rangs obtenus soit bien moindre que l'espérance théorique. Plus précisément, Junod explique que pour une complexité en données de 2^{43}, et bien que le théorème donne une complexité en temps de 2^{43}, la complexité observée est de l'ordre de 2^{41}. En prenant $N = 2^{43}$ et en appliquant notre borne, on obtient une entropie sur la clef d'environ 11 bits. Or, la recherche exhaustive s'effectue sur $56 - 26 = 30$ bits. Si l'on prend donc $\ell = 2^{H(\boldsymbol{K}'|\boldsymbol{Y})}$, notre borne donne donc une complexité de 2^{41} ce qui correspond à la complexité observée par Junod. Une fois encore, cela confirme l'intérêt de cette approche.

Pour avoir une valeur précise de la probabilité de succès que l'on obtient, il faudra alors utiliser la formule donnée par Junod sur la distribution du rang de la bonne clef.

Application à la cryptanalyse multidimensionnelle. On va maintenant voir que cette borne peut aussi s'appliquer à la cryptanalyse multidimensionnelle. L'idée est toujours d'utiliser la borne d'indépendance sur l'entropie (8.6) afin de borner l'information mutuelle par une somme d'information plus simples à calculer. Comme les différentes approximations utilisées sont dépendantes, on ne peut pas les traiter individuellement. Il va donc falloir s'y prendre autrement. Les variables

$$\boldsymbol{T}^i \stackrel{\text{def}}{=} g(\boldsymbol{P}^i) = \begin{pmatrix} \langle \pi_1, \boldsymbol{P}^i \rangle \oplus \langle \gamma_1, E_{\boldsymbol{K}}(\boldsymbol{P}^i) \rangle \oplus 1 \\ \vdots \\ \langle \pi_d, \boldsymbol{P}^i \rangle \oplus \langle \gamma_d, E_{\boldsymbol{K}}(\boldsymbol{P}^i) \rangle \oplus 1 \end{pmatrix},$$

sont indépendantes (car les couples sont supposés être obtenus de façon indépendante) et identiquement distribuées. On va donc décomposer l'information mutuelle en somme sur les couples :

$$I\left(\boldsymbol{K};\boldsymbol{T}\right) \leq \sum_{i=1}^{N} H\left(\boldsymbol{T}^i\right) - H\left(\boldsymbol{T}^i|\boldsymbol{K}\right).$$

Cette approche nous permet d'obtenir le résultat suivant.

Théorème 8.5. *Dans le cadre d'une cryptanalyse linéaire multidimensionnelle utilisant d approximations de base, l'information sur la clef obtenue en traitant N couples est bornée par :*

$$I\left(\boldsymbol{K};\boldsymbol{T}\right) \leq N \cdot \left(d - H\left(p^*\right)\right). \tag{8.12}$$

Démonstration : Tout d'abord, les variables $\boldsymbol{T}^i$ sont identiquement distribuées ce qui transforme la somme en

$$I(\boldsymbol{K};\boldsymbol{T}) \leq N \cdot (H(\boldsymbol{T}) - H(\boldsymbol{T}|\boldsymbol{K})).$$

De plus, $H(\boldsymbol{T}) = d$. Cela provient du fait que l'on sait que la distribution de $\boldsymbol{T}$ appartient à une famille de 2^d distributions ayant la propriété particulière donnée dans (4.6) que $p^*_{\boldsymbol{t}}(\tilde{\boldsymbol{k}} \oplus \boldsymbol{h}) = p^*_{\boldsymbol{t}\oplus\boldsymbol{h}}(\tilde{\boldsymbol{k}})$. Plus précisément,

$$H(\boldsymbol{T}) = -\sum_{\boldsymbol{t}\in\mathbb{F}_2^d} \Pr[\boldsymbol{T}=\boldsymbol{t}] \log_2(\Pr[\boldsymbol{T}=\boldsymbol{t}]).$$

Comme on ne connaît que la distribution de $\boldsymbol{T}$ selon la valeur de la clef $\tilde{\boldsymbol{K}}$, on partitionne l'espace

$$\Pr[\boldsymbol{T}=\boldsymbol{t}] = \sum_{\tilde{\boldsymbol{k}}\in\mathbb{F}_2^d} \Pr\left[\tilde{\boldsymbol{K}}=\tilde{\boldsymbol{k}}\right]\ p^*_{\boldsymbol{t}}(\tilde{\boldsymbol{k}}).$$

Or la distribution de la clef $\tilde{\boldsymbol{K}}$ est, a priori, uniforme et pour un $\boldsymbol{t}$ fixé, $\sum_{\tilde{\boldsymbol{k}}\in\mathbb{F}_2^d} p^*_{\boldsymbol{t}}(\tilde{\boldsymbol{k}})$ vaut 1. En effet, en utilisant (4.6)

$$\begin{aligned}\sum_{\tilde{\boldsymbol{k}}\in\mathbb{F}_2^{n_k}} p^*_{\boldsymbol{t}}(\tilde{\boldsymbol{k}}) &= \sum_{\tilde{\boldsymbol{k}}\in\mathbb{F}_2^d} p^*_{\boldsymbol{t}\oplus\tilde{\boldsymbol{k}}}(\boldsymbol{0}) \\ &= \sum_{\tilde{\boldsymbol{a}}\in\mathbb{F}_2^d} p^*_{\boldsymbol{a}}(\boldsymbol{0}),\end{aligned}$$

qui vaut donc 1 car p^* est une distribution. On obtient alors que

$$\Pr[\boldsymbol{T}=\boldsymbol{t}] = 2^{-d}, \tag{8.13}$$

et donc $H(\boldsymbol{T}) = d$. On a, à ce stade de la preuve,

$$I(\boldsymbol{K};\boldsymbol{T}) \leq N \cdot (d - H(\boldsymbol{T}|\boldsymbol{K})).$$

Reste alors à calculer l'entropie conditionnelle. Le fait de connaître la clef fait que l'on connaît la distribution p^* de $\boldsymbol{T}$. L'entropie conditionnelle vaut donc l'entropie de la distribution p^* car bien que p^* dépende de la valeur de $\boldsymbol{K}$, son entropie reste la même :

$$\begin{aligned}H(\boldsymbol{T}|\boldsymbol{K}) &= \sum_{\tilde{\boldsymbol{k}}\in\mathbb{F}_2^d} \Pr\left[\tilde{\boldsymbol{K}}=\tilde{\boldsymbol{k}}\right] H\left(\boldsymbol{T}|\tilde{\boldsymbol{K}}=\tilde{\boldsymbol{k}}\right) \\ &= H(p^*) \sum_{\tilde{\boldsymbol{k}}\in\mathbb{F}_2^d} \Pr\left[\tilde{\boldsymbol{K}}=\tilde{\boldsymbol{k}}\right] \\ &= H(p^*).\end{aligned}$$

On obtient donc la formule

$$I(\boldsymbol{K};\boldsymbol{T}) \leq N \cdot (d - H(p^*)).$$

Dans le cas de la cryptanalyse multidimensionnelle de type 2 et 3, on a, pour un couple, plusieurs compteurs $\boldsymbol{T}^i$. Le compteur correspondant à une clef externe $\boldsymbol{k}_{\mathrm{O}}$ sera noté $\boldsymbol{T}^i(\boldsymbol{k}_{\mathrm{O}})$. On a alors,

$$I(\boldsymbol{K};\boldsymbol{T}) \leq \sum_{i=1}^{N} \sum_{\boldsymbol{k}_{\mathrm{O}}} H\left(\boldsymbol{T}^i(\boldsymbol{k}_{\mathrm{O}})\right) - H\left(\boldsymbol{T}^i(\boldsymbol{k}_{\mathrm{O}})|\boldsymbol{K}\right).$$

Là les choses sont un peu compliquées mais l'on va obtenir la même formule que pour la cryptanalyse de type 1. En effet, on peut décomposer la somme $\sum_{\boldsymbol{k}_O} H\left(\boldsymbol{T}^i(\boldsymbol{k}_O)|\boldsymbol{K}\right)$ en deux : le terme correspondant à la bonne sous-clef et les $2^{k_O}-1$ autres correspondant aux mauvaises (si on note, 2^{k_O} le nombre de clefs externes). On a vu que quelle que soit la bonne clef, l'entropie sur le bon compteur connaissant la clef est $H(p^*)$. Dans le cas des mauvais candidats, les compteurs suivent, selon l'hypothèse de répartition aléatoire par fausse clef, une distribution uniforme sur $\mathbb{F}_2^d$ et donc leur entropie connaissant la bonne clef vaut d.

$$\sum_{\boldsymbol{k}_O} H\left(\boldsymbol{T}^i(\boldsymbol{k}_O)|\boldsymbol{K}\right) = -\sum_{\boldsymbol{a}\in\mathbb{F}_2^d} H(p^*) + (2^{k_O}-1)\cdot d.$$

En ce qui concerne l'entropie $H\left(\boldsymbol{T}^i(\boldsymbol{k}_O)\right)$, on va effectuer une partition de l'espace en deux.

$$\begin{aligned}\Pr\left[\boldsymbol{T}^i(\boldsymbol{k}_O)=\boldsymbol{t}\right] &= \Pr\left[\boldsymbol{T}^i(\boldsymbol{k}_O)=\boldsymbol{t}|\boldsymbol{k}_O=\boldsymbol{k}_O^*\right]\Pr\left[\boldsymbol{k}_O=\boldsymbol{k}_O^*\right]\\ &+ \Pr\left[\boldsymbol{T}^i(\boldsymbol{k}_O)=\boldsymbol{t}|\boldsymbol{k}_O\neq\boldsymbol{k}_O^*\right]\Pr\left[\boldsymbol{k}_O\neq\boldsymbol{k}_O^*\right].\end{aligned}$$

Or, $\Pr\left[\boldsymbol{k}_O=\boldsymbol{k}_O^*\right]=2^{-k_O}$, $\Pr\left[\boldsymbol{T}^i(\boldsymbol{k}_O)=\boldsymbol{t}|\boldsymbol{k}_O\neq\boldsymbol{k}_O^*\right]=2^{-d}$ et on note

$$p_{\boldsymbol{t}}^* \stackrel{\text{def}}{=} \Pr\left[\boldsymbol{T}^i(\boldsymbol{k}_O)=\boldsymbol{t}|\boldsymbol{k}_O=\boldsymbol{k}_O^*\right].$$

$$H\left(\boldsymbol{T}^i(\boldsymbol{k}_O)\right) = -\sum_{\boldsymbol{t}\in\mathbb{F}_2^d}\left(2^{-k_O}p_{\boldsymbol{t}}^* + (1-2^{-k_O})2^{-d}\right)\cdot\log_2(2^{-k_O}p_{\boldsymbol{t}}^* + (1-2^{-k_O})2^{-d}).$$

En utilisant (8.13), on obtient $H\left(\boldsymbol{T}^i(\boldsymbol{k}_O)\right)=d$ et donc

$$I(\boldsymbol{K};\boldsymbol{T}) \leq N\cdot(d-H(p^*)).$$

□

La figure 8.2 contient les résultats obtenus en appliquant cette borne à la cryptanalyse multidimensionnelle MK2 sur SERPENT présentée dans [HCN09a]. On compare ici la borne obtenue sur l'information mutuelle à l'avantage de l'attaque. La raison pour laquelle cette comparaison est valable est la suivante. L'entropie $H(\boldsymbol{K}|\boldsymbol{T})$ quantifie (en nombre de bits) l'incertitude sur la clef une fois les compteurs connus. On s'attend donc à ce que garder une liste de $2^{H(\boldsymbol{K}|\boldsymbol{T})}$ compteurs soit suffisant pour l'attaque. Or, dans ce cas, l'avantage de l'attaque vaut exactement $H(\boldsymbol{K})-H(\boldsymbol{K}|\boldsymbol{T})=I(\boldsymbol{K};\boldsymbol{T})$. Le rôle de cette information mutuelle est donc assez similaire à celui de l'avantage. Il est donc naturel de comparer ces deux grandeurs.

On remarque que la courbe donnée par la borne capture bien mieux le comportement probabiliste de l'attaque que l'estimation provenant de l'étude présentée dans [HCN09a]. Cependant, on peut se demander pourquoi cette borne s'avère être en dessous de la courbe empirique pour N petit. Une explication possible est le fait que les biais des approximations ont été estimés et donc potentiellement sous-estimés. Une autre explication possible est le fait que le nombre d'expériences effectuées pour tracer cette partie de la courbe était sans doute insuffisant. En effet, la courbe est censé être strictement croissante ce qui n'est pas le cas pour N inférieur à 2^{26} ce qui est un signe flagrant d'un nombre d'expériences trop faible.

Il serait donc intéressant d'effectuer des simulations sur une version jouet pour laquelle il est possible de calculer empiriquement les biais et pour laquelle il soit possible d'effectuer un grand nombre de simulations de façon à tracer une courbe précise.

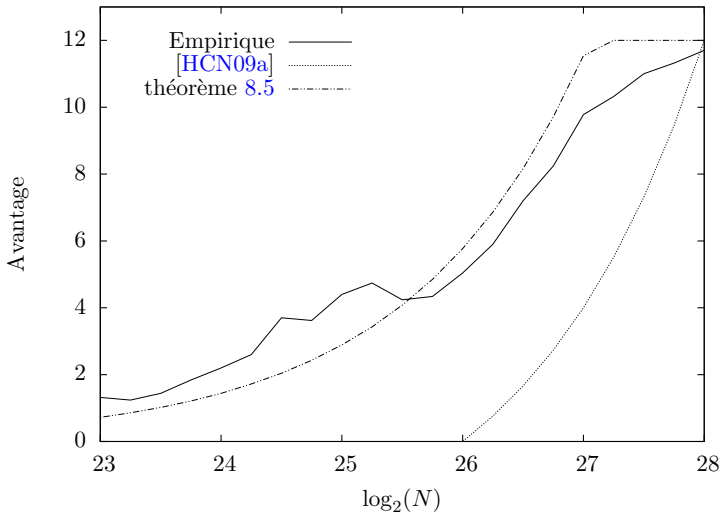

FIGURE 8.2 – Avantage empirique et théorique pour la cryptanalyse MK2 pour d=4[HCN09a].

8.2 Application : cryptanalyse du DES

On présente dans cette section les résultats expérimentaux obtenus pour une cryptanalyse multiple de type 1. Le but est de montrer qu'avec une forte probabilité, la bonne clef appartient à la liste $\mathcal{L}$ des $2^{H(\boldsymbol{K}'|\boldsymbol{Y})}$ candidats les plus vraisemblables.

Dans un premier temps nous nous attarderons sur les approximations utilisées (sous-section 8.2.1) puis nous expliquerons les techniques utilisées pour accélérer les phases de distillation (sous-section 8.2.2) et d'analyse (sous-section 8.2.3). Nous viendrons ensuite au problème de l'estimation de la complexité en temps d'une telle attaque (sous-section 8.2.4) puis, dans une dernière sous-section, nous calculerons les complexités attendues et nous les comparerons aux résultats expérimentaux ainsi qu'à la cryptanalyse de type 3 proposée par Matsui [Mat94a]. On verra que pour un nombre faible d'échantillons, l'attaque multiple de type 1 est plus puissante que la cryptanalyse de type 3 utilisant deux approximations sur 14 tours.

8.2.1 Approximations utilisées

On a recherché les meilleurs approximations linéaires du DES grâce à l'algorithme 7 présenté dans le chapitre 4.

On a pour cela fixé la borne inférieure sur le biais à $\varepsilon_{\min} = 2^{-28.84}$. L'algorithme nous a alors donné 74086 approximations linéaires. La figure 8.3 montre le code formé par les masques de clef par ces approximations. Comme on le voit, ce code possède une certaine structure. On va donc essayer d'ordonner les approximations afin de former des groupes faisant intervenir les mêmes bits de clefs. La figure 8.4 illustre ce groupement d'approximations.

Pour notre attaque, on retient, pour le moment, 6 groupes dont les caractéristiques sont données dans la tableau 8.1.

On remarque que G_2 contient les mêmes approximations que G_1 à inversion des masques d'entrée sortie près. Le même phénomène est visible pour les autres groupes. Cela vient du fait que déchiffrer avec le DES revient à chiffrer en changeant l'ordre des sous-clefs. C'est

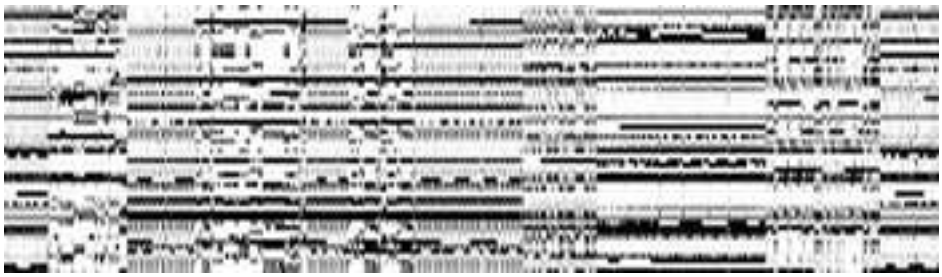

FIGURE 8.3 – Code engendré par les 74086 κ_j obtenus.

Groupe	n	Nb. de π_j différents	Nb. de γ_j différents	d
G_1	12384	1500	82	19
G_2	12384	82	1500	19
G_3	4100	64	82	13
G_4	4100	82	64	13
G_5	3476	82	1152	18
G_6	3476	1152	82	18

TABLE 8.1 – Groupes d'approximations formés.

pourquoi on obtient les groupes ayant un même nombre d'approximations avec une même dimension de l'espace des κ_j et des nombres de masques en entrée et sortie inversés.

Pour des raisons de rapidité et étant donné le faible apport, en terme de biais, des deux derniers groupes, on optera finalement pour une attaque utilisant les groupes G_1, G_2, G_3 et G_4. Les deux meilleures approximations sur 16 tours, correspondant à celle utilisée par Matsui pour son **Algorithme 1**, se trouvent dans les groupes G_3 et G_4. C'est pourquoi on n'utilise pas seulement G_1 et G_2.

On utilise donc 32968 approximations sur les 74086 meilleures trouvées. On a cependant fait attention de garder les approximations ayant un biais significativement plus élevé que les autres.

8.2.2 Accélération de la phase de distillation

Maintenant que les approximations sont choisies, on va s'intéresser au calcul des compteurs T_j. Pour une cryptanalyse simple de type 1, le coût est de N vu qu'il suffit de calculer un produit scalaire par couple. Pour la cryptanalyse multiple, il faut, a priori, calculer n produits scalaires et on a donc une complexité en $n\,N$. Comme dans notre cas, $n \approx 2^{15}$, on a plutôt intérêt à diminuer cette complexité.

Cela se fait facilement étant donné le tableau 8.1. Prenons, pour exemple, le groupe G_1. On va grouper les approximations en fonction de leur masque de sortie. On obtient 82 classes d'approximations. Pour chacune de ces classes, il s'avère que la dimension de l'espace généré par les masques d'entrée est au plus 14.

On s'intéresse à une de ces 82 classes. On note $\pi^1, \ldots, \pi^l$ une base de l'espace des masques d'entrée et γ le masque de sortie commun. On va alors évaluer, pour chaque couple, le vecteur

$$\left(\langle \pi^1, \boldsymbol{m}^i \rangle, \ldots, \langle \pi^l, \boldsymbol{m}^i \rangle, \langle \gamma, \boldsymbol{c}^i \rangle\right),$$

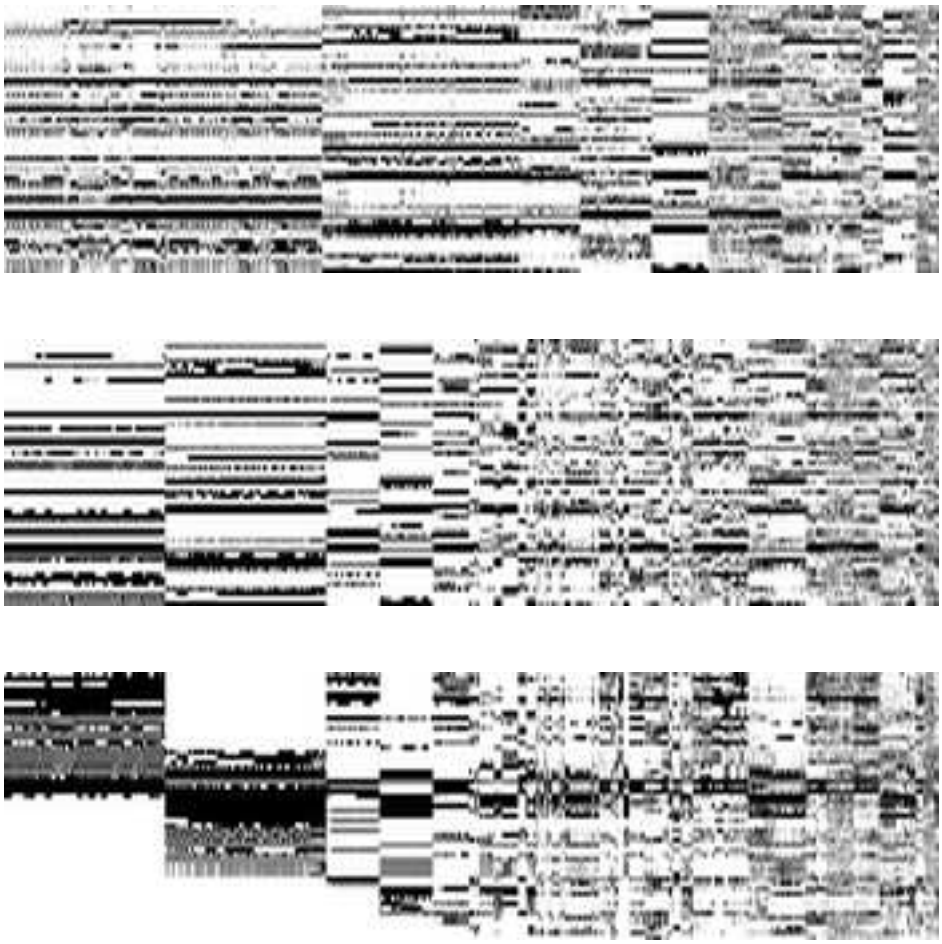

FIGURE 8.4 – Regroupement des approximations en fonction de leurs masques de clefs.

et stocker le nombre d'occurrences de chacune des 2^{l+1} valeurs possibles. On peut alors, grâce à une transformée de Walsh sur le tableau contenant ces occurrences, récupérer les valeurs des compteurs T_j pour les approximations de cette classe. Le coût de cette transformée est négligeable par rapport au coût des évaluations des produits scalaires vu que celle-ci n'est effectuée qu'une fois par classe. En effet, on a, en général, $15 \cdot 2^{15} \ll 15 \cdot N$.

Cette astuce peut être utilisée pour tous les groupes et on obtient alors une complexité de la phase de distillation de l'ordre de $N \cdot (82 + 82 + 64 + 64) \cdot 15$

8.2.3 Accélération de la phase d'analyse

Pour la phase d'analyse, on va utilise le fait d'avoir séparé les approximations en groupes. Chaque groupe est formé d'approximations dont les masques de clef engendrent un espace de petite dimension (≤ 19). On peut donc utiliser, de nouveau, une transformée de Walsh sur les compteurs pour obtenir les vraisemblances des clefs.

En effet, on rappelle que l'on utilise une statistique équivalente à la vraisemblance d'un

candidat $\boldsymbol{k}$ et facile à calculer en fonction du vecteur reçu $\boldsymbol{y}$:

$$v(\boldsymbol{k}) \stackrel{\text{def}}{=} \sum_{j=1}^{n} (-1)^{<\kappa_j, \boldsymbol{k}>} \frac{y_j}{\sigma_j^2}.$$

Cette somme peut se décomposer en quatre sommes sur chacun des groupes :

$$v(\boldsymbol{k}) = v_1(\boldsymbol{k}) + v_2(\boldsymbol{k}) + v_3(\boldsymbol{k}) + v_4(\boldsymbol{k}).$$

Pour un groupe donné, les masques de clef des approximations définissent un espace vectoriel de dimension d_i. On peut donc écrire ces masques comme combinaison linéaire de d_i masques de base (comme plus haut) $(\kappa^1, \ldots, \kappa^l)$. Si l'on définit la fonction

$$f(\boldsymbol{x}) \stackrel{\text{def}}{=} \begin{cases} \frac{y_j}{\sigma_j^2} \text{ si } \exists j,\ \oplus_i x_i \cdot \kappa^i = \kappa_j, \\ 0 \text{ sinon.} \end{cases}$$

Alors, la transformée de cette fonction en un point $\boldsymbol{k} = (k_1, \ldots, k_l)$ est justement la vraisemblance $v_i(\boldsymbol{k})$ du candidat $\boldsymbol{k}$ sur ce groupe d'approximations.

Calcul du rang de la bonne clef. Le problème est que l'on récupère de l'information sur 42 bits de clef. Comme l'on souhaite faire tourner l'attaque plusieurs fois, on ne peut se permettre de calculer les vraisemblances des 2^{42} candidats pour calculer le rang de la bonne clef.

L'astuce pour estimer la complexité de l'attaque, est que l'on connaît cette bonne clef vu que l'on aura généré nous-mêmes les échantillons. On peut donc calculer la vraisemblance $v(\boldsymbol{k}^*)$ de cette clef. Le problème est alors différent : il faut juste calculer le nombre de candidats ayant une vraisemblance supérieure à $v(\boldsymbol{k}^*)$.

Tout d'abord, il faut savoir que si l'on fusionne les groupes G_1 et G_3 (ou G_2 et G_4), la dimension des masques de clef n'est pas $19 + 13 = 32$ mais 26 car 6 bits d'information sont communs aux deux groupes. Si l'on regarde les deux groupes fusionnés $G_{13} = G_1 \cup G_3$ et $G_{24} = G_2 \cup G_4$, ils ont eux, 10 bits d'information en commun. On obtient donc au final 42 bits d'information sur la clef comme précisé plus haut.

Notons $\mathcal{L}_i$ la liste de candidats obtenue par le groupe G_i :

$$\mathcal{L}_i \stackrel{\text{def}}{=} \{(\boldsymbol{k}, v_i(\boldsymbol{k})), \boldsymbol{k} \in \mathbb{F}_2^{d_i}\}.$$

On définit les listes fusionnées

$$\mathcal{L}_{13} \stackrel{\text{def}}{=} \{(\boldsymbol{k}_1||\boldsymbol{k}_3, v_1(\boldsymbol{k}_1) + v_3(\boldsymbol{k}_3)), (\boldsymbol{k}_1, v_1(\boldsymbol{k}_1), \boldsymbol{k}_3, v_3(\boldsymbol{k}_3)) \in \mathcal{L}_1 \times \mathcal{L}_3\},$$
$$\mathcal{L}_{24} \stackrel{\text{def}}{=} \{(\boldsymbol{k}_2||\boldsymbol{k}_4, v_2(\boldsymbol{k}_2) + v_4(\boldsymbol{k}_4)), (\boldsymbol{k}_2, v_2(\boldsymbol{k}_2), \boldsymbol{k}_4, v_4(\boldsymbol{k}_4)) \in \mathcal{L}_2 \times \mathcal{L}_4\}.$$

On peut calculer $\mathcal{L}_{13}$ de façon efficace. On sépare $\mathcal{L}_1$ en 2^6 listes correspondant aux valeurs des candidats sur les 6 bits communs. On trie ces listes ainsi que $\mathcal{L}_3$ et on peut ensuite parcourir les 2^{26} candidats de la liste fusionnée avec une complexité en 2^{26}.

On peut, enfin, calculer le nombre de candidats ayant une vraisemblance supérieure à $v(\boldsymbol{k}^*)$ de la façon suivante. De nouveau, on sépare $\mathcal{L}_{13}$ et $\mathcal{L}_{24}$ en 2^{10} listes correspondant aux valeurs des candidats sur les 10 bits communs. On a donc $2 \cdot 2^{10}$ sous-listes de taille 2^{16}, $\mathcal{L}_{13,i}$ et $\mathcal{L}_{24,i}$.

Pour un i fixé, i.e. une valeur pour les 10 bits communs fixée, on parcourt $\mathcal{L}_{13,i}$ et, pour chaque candidat $\boldsymbol{k}_{13,i}$ de cette sous-liste, on effectue une recherche dichotomique dans la sous-liste $\mathcal{L}_{24,i}$ afin de trouver le nombre de candidats $\boldsymbol{k}_{24,i}$ tels que $v_{13}(\boldsymbol{k}_{13,i}) + v_{24}(\boldsymbol{k}_{24,i}) > v(\boldsymbol{k}^*)$.

La complexité globale d'une phase d'analyse menée ainsi est donc dominée par la génération des listes fusionnées $\mathcal{L}_{13}$ et $\mathcal{L}_{24}$ et est donc de l'ordre de 2^{26}.

8.2.4 Calcul de la complexité en temps

On a déjà vu quelle était la complexité en temps de la phase de distillation $N \cdot 292 \cdot 15$. De même, on a vu que la phase d'analyse a une complexité 2^{26}. On en déduit donc que cette dernière a une complexité négligeable dans l'attaque car la phase de recherche est égale au nombre de candidats à tester (ℓ) multiplié par la complexité de la recherche sur les bits restant (2^{14}).

On va donc s'attarder sur la complexité de cette phase de recherche finale. On l'a dit, la complexité est $\ell \cdot 2^{14}$. On pourrait donc calculer la complexité de l'attaque en prenant l'espérance de la valeur de ℓ mais comme cela a été expliqué dans la sous-section 8.1.3, ceci donne des résultats pessimistes. On va donc utiliser la borne obtenue à la section 8.1.

On prend donc $\ell = 2^{H(\boldsymbol{K}'|\boldsymbol{Y})}$ avec pour estimation de cette entropie les bornes dérivées dans la section 8.1. On a alors une complexité de la phase de recherche qui est $2^{14}\, 2^{H(\boldsymbol{K}'|\boldsymbol{Y})}$. En utilisant notre borne comme une approximation, on obtient $2^{14}\, 2^{H(\boldsymbol{K}') - \sum_j I\left(K'_j; Y_j\right)}$.

Unités de complexité. Si on récapitule les complexités de notre attaque et de celle de Matsui [Mat94a] en utilisant l'astuce présentée dans [CSQ07] et la bonne statistique [Jun01], on obtient

Attaque	Distillation	Analyse	Recherche
Matsui	$N \cdot 2 \cdot 13 \cdot \nu$	$12 \cdot 2^{12} \cdot \nu$	$2^{H(\boldsymbol{K}'|\boldsymbol{Y})} \cdot \theta$
Type 1 multiple	$N \cdot (82 + 82 + 64 + 64) \cdot 15 \cdot \nu$	$2^{26} \cdot \nu$	$2^{H(\boldsymbol{K}'|\boldsymbol{Y})} \cdot \theta$

TABLE 8.2 – Complexité des phases pour deux cryptanalyses linéaires multiples.

Les variables ν et θ sont des unités de complexité qui correspondent à une opération binaire pour ν et un chiffrement par l'algorithme attaqué (ici le DES) pour θ. La question de la comparaison des différentes complexités se pose alors.

Suite à des simulations avec une implémentation standard du DES, on peut estimer que $\theta \approx 600 \cdot \nu$. Cependant, avec des versions plus sophistiquée du DES on peut descendre bien en-dessous (à notre désavantage). On va donc regarder les résultats obtenus pour les trois valeurs de θ suivantes

- $\theta = 600 \cdot \nu$;
- $\theta = 400 \cdot \nu$;
- $\theta = 200 \cdot \nu$.

Ainsi, on prend en compte les améliorations possibles de l'implémentation du DES.

8.2.5 Résultats

On a donc tous les outils en main pour effectuer cette attaque et comparer sa complexité à celle de Matsui.

Résultats théoriques. Afin de comparer notre attaque à celle de Matsui, on utilise les bornes données en section 8.1 et les complexités du tableau 8.2. On obtient alors les complexités données dans la figure 8.5. On voit que pour des valeurs de N inférieures à 2^{42}, la

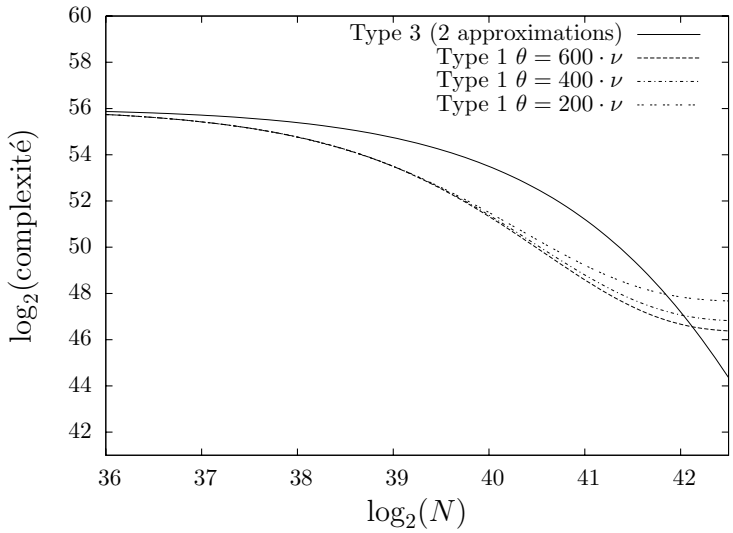

FIGURE 8.5 – Complexités théoriques de deux cryptanalyses linéaires multiples.

cryptanalyse multiple de type 1 arrive à de meilleures performances qu'une cryptanalyse de type 2 utilisant pourtant 2 approximations sur 14 tours.

On observe qu'à partir d'une valeur de N supérieure à 2^{41} l'écart entre les deux attaques diminue. C'est à ce moment que la complexité de la phase de distillation commence à devenir aussi grande que celle de la phase d'analyse. Passé un certain seuil $N_{\max}$, la complexité globale de l'attaque commence même à augmenter. Dans ce cas, il est plus intéressant de se contenter d'utiliser les $N_{\max}$ premiers couples et de jeter les autres. C'est pourquoi la complexité de cette attaque stagne à partir de cette valeur $N_{\max}$ du nombre d'échantillons.

Résultats expérimentaux. Si on regarde la figure 8.5, on voit que pour 2^{39} couples la complexité est de 2^{54}. Celle-ci étant dominée par la phase de recherche, cela signifie que l'entropie que l'on a sur les 42 bits de clef $H\left(\boldsymbol{K}'|\boldsymbol{Y}\right)$ est de 40 bits. On s'attend donc à ce que prendre une liste de 2^{40} candidats donne une bonne probabilité de succès.

Ceci est confirmé par les 19 expériences effectuées pour cette attaque. Les rangs obtenus pour la bonne clef sont listés par ordre croissant ci-dessous.

$$2^{31.34}, 2^{33.39}, 2^{34.65}, 2^{35.24}, 2^{36.56}, 2^{37.32}, 2^{37.72}, 2^{37.99}, 2^{38.11}, 2^{38.52}, 2^{38.97},$$

$$2^{39.04}, 2^{39.19}, 2^{39.27}, 2^{39.53}, 2^{39.85}, 2^{40.28}, 2^{40.82}, 2^{40.88}.$$

On voit que pour 16 cas sur 19 une taille de liste de 2^{40} est suffisante ce qui donne une probabilité de succès expérimentale de 84%.

Ce nombre d'expériences qui est de 19 est un peu trop petit pour estimer finement la probabilité de succès mais il est à rapporter au fait qu'une expérience représente une semaine de calcul sur un cluster de 40 processeurs.

Chapitre 9

Cryptanalyse différentielle multiple

Sommaire

Dans ce chapitre, l'on va s'intéresser à l'utilisation de plusieurs différentielles lors d'une cryptanalyse différentielle sur le dernier tour. Quasiment toutes les cryptanalyses différentielles utilisent plusieurs différentielles avec des différences en entrée différentes de façon à pouvoir former plus d'échantillons à partir d'un même nombre de couples clair/chiffré. Cependant, ces différentielles ont la même différence en sortie de façon à pouvoir utiliser un crible unique pour tous les couples (et, on le verra, accélerer ainsi l'analyse des données disponibles). À ma connaissance, il n'existe aucune attaque utilisant un groupe quelconque de différentielles avec plusieurs différences en entrée et en sortie (si ce n'est la différentielle tronquée mais dans ce cas les différences en sortie sont les même pour chacune des différences en entrée ce qui est restrictif).

Dans un premier temps, dans la section 9.1, on présentera la cryptanalyse différentielle utilisant des différentielles ayant la même différence en sortie telle qu'on la trouve dans la littérature. On passera ensuite à la présentation et à l'étude de la *cryptanalyse différentielle multiple* qui utilise un ensemble quelconque de différentielles dans la section 9.2. Cette attaque peut être considérée comme nouvelle au sens où aucune attaque parue à ce jour n'utilise des différentielles avec différentes différences en sortie. De plus, l'étude théorique d'une cryptanalyse utilisant plusieurs différentielles n'a encore jamais été faite proprement (et ce même si l'on a la même différence en sortie). On terminera par présenter une confirmation expérimentale de ces travaux et proposer une cryptanalyse différentielle multiple sur 18 tours du chiffrement PRESENT.

9.1 Travaux actuels

À ce jour, et à notre connaissance, il n'existe aucune base théorique relative à l'utilisation de plusieurs différentielles. L'analyse des attaques s'effectue en calculant, de façon plus ou moins grossière, le nombre de bonnes paires obtenues en moyenne pour un certain nombre N de couples ainsi que la valeur moyenne des compteurs correspondant aux mauvais candidats. On va rapidement détailler le fonctionnement et l'analyse des cryptanalyses différentielles actuelles qui utilisent plusieurs différentielles ayant la même différence en sortie.

9.1.1 Structures de données

Dans le cadre de la cryptanalyse différentielle multiple, on utilise plusieurs différentielles. Pour le moment, comme l'on se préoccupe des attaques existantes, on suppose que les différentielles ont la même différence en sortie δ_r. On peut choisir judicieusement les clairs de façon à optimiser le nombre d'échantillons formés avec N couples. Notons Δ_0 l'ensemble des différences en entrée des différentielles considérées. On choisit des ensembles de clairs comme des translatés d'espaces vectoriels appelés *structures*.

Définition 9.1. Une *structure* $\mathcal{S}(\boldsymbol{p}^0, \Delta_0)$ correspondant à un clair $\boldsymbol{p}^0$ et à un ensemble de différentielles en entrée $\Delta_0 = \{\delta_0^{(1)}, \ldots, \delta_0^{(|\Delta_0|)}\}$, est défini par

$$S(\boldsymbol{p}^0, \Delta_0) \stackrel{\text{def}}{=} \left\{ \boldsymbol{p}, \boldsymbol{p} \oplus \boldsymbol{p}^0 \in Vect(\delta_0^{(1)}, \ldots, \delta_0^{(|\Delta_0|)}) \right\}.$$

Pour une structure de taille $|\mathcal{S}|$, on a donc $\frac{|\mathcal{S}|}{2}$ paires pour chacune des différences en entrée.

9.1.2 Criblage des paires

De façon similaire à la cryptanalyse différentielle simple, on nomme bonne paire une paire qui vérifie une des différentielles considérées et qui incrémentera donc le compteur de la bonne clef. Une bonne paire est donc un couple $\boldsymbol{p}, \boldsymbol{p}'$ tel que $\boldsymbol{p} \oplus \boldsymbol{p}' \in \Delta_0$ et dont la différence entre les chiffrés après r tours vaut bien δ_r Le but est toujours d'essayer de détecter le plus de mauvaises paires à l'aide d'un crible. Or, ici, le nombre de paires générées est $\frac{|\Delta| \cdot N}{2}$ et donc il sera très peu efficace de générer toutes les paires puis de les passer toutes au crible. L'astuce est d'utiliser une table de hachage. On va ranger les couples clair/chiffré dans une table à une adresse correspondant à certains bits du chiffré. Ces bits correspondent aux bits étant nuls pour toutes les différences du crible $\mathcal{C}_{\delta_r}$. Deux couples situés dans deux cases différentes formeront alors une paire qui, de façon certaine, ne passera pas le crible. Il suffira donc de passer au crible les paires formées par les couples d'une même case ayant une différence entre clairs présente dans Δ_0. Cela permet donc de générer les paires et d'effectuer le crible de façon efficace.

9.1.3 Analyse de la complexité et de la probabilité de succès

Pour ce qui est de l'analyse de ce type d'attaques, elle est effectuée de façon assez grossière. Ces analyses sont généralement menées de la façon suivante :

1. calcul du nombre total de bonnes paires ;
2. calcul de la probabilité pour un couple de passer le crible ;

3. calcul du nombre total de paires ayant passé le crible ;
4. calcul de la valeur moyenne des compteurs correspondant aux mauvais candidats ;
5. calcul de la probabilité de succès avec la formule donnée dans [Sel08].

Le calcul à l'étape 1 est facile à effectuer. Pour N couples on peut former $N/2$ paires pour chaque différentielle et donc ce nombre de bonnes paires est égal à $\sum_{\delta^{(j)} \in \Delta} \frac{N}{2} \cdot p_0^{(j)}$.

Les calculs des étapes 2 et 3 sont, a priori, simples à mener car il suffit de multiplier le nombre total de paires $\frac{|\Delta|\, N}{2}$ par la probabilité de passer le crible. Cependant, il arrive que ces calculs soient menés « à la louche » donnant des résultats inexacts.

Toutes les paires n'incrémentant pas le même nombre de compteurs, il est assez compliqué de faire une analyse exacte de la complexité de l'étape 4. Une erreur contenue dans l'analyse de l'attaque dans [Wan08] provient justement de ce calcul. Il est cependant possible d'utiliser l'approche proposée dans la sous-section 5.1.4.

Enfin, la dernière étape est d'estimer la probabilité de succès de l'attaque pour un avantage donné. Pour cela la formule proposée dans [Sel08] est utilisée dans tous les papiers depuis sa publication. Cependant, on a vu dans le chapitre 6 que celle-ci est loin d'être bonne dans le cas de la cryptanalyse différentielle. De plus, elle suppose que les compteurs considérés suivent des lois binomiales. Or, dans le cas de plusieurs différentielles, ce n'est plus le cas. On propose, maintenant, de regarder en détails cette question de l'analyse théorique propre de la cryptanalyse différentielle multiple. On en profitera pour regarder le cas de différentielles avec plusieurs différences en sortie.

9.2 Analyse de la cryptanalyse différentielle multiple

L'objectif de cette section est de donner des résultats théoriques généraux et précis sur la probabilité de succès d'une cryptanalyse différentielle multiple. On vérifiera ensuite expérimentalement les résultats obtenus sur SMALLPRESENT-[8] avant d'extrapoler une attaque sur 18 tours de PRESENT ce qui est la meilleure cryptanalyse différentielle de ce chiffrement à ce jour.

9.2.1 Complexité en données et probabilité de succès

Notations et définitions. On va, ici, s'intéresser au cadre général dans lequel on a des différentielles avec plusieurs différences en entrée mais aussi plusieurs différences en sortie. Pour deux différences en entrée, on ne considérera pas forcément les même différences en sortie (et réciproquement). On suppose que l'on a un ensemble de différentielles Δ. On note Δ_0 l'ensemble des différences en entrée des différentielles contenues dans Δ.

$$\Delta_0 \stackrel{\text{def}}{=} \{\delta_0, \exists \delta_r, (\delta_0, \delta_r) \in \Delta\}.$$

Pour une différence en entrée $\delta_0^{(i)}$ donnée de Δ_0, on note $\Delta_r^{(i)}$ l'ensemble des différences en sorties correspondantes

$$\Delta_r^{(i)} \stackrel{\text{def}}{=} \{\delta_r, (\delta_0^{(i)}, \delta_r) \in \Delta\}.$$

On verra plus loin en quoi cette façon de regrouper les différentielles est naturelle. On a donc des différentielles $(\delta_0^{(i)}, \delta_r^{(i,j)})$ dont on note les probabilités $p_0^{(i,j)}$:

$$p_0^{(i,j)} \stackrel{\text{def}}{=} \Pr_{\boldsymbol{M},\boldsymbol{K}}\left[E_{\boldsymbol{K}}(\boldsymbol{M}) \oplus E_{\boldsymbol{K}}(\boldsymbol{M} \oplus \delta_0^{(i)}) = \delta_r^{(i,j)}\right].$$

On va alors définir, comme dans toute cryptanalyse statistique, des compteurs permettant d'extraire l'information des échantillons disponibles.

Définition 9.2. Soit $D^{(i)}_{\boldsymbol{m}}(\boldsymbol{k})$ le compteur correspondant au clair $\boldsymbol{m}$, à un candidat $\boldsymbol{k}$ et à l'ensemble de différentielles $(\delta_0^{(i)}, \delta_r)$ avec $\delta_r \in \Delta_r^{(i)}$.

$$D^{(i)}_{\boldsymbol{m}}(\boldsymbol{k}) \stackrel{\text{def}}{=} \begin{cases} 1 \text{ si } F_{\boldsymbol{k}}^{-1}\big(E_{\boldsymbol{k}^*}(\boldsymbol{m})\big) \oplus F_{\boldsymbol{k}}^{-1}\big(E_{\boldsymbol{k}^*}(\boldsymbol{m} \oplus \delta_0^{(i)})\big) \in \Delta_r^{(i)}, \\ 0 \text{ sinon.} \end{cases}$$

Pour une différence en entrée, un clair et une clef fixés, on n'a qu'une différence en sortie possible. Les variables $D^{(i)}_{\boldsymbol{m}}(\boldsymbol{k})$ suivent donc une loi de Bernoulli de paramètre

$$p_0^{(i)} \stackrel{\text{def}}{=} \sum_{j=1}^{\left|\Delta_r^{(i)}\right|} p_0^{(i,j)},$$

si $\boldsymbol{k} = \boldsymbol{k}^*$ et

$$p^{(i)} \stackrel{\text{def}}{=} \left|\Delta_r^{(i)}\right| \cdot 2^{-s},$$

sinon.

On a alors deux approches possibles, la première est de considérer les sommes sur $\boldsymbol{m}$ de ces compteurs.

$$D^{(i)}(\boldsymbol{k}) \stackrel{\text{def}}{=} \sum_{\boldsymbol{m}} D^{(i)}_{\boldsymbol{m}}(\boldsymbol{k}).$$

On obtient alors des variables binomiales suivant des lois ayant pour paramètres $(N, p_0^{(i)})$ ou $(N, p^{(i)})$. La distribution de la somme de ces variables n'est alors pas évidente à calculer car les probabilités seront, en général, différentes selon le i. La seconde approche est de sommer les compteurs pour un clair $\boldsymbol{m}$ fixé.

$$D_{\boldsymbol{m}}(\boldsymbol{k}) \stackrel{\text{def}}{=} \sum_{i=1}^{|\Delta_0|} D^{(i)}_{\boldsymbol{m}}(\boldsymbol{k}).$$

On obtient alors des variables identiquement distribuées mais pas forcément indépendantes. Un résultat que l'on peut trouver dans [Gal68] permet de donner une bonne estimation des queues de la distribution d'une somme de variables i.i.d. (ce résultat permet entre autre de retrouver le comportement asymptotique de la distribution binomiale utilisé dans le chapitre 6). L'approche qui semble pouvoir être la plus fructueuse est donc la seconde. On obtient donc des variables identiquement distribuées selon une somme de lois de Bernoulli ayant des paramètres différents. Le problème est que ces variables ne sont pas indépendante. En effet, on peut remarquer que $D^{(i)}_{\boldsymbol{m}}(\boldsymbol{k}) = D^{(i)}_{\boldsymbol{m}\oplus\delta_0^{(i)}}(\boldsymbol{k})$. On a donc une dépendance qui provient du fait que l'on compte deux fois chaque paire. Pour éviter cela, on va devoir effectuer la somme sur $\boldsymbol{m}$ avec précaution et ce ne sera pas toujours possible. Pour étudier ce problème, on va représenter une structure sous forme de graphe. Soit Δ_0 un ensemble de différences en entrée, on construit le graphe $\mathcal{G}_{\Delta_0}$ dont les points sont les clairs de la structure. On relie $\boldsymbol{m}_1$ et $\boldsymbol{m}_2$ si et seulement si $\boldsymbol{m}_1 \oplus \boldsymbol{m}_2 \in \Delta_0$. Le problème de ne pas sommer deux fois le même compteur est équivalent au problème de trouver un ensemble $\mathcal{P}$ de clairs tel que l'ensemble des arêtes incidentes aux points de $\mathcal{P}$ contient toutes les arêtes du graphe avec multiplicité 1.

Définition 9.3. On dit d'un ensemble de différences en entrée Δ_0 qu'il est *admissible* si le graphe $\mathcal{G}_{\Delta_0}$ est biparti.

Si Δ_0 est admissible, alors il suffit de prendre pour $\mathcal{P}$ une de ces deux parties. De plus, les graphes correspondant à différentes structures sont isomorphes et donc la façon dont est choisi l'ensemble $\mathcal{P}$ sera la même pour toutes les structures. Regardons maintenant comment déterminer simplement si Δ_0 est admissible ou non.

Proposition 9.4. *Soit $\Delta_0 = \{\delta_0^{(1)}, \dots, \delta_0^{(n)}\}$ un ensemble de différences. Alors,*

$$\Delta_0 \text{ est admissible} \iff \text{le graphe } \mathcal{G}_{\Delta_0} \text{ ne contient pas de cycles impairs.}$$

Si on note G la matrice formée par les $\delta_0^{(i)}$ écrits en colonne et G' la forme systématique de G (obtenue avec un pivot de Gauss), alors

$$\Delta_0 \text{ est admissible} \iff (1 \cdots 1) \cdot G' = (1 \cdots 1).$$

Démonstration. La condition sur les cycles impairs découle directement de la définition d'un graphe biparti. D'un point de vue codes correcteurs, on regarde le code formé par les $\delta_0^{(i)}$ écrits en colonne. La condition sur les cycles est alors équivalente au fait que toute combinaison impaire des colonnes ne soit pas nulle. Ce qui équivaut au fait que le dual de ce code ne contient que des mots de poids pair. Or, un code ne contient que des mots de poids pair si et seulement si son dual contient le mot tout à 1. Comme le dual du dual d'un code est ce même code, on en déduit que le mot tout à 1 doit appartenir au code généré par les $\delta_0^{(i)}$ en colonne. Une technique efficace qui permet de déterminer si c'est le cas est de mettre la matrice G du code sous forme systématique (G') et d'encoder le vecteur tout à 1. On a alors équivalence entre le fait que le vecteur à 1 appartienne au code et $(1 \cdots 1) \cdot G' = (1 \cdots 1)$. □

On va avoir besoin de supposer l'indépendance des variables $(D_{\boldsymbol{m}}(\boldsymbol{k}))_{\boldsymbol{m}\in\mathcal{P}}$. Bien que l'on ait retiré la plus grande source de dépendance, il reste encore des corrélations entre les compteurs. En effet, si $|\Delta_0| > 2$ alors on calcule plus de $E(\boldsymbol{m}) \oplus E(\boldsymbol{m} \oplus \delta_0)$ qu'il n'y a de clairs dans la structure. On obtient donc un système surdéterminé qui nous permet d'exprimer certaines différences $E(\boldsymbol{m}) \oplus E(\boldsymbol{m} \oplus \delta_0)$ en fonction des autres i.e. utilisant un autre $\boldsymbol{m}$ ou un autre δ_0. On a vu dans la section 4.4 que bien que l'on ait des compteurs dépendant de variables linéairement liées, la somme de ces compteurs sur un grand nombre d'échantillons pouvait rendre cette dépendance négligeable. Ici on ne peut pas utiliser le même raisonnement car les probabilités considérées sont telles que l'approximation gaussienne utilisée alors n'est pas pertinente. Cependant, l'information donnée par les compteurs n'est pas la valeur de $E(\boldsymbol{m}) \oplus E(\boldsymbol{m} \oplus \delta_0)$ mais son appartenance à un ensemble ou non. Vu que ces ensembles sont petits et que la probabilité que la différence appartienne à celui-ci est faible, il est assez évident que la quantité d'information que l'on peut obtenir de plusieurs compteurs pour deviner un autre est assez limitée. Par manque de temps, on ne quantifiera pas cette information et on supposera juste que l'hypothèse suivante est raisonnable.

Hypothèse 9.5. *Hypothèse d'indépendance des différentielles.*
Soit $\mathcal{P}$ un ensemble Δ_0-admissible, alors les compteurs $(D_{\boldsymbol{m}}(\boldsymbol{k}))_{\boldsymbol{m}\in\mathcal{P}}$ sont statistiquement indépendants.

On a donc des compteurs i.i.d. qui suivent une distribution qui résulte de la somme de $|\Delta_0|$ variables de Bernoulli de probabilités $p_0^{(i)}$ ou $p^{(i)}$. Les N couples clair/chiffré disponibles lors d'une cryptanalyse différentielle multiple sont en fait des couples obtenus à partir de structures de clairs. On a donc $N/|\mathcal{S}|$ structures de taille $|\mathcal{S}|$ que l'on va indexer $(\mathcal{S}_j)_j$. Pour chacune de ces structures $\mathcal{S}_j$ on définit un ensemble Δ_0-admissible $\mathcal{P}_j$. On peut maintenant définir le compteur final qui sera considéré

$$D(\boldsymbol{k}) \stackrel{\text{def}}{=} \sum_j \sum_{\boldsymbol{m}\in\mathcal{P}_j} D_{\boldsymbol{m}}(\boldsymbol{k}).$$

Une quantité important ici est le nombre de compteurs de base que l'on va additionner qui est égal au nombre de paires considérées. On définit $N_s \stackrel{\text{def}}{=} \frac{|\Delta_0|\,N}{2}$ le nombre d'échantillons disponibles (comprendre le nombre de paires formées).

Distribution de $D(k)$. Le théorème de Le Cam que l'on peut trouver dans l'annexe A, nous dit qu'une somme de variables de Bernoulli a une distribution proche d'une distribution de Poisson (si leurs paramètres sont petits). En appliquant ce théorème aux $D_{\boldsymbol{m}}^{(i)}(\boldsymbol{k})$ on obtient que la distribution des compteurs $D_{\boldsymbol{m}}(\boldsymbol{k})$ est proche d'une distribution de Poisson de paramètre $\sum_{i=1}^{|\Delta_0|} p_0^{(i)}$ ou $\sum_{i=1}^{|\Delta_0|} p^{(i)}$ selon que $\boldsymbol{k} = \boldsymbol{k}^*$ ou non. Si on regarde maintenant ce que cela induit pour $D(\boldsymbol{k})$, on obtient que sa distribution est proche d'une loi de Poisson de paramètre $N_s p_0$ ou $N_s p$ où

$$p_0 \stackrel{\text{def}}{=} \frac{\sum_{i=1}^{|\Delta_0|} p_0^{(i)}}{|\Delta_0|} \text{ et } p \stackrel{\text{def}}{=} \frac{\sum_{i=1}^{|\Delta_0|} p^{(i)}}{|\Delta_0|}. \tag{9.1}$$

Le problème est que cette approximation de la distribution de $D(\boldsymbol{k})$ par une loi de Poisson s'avère ne pas être assez précise au niveau des queues de la distribution. Un outil utile lorsqu'il s'agit d'estimer les queues de la distribution d'une somme de variables i.i.d. est le résultat suivant que l'on peut trouver dans [Gal68] c'est une généralisation du résultat de [AG89] exposé dans le chapitre 6 qui est spécifique des variables de Bernoulli.

Théorème 9.6 ([Gal68] (5.4.23-5.4.24))**.** *Soit $D(\boldsymbol{k})$ une somme de N_s variables aléatoires, discrètes, indépendantes et identiquement distribuées. Soit $\mu(s)$ le logarithme de la fonction génératrice des moments des $D_{\boldsymbol{m}}(\boldsymbol{k})$. Alors,*

$$\Pr\left[D(\boldsymbol{k}) \geq \mu'(\eta) N_s\right] = e^{N_s[\mu(\eta)-\eta\mu'(\eta)]}\left[\frac{1}{|\eta|\sqrt{\pi 2 N_s \mu''(\eta)}} + o\left(\frac{1}{\sqrt{N_s}}\right)\right]. \tag{9.2}$$

On va utiliser ce résultat afin de dériver une formule précise des queues des distributions de $D(\boldsymbol{k}^*)$ et $D(\boldsymbol{k})$.

Théorème 9.7. *Soit $D(\boldsymbol{k}) = \sum_{\boldsymbol{m}} D_{\boldsymbol{m}}(\boldsymbol{k})$ une somme de N_s variables i.i.d. On définit deux fonctions bivariées avec τ et q des réels de $[0;1]$ tels que $\tau \neq q$.*

$$G_-(\tau, q) \stackrel{\text{def}}{=} e^{-N_s D(\tau||q)} \cdot \left[\frac{q\sqrt{(1-\tau)}}{(q-\tau)\sqrt{2\pi\tau N_s}} + \frac{1}{\sqrt{8\pi\tau N_s}}\right], \tag{9.3}$$

$$G_+(\tau, q) \stackrel{\text{def}}{=} e^{-N_s D(\tau||q)} \cdot \left[\frac{(1-q)\sqrt{\tau}}{(\tau-q)\sqrt{2\pi N_s(1-\tau)}} + \frac{1}{\sqrt{8\pi\tau N_s}}\right]. \tag{9.4}$$

Alors, les queues de la fonction de répartition de $D(\boldsymbol{k})$ peuvent être approximées par

$$\begin{aligned}\Pr\left[D(\boldsymbol{k}) \leq \tau N_s\right] &= G_-(\tau, p)\left[1 + o\left(\frac{p-\tau}{p}\right)\right],\\ \Pr\left[D(\boldsymbol{k}) \geq \tau N_s\right] &= G_+(\tau, p)\left[1 + o\left(\frac{p-\tau}{p}\right)\right].\end{aligned}$$

Ce résultat est aussi valable pour $D(\boldsymbol{k}^*)$, on remplacera alors p par p_0.

Éléments de preuve : La preuve de ce théorème est une application directe de (9.2) à notre cas. Celle-ci est calculatoire et se trouve donc à l'annexe B.4. □

On va donc définir la fonction suivante qui est une bonne approximation de la fonction de répartition de $D(\boldsymbol{k})$.

Définition 9.8. On note $G_p(\tau, q)$ la fonction de répartition d'une variable suivant une distribution de Poisson de paramètre $N_S\, q$. Soit

$$G(\tau, q) \stackrel{\text{def}}{=} \begin{cases} G_-(\tau, q) & \text{si } \tau < N_s q - 3 \cdot \sqrt{N_s q},\\ G_+(\tau, q) & \text{si } \tau > N_s q + 3 \cdot \sqrt{N_s q},\\ G_p(\tau, q) & \text{sinon.}\end{cases}$$

On définit, alors, $G_0(\tau)$ et $G(\tau)$ les fonctions de répartition des compteurs $D(\boldsymbol{k}^*)$ et $D(\boldsymbol{k})$:

$$G_0(\tau) \stackrel{\text{def}}{=} G(\tau, p_0) \quad \text{et} \quad G(\tau) \stackrel{\text{def}}{=} G(\tau, p).$$

La valeur $\sqrt{N_s q}$ est une estimation de l'écart type de la variable et la valeur 3 est choisie arbitrairement : elle est suffisamment grande pour que l'approximation des queues soit valide et suffisamment petite pour que l'estimation par une loi de Poisson soit valide sur leurs intervalles respectifs.

Complexité en données et probabilité de succès. Le théorème 9.7 montre que les queues des distributions de $D(\boldsymbol{k})$ et $D(\boldsymbol{k}^*)$ sont proches des formules présentées dans le chapitre 6 pour estimer la distribution binomiale. C'est pourquoi on va pouvoir utiliser les mêmes outils pour le calcul de la complexité en données et de la probabilité de succès de la cryptanalyse différentielle multiple.

Corollaire 9.9. *Si on effectue une cryptanalyse différentielle multiple ayant pour paramètres p_0 et p définis par (9.1) et gardant une liste de taille ℓ parmi 2^{n_k} candidats avec un nombre de couples clair/chiffré*

$$N = -2 \cdot \frac{\ln(2\sqrt{\pi}\ell\, 2^{-n_k})}{|\Delta_0|\, D(p_0||p)},$$

alors on obtient une probabilité de succès proche de 0,5.

Éléments de preuve : En utilisant les approximations données dans la définition 9.8, et avec une preuve similaire à celle du théorème 6.13, on obtient l'estimation $N_s = -\frac{\ln(2\sqrt{\pi}\ell\, 2^{-n_k})}{D(p_0||p)}$. Il reste ensuite à injecter ce résultat dans l'égalité $N = \frac{2N_s}{|\Delta_0|}$. □

En ce qui concerne la probabilité de succès, on peut aussi utiliser les résultats du chapitre 6.

Corollaire 9.10. *Soit une cryptanalyse différentielle multiple telle que le nombre de candidats gardés ℓ est inférieur au quart du nombre total n de candidats. On note p_0 et p les deux probabilités considérées. En utilisant les fonctions définies dans la définition 9.8, une bonne estimation de la probabilité de succès est alors,*

$$P_S \approx 1 - G_0\left[G^{-1}\left(1 - \frac{\ell - 1}{n - 2}\right) - 1\right]. \tag{9.5}$$

Remarque sur la non utilisation de (5.12). La formule donnée par (5.12) est valable dans le cas d'une cryptanalyse différentielle simple et prend en compte le fait que pour une clef fixée, la probabilité d'une différentielle suit une loi binomiale autour de sa probabilité en moyenne sur les clefs. Le fait d'utiliser un grand nombre de différentielles va atténuer cet effet. On verra d'ailleurs que sans prendre en compte ce phénomène on obtient de très bons résultats.

9.2.2 Quelques mots sur la complexité en temps et en mémoire

Avant d'aller plus loin intéressons-nous aux autres complexités de cette attaque. On va pour cela étudier chacune de ces étapes, l'algorithme 15 résume le déroulement d'une cryptanalyse différentielle multiple.

La première étape est de former les paires potentiellement intéressantes. On a vu dans la sous-section 9.1.2 que l'on utilisait pour cela un crible. Un crible efficace est de générer, pour chaque ensemble de différences $\Delta_r^{(i)}$ correspondant à une différence en entrée, l'ensemble

$$\Delta_{r+1}^{(i)} \stackrel{\text{def}}{=} \{\delta, \exists \delta_r \in \Delta_r^{(i)}, \Pr_{\boldsymbol{P},\boldsymbol{K}}\left[F_{\boldsymbol{K}}(\boldsymbol{P}) \oplus F_{\boldsymbol{K}}(\boldsymbol{P} \oplus \delta_r) = \delta\right] \neq 0\}.$$

Cet ensemble contient donc toute les différences pouvant être atteintes par un couple ayant une différence dans $\Delta_r^{(i)}$. Passer une paire au crible revient donc à rechercher sa différence dans cet ensemble, ce qui, si celui-ci est sous forme d'arbre binaire de recherche, prend un temps logarithmique en $\left|\Delta_{r+1}^{(i)}\right|$. Il faudra penser à prendre en compte le stockage des $\left|\Delta_{r+1}^{(i)}\right|$ dans la complexité en mémoire. Cette étape peut être rendue plus efficace encore en utilisant l'astuce répandue de la table de hachage présentée en sous-section 9.1.2. Quand cette astuce est appliquée, la complexité en temps de la génération des paires devient négligeable par rapport au reste de l'algorithme. C'est pourquoi, par souci de clarté, et comme il est très peu probable de pouvoir monter une attaque pour laquelle cette astuce ne serait pas applicable, on ne prendra pas en compte cette étape dans le calcul de la complexité en temps de l'attaque.

Une fois les paires générées, il faut effectuer le déchiffrement partiel et tester la valeur de la différence obtenue. Si on note p_{crible} la probabilité qu'une paire passe le crible, on doit traiter $N_s\, p_{crible}$ paires à cette étape. Le déchiffrement partiel doit s'effectuer pour les 2^{n_k} candidats et l'on doit tester l'appartenance des $2^{n_k} N_s\, p_{crible}$ différences à l'un des $\Delta_r^{(i)}$. On note $S_r = \max_i \left|\Delta_r^{(i)}\right|$, la complexité en nombre de comparaisons est donc de l'ordre de $2^{n_k} N_s\, p_{crible} \log_2(1 + S_r)$.

La génération de la liste pouvant se faire en un temps linéaire en 2^{n_k}, cette étape a une complexité négligeable comparée à la précédente et on néglige celle-ci dans le calcul de la complexité.

Enfin, la dernière étape consiste à tester les $\ell 2^{k-n_k}$ clefs maîtres correspondant à chaque candidat gardé. Cela s'effectue en vérifiant que le résultat du chiffrement avec une clef maître de l'un des messages correspond au chiffré connu.

Algorithme 15 : Cryptanalyse différentielle multiple.

Entrées : N clairs choisis et les chiffrés correspondant.
Sorties : La clef $\boldsymbol{k}^*$ utilisée pour chiffrer les échantillons.
Initialiser une table C de 2^{n_k} compteurs à 0.
pour $1 \leq i \leq |\Delta_0|$ **faire**
  pour chaque *couple* $(\boldsymbol{m}, \boldsymbol{m} \oplus \delta_0^{(i)})$ **faire**
    si $E_{\boldsymbol{k}^*}(\boldsymbol{m}) \oplus E_{\boldsymbol{k}^*}(\boldsymbol{m} \oplus \delta_0^{(i)}) \in \Delta_{r+1}^{(i)}$ **alors**
      pour chaque candidat $\boldsymbol{k}$ **faire**
        Calculer $\delta = F_{\boldsymbol{k}}^{-1}(E_{\boldsymbol{k}^*}(\boldsymbol{m})) \oplus F_{\boldsymbol{k}}^{-1}(E_{\boldsymbol{k}^*}(\boldsymbol{m} \oplus \delta_0^{(i)}))$;
        si $\delta \in \Delta_r^{(i)}$ **alors** $C[\boldsymbol{k}] \leftarrow C[\boldsymbol{k}] + 1$;
      fin
    fin
  fin
fin
Générer la liste $\mathcal{L}$ des ℓ candidats ayant les compteurs $C[\boldsymbol{k}]$ les plus élevés;
pour chaque $\boldsymbol{k} \in \mathcal{L}$ **faire**
  pour chaque clef maître possible $\boldsymbol{k}_m$ correspondant à $\boldsymbol{k}$ **faire**
    si $E_{\boldsymbol{k}_m}(\boldsymbol{m}) = E_{\boldsymbol{k}^*}(\boldsymbol{m})$ **alors retourner** $\boldsymbol{k}_m$;
  fin
fin

Au final, on obtient le tableau 9.1 qui récapitule les termes importants de la complexité en temps de l'attaque.

Chiffrements	Déchiffrements partiels	Comparaisons
$\mathcal{O}\left(\ell 2^{k-n_k}\right)$	$\mathcal{O}\left(2^{n_k} N_s p_{crible}\right)$	$\mathcal{O}\left(2^{n_k} N_s p_{crible} \log_2(1+S_r)\right)$

TABLE 9.1 – Complexité en temps d'une cryptanalyse différentielle.

En ce qui concerne la complexité en mémoire, elle est essentiellement due au stockage des cribles et des compteurs et est donc de l'ordre de

$$2^{n_k} + \sum_{i=1}^{|\Delta_0|} \left|\Delta_{r+1}^{(i)}\right|.$$

9.2.3 Application à SMALLPRESENT-[8] et PRESENT

Afin de valider le modèle utilisé et la précision de la formule (9.5), une cryptanalyse différentielle multiple sur 11 tours de SMALLPRESENT-[8] a été effectuée. Les 55 différentielles sur 9 tours utilisées ainsi que leurs probabilités se trouvent dans le tableau 9.2. Trois probabilités sont données pour chacune de ces différentielles. La première est la probabilité estimée théoriquement en prenant un grand nombre de chemins différentiels. La seconde et la troisième sont des probabilités estimées empiriquement en chiffrant tous les messages avec un grand nombre de clefs. Les deux valeurs correspondent à deux versions de SMALLPRESENT-[8],

0x3 → 0x40400000	$2^{-30.28}$	$2^{-29.80}$	$2^{-29.85}$
0x3 → 0x04040000	$2^{-30.33}$	$2^{-29.80}$	$2^{-29.84}$
0x3 → 0x50500000	$2^{-30.46}$	$2^{-29.96}$	$2^{-30.07}$
0x3 → 0x05050000	$2^{-30.58}$	$2^{-29.98}$	$2^{-29.99}$
0x3 → 0x10100000	$2^{-30.59}$	$2^{-29.90}$	$2^{-30.10}$
0x3 → 0x01010000	$2^{-30.64}$	$2^{-29.94}$	$2^{-30.45}$
0x3 → 0x80800000	$2^{-30.70}$	$2^{-30.17}$	$2^{-30.24}$
0x3 → 0x08080000	$2^{-30.70}$	$2^{-30.10}$	$2^{-30.01}$
0x3 → 0x0a0a0000	$2^{-30.97}$	$2^{-30.27}$	$2^{-30.32}$
0x7 → 0x40400000	$2^{-29.47}$	$2^{-29.20}$	$2^{-29.21}$
0x7 → 0x04040000	$2^{-29.54}$	$2^{-29.23}$	$2^{-23.23}$
0x7 → 0x50500000	$2^{-29.59}$	$2^{-29.26}$	$2^{-29.30}$
0x7 → 0x10100000	$2^{-29.74}$	$2^{-29.33}$	$2^{-29.70}$
0x7 → 0x05050000	$2^{-29.76}$	$2^{-29.37}$	$2^{-29.43}$
0x7 → 0x01010000	$2^{-29.86}$	$2^{-29.54}$	$2^{-29.56}$
0x7 → 0x0a0a0000	$2^{-30.00}$	$2^{-29.63}$	$2^{-29.65}$
0x7 → 0x80800000	$2^{-30.19}$	$2^{-29.61}$	$2^{-29.72}$
0x7 → 0x08080000	$2^{-30.21}$	$2^{-29.66}$	$2^{-29.66}$
0x7 → 0x40500000	$2^{-30.76}$	$2^{-30.22}$	$2^{-30.09}$
0xD → 0x05050000	$2^{-29.81}$	$2^{-29.30}$	$2^{-29.39}$
0xD → 0x40400000	$2^{-29.82}$	$2^{-29.42}$	$2^{-29.42}$
0xD → 0x04040000	$2^{-29.91}$	$2^{-29.50}$	$2^{-29.46}$
0xD → 0x10100000	$2^{-30.01}$	$2^{-29.50}$	$2^{-29.83}$
0xD → 0x50500000	$2^{-30.08}$	$2^{-29.60}$	$2^{-29.71}$
0xD → 0x01010000	$2^{-30.15}$	$2^{-29.52}$	$2^{-30.14}$
0xD → 0x0a0a0000	$2^{-30.25}$	$2^{-29.74}$	$2^{-29.78}$
0xD → 0x80800000	$2^{-30.39}$	$2^{-29.82}$	$2^{-29.96}$

0x5 → 0x40400000	$2^{-30.20}$	$2^{-29.76}$	$2^{-29.80}$
0x5 → 0x04040000	$2^{-30.25}$	$2^{-29.87}$	$2^{-29.73}$
0x5 → 0x50500000	$2^{-30.34}$	$2^{-29.87}$	$2^{-29.76}$
0x5 → 0x10100000	$2^{-30.50}$	$2^{-30.06}$	$2^{-30.28}$
0x5 → 0x05050000	$2^{-30.52}$	$2^{-30.02}$	$2^{-30.06}$
0x5 → 0x01010000	$2^{-30.55}$	$2^{-29.96}$	$2^{-29.94}$
0x5 → 0x08080000	$2^{-30.57}$	$2^{-30.01}$	$2^{-29.97}$
0x5 → 0x80800000	$2^{-30.57}$	$2^{-29.98}$	$2^{-30.04}$
0x5 → 0x0a0a0000	$2^{-30.77}$	$2^{-30.08}$	$2^{-30.04}$
0xB → 0x40400000	$2^{-30.21}$	$2^{-29.60}$	$2^{-29.88}$
0xB → 0x04040000	$2^{-30.26}$	$2^{-29.75}$	$2^{-29.92}$
0xB → 0x50500000	$2^{-30.41}$	$2^{-29.96}$	$2^{-29.99}$
0xB → 0x05050000	$2^{-30.59}$	$2^{-29.97}$	$2^{-30.06}$
0xB → 0x08080000	$2^{-30.64}$	$2^{-29.94}$	$2^{-30.02}$
0xB → 0x80800000	$2^{-30.65}$	$2^{-29.95}$	$2^{-30.06}$
0xB → 0x10100000	$2^{-30.73}$	$2^{-30.13}$	$2^{-30.33}$
0xB → 0x01010000	$2^{-30.81}$	$2^{-30.13}$	$2^{-30.18}$
0xB → 0x0a0a0000	$2^{-30.86}$	$2^{-30.09}$	$2^{-30.10}$
0xF → 0x40400000	$2^{-29.49}$	$2^{-29.26}$	$2^{-29.36}$
0xF → 0x04040000	$2^{-29.56}$	$2^{-29.23}$	$2^{-29.31}$
0xF → 0x50500000	$2^{-29.80}$	$2^{-29.46}$	$2^{-29.45}$
0xF → 0x05050000	$2^{-29.82}$	$2^{-29.39}$	$2^{-29.37}$
0xF → 0x80800000	$2^{-29.88}$	$2^{-29.32}$	$2^{-29.37}$
0xF → 0x08080000	$2^{-29.88}$	$2^{-29.58}$	$2^{-29.38}$
0xF → 0x10100000	$2^{-30.10}$	$2^{-29.69}$	$2^{-29.76}$
0xF → 0x01010000	$2^{-30.16}$	$2^{-29.68}$	$2^{-29.94}$
0xF → 0x0a0a0000	$2^{-30.22}$	$2^{-29.67}$	$2^{-29.8}$
0xF → 0x00110000	$2^{-30.60}$	$2^{-29.97}$	$2^{-29.78}$

TABLE 9.2 – Les 55 différentielles utilisées pour la cryptanalyse de SMALLPRESENT-[8].

la première utilise l'algorithme de cadencement de clefs utilisant une clef de 40 bits présenté dans la sous-section 2.2.2 et la seconde correspond à la version originellement proposée par Leander [Lea10] qui utilise le même algorithme que PRESENT et donc une clef de 80 bits. Les paramètres suivants ont été utilisés pour l'attaque

$$\ell = 2^{12} \quad \text{et} \quad n = 2^{32}.$$

La figure 9.1 représente la probabilité de succès de l'attaque en fonction du nombre de clairs disponibles N. Une des courbes est la probabilité expérimentale obtenue avec 200 attaques. Les trois autres sont les probabilités théoriques obtenues en utilisant

- une approximation gaussienne : c'est la formule de Selçuk (5.5) [1] ;
- une approximation de Poisson : on utilise alors le théorème 6.18 ;
- l'approximation donnée par (9.5) : c'est le théorème 9.10.

Les résultats expérimentaux, obtenus en effectuant 200 cryptanalyses chacun, confirme la précision de la formule donnée pour la probabilité de succès d'une cryptanalyse différentielle multiple. Notons que les courbes ont été tracées en utilisant les probabilités déterminées empiriquement des différentielles et non les probabilités estimées théoriquement.

1. C'est la technique utilisée actuellement pour calculer cette probabilité de succès.

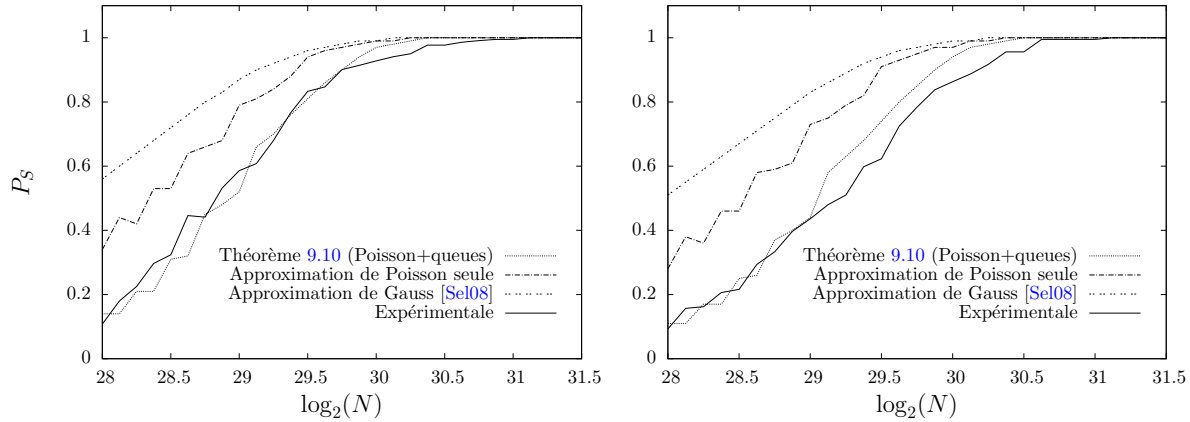

FIGURE 9.1 – Probabilité de succès de l'attaque sur SMALLPRESENT-[8] en fonction de N (clef de 40 bits à gauche et 80 bits à droite).

Les valeurs de N obtenues en utilisant le corollaire 9.9 ainsi que les valeurs empiriques des probabilités de succès correspondantes sont données dans le tableau 9.3.

c	version	N	P_S
1.0	40 bits	$2^{28.90}$	0.53
	80 bits	$2^{29.10}$	0.47
1.5	40 bits	$2^{29.50}$	0.82
	80 bits	$2^{29.70}$	0.74
2.0	40 bits	$2^{29.90}$	0.91
	80 bits	$2^{30.10}$	0.89

TABLE 9.3 – Valeurs de N obtenues avec le corollaire 9.9 et probabilités de succès associées.

Attaque sur 18 tours de PRESENT. Fort de ces résultats expérimentaux, on peut proposer une cryptanalyse différentielle multiple sur 18 tours du chiffrement PRESENT. Pour cela, on utilise des différentielles sur 16 tours de la forme

$$(\texttt{0x000000000000?00?}, \texttt{0x????????00000000}).$$

On a donc des structures de taille 2^8. La nombre de différences en entrée est $|\Delta_0| = 17$ et pour chacune d'elle on a 33 différences en sortie. Les probabilités en jeu sont

$$p_0 = 2^{-58.50} \quad \text{et} \quad p = 2^{-58.96}.$$

On a $\Delta_0 = \{\texttt{0x1001}\} \cup \{\texttt{0xZ00Z}\}$ où $\texttt{Z} \in \{\texttt{0x2}, \texttt{0x4}, \texttt{0xA}, \texttt{0xC}\}$. On a donc bien un ensemble admissible et notre théorie peut donc s'appliquer à l'attaque. La taille du crible après 2 tours est d'un peu moins de 2^{32} différences et 52 bits de clef sont impliqués dans le déchiffrement partiel.

On va maintenant proposer deux jeux de paramètres pour l'attaque. La première utilise tous les couples clair/chiffré possibles pour diminuer au maximum la complexité en temps quand la seconde ne retrouve qu'un bit d'information mais utilise le moins de données possible. Ces

ℓ	N	P_S	Complexité en temps [2]
$2^{36.0}$	$2^{64.0}$	$0,94$	$2^{71.7}$
$2^{47.0}$	$2^{62.0}$	$0,81$	$2^{75.0}$
$2^{51.0}$	$2^{60.0}$	$0,76$	$2^{79.0}$

TABLE 9.4 – Complexités des attaques proposées sur 18 tours de PRESENT.

attaques ne sont pas les meilleures sur PRESENT mais elles permettent d'améliorer de 2 tours la meilleure cryptanalyse différentielle actuelle [Wan08].

Discussion. Nous proposons donc la meilleure cryptanalyse différentielle multiple sur le chiffrement PRESENT à ce jour. Cependant, une telle attaque n'est pas optimale. En effet, on perd beaucoup d'information en sommant les compteurs correspondant aux différentielles plutôt que de les considérer individuellement. Le but premier était de donner les outils pour analyser proprement les cryptanalyses différentielles multiples de la littérature. Maintenant que cela est fait, une piste de travail intéressante est de regarder comment traiter de façon optimale les compteurs obtenus. Cela impliquera potentiellement un certain nombre de complications comme le problème d'indépendance de ces compteurs ainsi que leur liens avec la clef utilisée pour chiffrer les échantillons (phénomène estompé lorsque l'on somme les compteurs).

2. Voir sous-section 9.2.2 pour plus de détails. À noter ici que l'on effectue un déchiffrement partiel sur deux tours et que donc l'on peut utiliser un crible intermédiaire afin de ne pas tester les 2^{28} candidats pour des paires qui peuvent déjà être détectées comme fausses après un tour. Cette astuce utilisée aussi dans [Wan08] permet de diminuer drastiquement la complexité de l'étape consistant à déchiffrer.

Conclusions

Les résultats présentés dans ce document répondent à plusieurs problématiques mais soulèvent aussi beaucoup de questions et donc de perspectives de recherche.

Approfondir les sujets traités. Tout d'abord, il y a plusieurs pistes à explorer pour prolonger et approfondir certains travaux présentés ici.

En particulier pour ce qui est de la cryptanalyse linéaire multiple. Outre la prise en compte de l'effet hull qui risque de monopoliser l'attention d'une partie de la communauté pour les années à venir, l'approfondissement de l'utilisation d'algorithmes de décodage paraît être une piste intéressante. En effet, il n'existe aucun résultat sur l'algorithme de décodage par résonance stochastique ce qui serait d'une grande utilité pour choisir les bons paramètres et évaluer ses performances. De plus, on pourrait imaginer utiliser d'autres types algorithmes comme le décodage itératif par exemple.

En ce qui concerne les résultats sur l'entropie, il est clair que la première chose à faire est d'appliquer cette approche pour des types de cryptanalyse autres que la cryptanalyse linéaire. L'exemple de la cryptanalyse différentielle multiple paraît, dans ce cas, être un bon candidat.

La cryptanalyse différentielle multiple est, d'ailleurs, aussi une source de questions et entre autre, gagnerait-on à ne pas additionner tous les compteurs ? Cette question dont la réponse peut, à première vue, sembler évidente, est en fait assez complexe. En effet, en additionnant tous ces compteurs on perd de l'information, cependant, comme la probabilité d'une différentielle est fortement dépendante de la clef utilisée, il est loin d'être clair qu'en gardant les compteurs correspondant à chaque différentielle (voire chaque chemin différentiel), l'on arrive à extraire significativement plus d'information sur la clef qu'en faisant la somme de tous ces compteurs.

Explorer d'autres contrées. Bien que les résultats des chapitres 6 et 8 ait été présentés dans le cadre des cryptanalyses statistiques des chiffrements itératifs par blocs, certains sont suffisamment généraux pour pouvoir s'appliquer dans d'autres contextes. On pense aux cryptanalyses d'autres chiffrements symétriques mais aussi aux attaques par canaux auxiliaires où il est question de distinguer des distributions en ayant accès à un nombre limité d'échantillons.

De plus, on a vu qu'une technique de recherche d'approximations linéaires issue de la théorie des codes correcteurs trouvait sa principale utilité dans le contexte des attaques physiques et on peut se demander si d'autres techniques utilisées pour les codes correcteurs ne pourraient pas être la solution à certains problèmes soulevés dans ce domaine.

Annexe A

Quelques éléments de statistiques

Sommaire

Cette annexe a pour objectif de rappeler ou présenter les résultats de théorie des probabilités et statistiques utilisés dans ce document.

A.1 Quelques distributions utiles

Cette section a pour but de rappeler quelques éléments sur les distributions utilisées dans ce document.

A.1.1 Loi uniforme continue

La distribution uniforme continue est, comme son nom l'indique, continue.

Définition A.1. $\mathcal{U}([a;b])$.
Soit X une variable définie sur un intervalle $[a;b]$ et suivant une loi uniforme. Alors, la densité $f(x)$ de cette variable est définie sur $[a;b]$ par

$$f(x) \stackrel{\text{def}}{=} \frac{1}{b-a}.$$

Moments. Si $X \sim \mathcal{U}([a;b])$, alors
- $\mathbb{E}(X) = \frac{a+b}{2}$;
- $\mathbb{V}(X) = \frac{(b-a)^2}{12}$.

Génération d'autres lois continues à partir de la loi uniforme.
notons que l'on peut générer n'importe qu'elle loi continue à partir de celle-ci. On va, pour cela, définir la pseudo-inverse d'une fonction de répartition.

Définition A.2. Soit un ensemble $\mathcal{X}$ muni d'une relation d'ordre. Soit X une variable aléatoire définie sur $\mathcal{X}$ et ayant pour fonction de répartition $G(x)$. On notera G^{-1} la pseudo-inverse de la fonction de répartition [1] F.

$$G^{-1}(y) \stackrel{\text{def}}{=} \inf\{x \in \mathcal{X},\ G(x) \geq y\}.$$

Bien que $G^{-1}(G(x))$ puisse être différent de x, et que $G(G^{-1}(y))$ puisse ne pas valoir y, on a $G(G^{-1}(G(x))) = G(x)$.

L'on a évoqué le fait que l'on pouvait générer une distribution continue quelconque à partir de la distribution uniforme continue. Ceci est précisé dans la proposition suivante.

Proposition A.3. *Soit une variable U suivant la loi uniforme continue sur l'intervalle $[0;1]$ et G une fonction de répartition. La variable $X \stackrel{\text{def}}{=} G^{-1}(U)$ est alors distribuée selon la loi ayant G pour fonction de répartition.*

Démonstration : On a $\Pr[X \leq x] = \Pr\left[G^{-1}(U) \leq x\right]$. Moralement, on a envie d'appliquer G de part et d'autre de l'inégalité et simplifier le terme de gauche en U. On va voir que l'on peut obtenir ce résultat en revenant à la définition de G^{-1} : $G^{-1}(U) \leq x \Leftrightarrow \inf\{t, G(t) \geq U\} \leq x$. Comme la fonction G est croissante, on a l'équivalence $\inf\{t, G(t) \geq U\} \leq x \Leftrightarrow U \leq G(x)$. Donc, $\Pr[X \leq x] = \Pr[U \leq G(x)] = G(x)$. □

A.1.2 Loi de Bernoulli

La distribution de Bernoulli est discrète.

Définition A.4. Bernoulli(p).
Soit X une variable suivant une loi de Bernoulli de paramètre $p \in [0;1]$. Alors,

$$\Pr[X = 1] = p \quad \text{et} \quad \Pr[X = 0] = 1 - p.$$

Moments. Si $X \sim$Bernoulli(p), alors
- $\mathbb{E}(X) = p$;
- $\mathbb{V}(X) = p(1-p)$.

1. Si G est inversible alors sa pseudo-inverse est exactement son inverse. On se permet donc cet abus de notation.

A.1.3 Loi binomiale

La distribution binomiale est une distribution discrète.

Définition A.5. $\mathcal{B}(N,p)$.
Soit X une variable suivant une loi binomiale de paramètres $N \in \mathbb{N}$ et $p \in [0;1]$. Alors,

$$\Pr[X=i] \stackrel{\text{def}}{=} \binom{N}{i} p^i (1-p)^{N-i}.$$

Moments. Si $X \sim \mathcal{B}(N,p)$, alors
- $\mathbb{E}(X) = Np$;
- $\mathbb{V}(X) = Np(1-p)$.

A.1.4 Loi hypergéométrique

La distribution hypergéométrique est une distribution discrète.

Définition A.6. $\mathrm{Hyp}(N,p,A)$.
Soit X une variable suivant une loi hypergéométrique de paramètres $A \in \mathbb{N}$, $p \in [0;1]$ et $N \in \{0,1,\ldots,A\}$ tels que pA soit entier. Alors, pour $0 \leq i \leq N$,

$$\Pr[X=i] \stackrel{\text{def}}{=} \frac{\binom{pA}{i} \cdot \binom{(1-p)A}{N-i}}{\binom{A}{N}}.$$

Moments. Si $X \sim \mathcal{B}(N,p)$, alors
- $\mathbb{E}(X) = Np$;
- $\mathbb{V}(X) = Np(1-p)\frac{A-N}{A-1}$.

Propriété.
Pour $N = 1$ on a une variable de Bernoulli de paramètre 1. Il semble naturel alors d'approximer la loi hypergéométrique par une loi binomiale $\mathcal{B}(N,p)$ pour $N \ll A$.

A.1.5 Loi de Poisson

La distribution de Poisson est une distribution discrète.

Définition A.7. $\mathcal{P}(\lambda)$.
Soit X une variable suivant une loi de Poisson de paramètre $\lambda \in \mathbb{R}_*^+$. Alors,

$$\Pr[X=i] \stackrel{\text{def}}{=} \frac{\lambda^i e^{-\lambda}}{i!}.$$

Moments. Si $X \sim \mathcal{P}(\lambda)$, alors
- $\mathbb{E}(X) = \lambda$;
- $\mathbb{V}(X) = \lambda$.

Propriétés.
La loi de Poisson permet d'estimer la distribution d'une somme de variables de Bernoulli ayant des paramètres différents. Cette estimation est d'autant meilleure que ces paramètres sont petits.

Théorème A.8. *Inégalité de Le Cam.*
Soient $X_1, \ldots, X_n$ des variables de Bernoulli indépendantes et de paramètre respectif p_j. On s'intéresse à la somme $S \stackrel{\text{def}}{=} \sum_{j=-1}^{n} X_j$ de ces variables et on note λ l'espérance de cette somme $\lambda \stackrel{\text{def}}{=} \mathbb{E}(S) = \sum_{j=1}^{n} p_j$. Alors, pour un ensemble d'entiers positifs $\mathcal{A} \subset \{0, 1, \ldots, n\}$,

$$\left| \Pr[S \in \mathcal{A}] - \sum_{j \in \mathcal{A}} \frac{\lambda^j e^{-\lambda}}{j!} \right| \leq \sum_{j=1}^{n} p_j^2.$$

A.1.6 Loi normale (ou loi de Gauss)

La distribution de Gauss ou loi normale est une distribution continue.

Définition A.9. $\mathcal{N}(\mu, \sigma^2)$.
Soit X une variable suivant une loi normale de paramètres $\mu \in \mathbb{R}$ et $\sigma^2 \in \mathbb{R}^+$. Alors, la densité $f(x)$ de la variable X est définie sur $\mathbb{R}$ par

$$f(x) \stackrel{\text{def}}{=} \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

Moments. Si $X \sim \mathcal{N}(\mu, \sigma^2)$, alors
- $\mathbb{E}(X) = \mu$;
- $\mathbb{V}(X) = \sigma^2$.

Propriétés et notations.
Soient deux variables indépendantes $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ et $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$, alors,

$$X_1 + X_2 \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2).$$

De plus, si $X \sim \mathcal{N}(\mu, \sigma^2)$, alors, pour tout réel λ,

$$\lambda X \sim \mathcal{N}(\lambda\mu, \lambda^2\sigma^2).$$

La fonction de répartition de la loi normale, ainsi que son inverse, sont difficiles à calculer et sont pourtant très utilisées. On les a donc mis en table en utilisant les deux remarques suivantes.

1. Si $X \sim \mathcal{N}(\mu, \sigma^2)$, alors $\frac{X-\mu}{\sigma} \sim \mathcal{N}(0, 1)$. On a donc uniquement besoin de connaître les valeurs pour cette distribution précise car on peut toujours s'y ramener. On a d'ailleurs attribué la lettre grecque *phi* à cette distribution appelée *loi normale centrée réduite*. La densité de cette distribution est notée φ et sa fonction de répartition Φ. On note parfois φ_{μ,σ^2} et Φ_{μ,σ^2} la densité et la fonction de répartition de la loi normale de paramètres μ et σ^2.

2. La loi normale est symétrique par rapport à son espérance et on peut donc se contenter de la moitié du tableau : $\Phi(x) = 1 - \Phi(-x)$.

Au final, pour une variable $X \sim \mathcal{N}\left(\mu, \sigma^2\right)$,

$$\Pr\left[X \leq x\right] = \Pr\left[\frac{X-\mu}{\sigma} \leq \frac{x-\mu}{\sigma}\right].$$

Si $x - \mu$ est positif, alors on peut regarder dans la table, sinon, on utilise la seconde remarque et on calcule $1 - \Phi\left(-\frac{x-\mu}{\sigma}\right)$.

L'on a évoqué le fait que la loi normale était utilisée pour les phénomènes de masse. Ce phénomène est dû au théorème suivant qui est l'un des principaux théorèmes utilisés en théorie des probabilités.

Théorème A.10. *(Théorème central limite). Soit $X_1, X_2, \ldots$ une suite de variables aléatoires i.i.d. On suppose, de plus, que l'espérance $\mu \stackrel{\text{def}}{=} \mathbb{E}\left(X_1\right)$ et l'écart-type $\sigma \stackrel{\text{def}}{=} \sqrt{\mathbb{V}\left(X_1\right)}$ de ces variables sont finis. On s'intéresse à la distribution de la variable*

$$S_n \stackrel{\text{def}}{=} X_1 + X_2 + \cdots + X_n.$$

Pour tout $x \in \mathbb{R}$, on a,

$$\left|\Pr\left[\frac{S_n - n\mu}{\sigma\sqrt{n}} \leq x\right] - \Phi(x)\right| \underset{n\to\infty}{\longrightarrow} 0.$$

On verra, dans le chapitre 6, qu'il faut manipuler ce théorème avec précaution car il se peut que la convergence soit relativement lente. Voici une version quantitative du théorème permettant de majorer la vitesse de convergence.

Théorème A.11. *(Berry-Esséen). Soient n variables $X_1, \ldots, X_n$ i.i.d. telles que $\mathbb{E}\left(X_1\right) = 0$. On note $\sigma^2 \stackrel{\text{def}}{=} \mathbb{E}\left(X_1^2\right)$, $\rho \stackrel{\text{def}}{=} \mathbb{E}\left(|X_1^3|\right)$ et note F_n la fonction de répartition de la variable S_n définie comme $S_n \stackrel{\text{def}}{=} \frac{\sum_{i=1}^n X_i}{\sigma\sqrt{n}}$. Alors, il existe une constante positive C telle que pour tout x et pour tout n,*

$$|F_n(x) - \Phi(x)| \leq \frac{C\,\rho}{\sigma^3\sqrt{n}}.$$

Remarque sur la valeur de C. La meilleure borne supérieure actuelle pour la constante C est de 0.7056 [She06].

A.1.7 Loi multinormale

La loi multinormale est une distribution continue.

Définition A.12. $\mathcal{N}\left(\boldsymbol{\mu}, \Sigma\right)$.
Soit $\boldsymbol{X}$ une variable de $\mathbb{R}^n$ suivant une loi multinormale de dimension n de paramètres $\mu \in \mathbb{R}^n$ et $\Sigma \in M_n(\mathbb{R})$ matrice carrée $n \times n$. Alors, la densité $f(x)$ de cette variable est définie sur $\mathbb{R}$ par

$$f(\boldsymbol{x}) \stackrel{\text{def}}{=} \frac{1}{(2\pi)^{n/2}\sqrt{|\Sigma|}} e^{-\frac{1}{2}t(\boldsymbol{x}-\boldsymbol{\mu})\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}.$$

Propriétés. On a les mêmes propriétés au niveau de la somme et de l'addition de deux variables suivant des loi multinormales que pour la loi normale.

Si la matrice Σ des covariances est diagonale, alors la variable $\boldsymbol{X}$ peut être décomposée en n variables indépendantes X_i suivant une loi normale $\mathcal{N}(\mu_i, \Sigma_{i,i})$.

A.1.8 Loi bêta

La loi bêta est une distribution continue.

Définition A.13. Beta(a, b).
Soit X une variable suivant une loi bêta de paramètres $a, b \in \mathbb{R}_*^+$. Alors, la densité $f(x)$ de cette variable est définie sur $[0; 1]$ par

$$f(x) \stackrel{\text{def}}{=} \frac{x^{a-1}(1-x)^{b-1}}{\int_0^1 t^{a-1}(1-t)^{b-1}dt}.$$

Moments. Si $X \sim \text{Beta}(a, b)$, alors
- $\mathbb{E}(X) = \frac{a}{a+b}$;
- $\mathbb{V}(X) = \frac{ab}{(a+b)^2(a+b+1)}$.

Propriétés et notations.
On va parler ici de la fonction de répartition d'une variable suivant une loi bêta. Pour cela on introduit la *fonction bêta incomplète*.

Définition A.14. Pour deux paramètres $a, b \in \mathbb{R}_*^+$, on définit la *fonction bêta incomplète* sur $[0; 1]$ par,

$$B_{a,b}(x) \stackrel{\text{def}}{=} \int_0^x t^{a-1}(1-t)^{b-1}dt.$$

La fonction de répartition d'une variable X suivant une la loi bêta est alors

$$F(x) \stackrel{\text{def}}{=} \frac{B_{a,b}(x)}{B_{a,b}(1)}.$$

Ce quotient $\frac{B_{a,b}(x)}{B_{a,b}(1)}$ est aussi appelé *fonction bêta incomplète régularisée*. Des intégrations par parties successives permettent de donner l'expression suivante pour des valeurs entières de a et b.

$$\frac{1}{B_{a,b}(1)} = (a+b-1) \cdot \binom{a+b-2}{a-1}. \tag{A.1}$$

A.2 Test binaire d'hypothèses

Une présentation plus complète de la problématique peut être trouvée dans [CT91, section 11.7]. Nous nous concentrerons ici sur les points importants dans notre contexte. Commençons par définir le contexte et la problématique générale des *tests binaires d'hypothèses*.

Définition A.15. (Test binaire d'hypothèses).
Soient $(x_1, \ldots, x_n) \in \mathcal{X}^n$ des réalisations de variables aléatoires i.i.d. distribuées selon une loi $\mathcal{Q} \in \{\mathcal{P}_0, \mathcal{P}_1\}$. La question consistant à choisir une des deux hypothèses

- $H_0 : \mathcal{Q} = \mathcal{P}_0$;
- $H_1 : \mathcal{Q} = \mathcal{P}_1$;

est nommé *test binaire d'hypothèses*. Une *fonction de décision* pour ce test d'hypothèse est une fonction g qui à n réalisations fait correspondre le numéro de l'hypothèse que l'on considérera valide :

$$g : \mathcal{X}^n \to \{0; 1\}.$$

Une façon de définir g est l'utilisation d'une *région d'acceptation*

$$A_n \stackrel{\text{def}}{=} \{(x_1, \ldots, x_n) \in \mathcal{X}^n,\ g(x_1, \ldots, x_n) = 0\}\,.$$

Lors d'un tel test, on peut se tromper de deux façons différentes :

- *erreur de type 1 :* H_1 est choisie alors que H_0 est vraie ;
- *erreur de type 2 :* H_0 est choisie alors que H_1 est vraie.

Le but, lorsque l'on souhaite effectuer un test statistique, est de définir une région d'acceptation qui mène à des probabilités d'erreur d'un ordre souhaité. Par exemple, si H_0 correspond à « individu malade » et H_1 à « individu sain », le principe de précaution privilégiera une probabilité d'erreur de type 1 très faible au risque de faire de fausses frayeurs à certains patients.

Le lemme de Neyman-Pearson (théorème A.16) donne la façon optimale de construire une région d'acceptation.

Théorème A.16. *(Lemme de Neyman-Pearson).*
Soient $X_1, \ldots, X_n$ des variables aléatoires i.i.d. suivant une distribution $\mathcal{Q}$. On considère le test d'hypothèse suivant

- *H_0 : $\mathcal{Q} = \mathcal{P}_0$;*
- *H_1 : $\mathcal{Q} = \mathcal{P}_1$.*

Pour un seuil $t \geq 0$, on définit une région d'acceptation

$$A_n(t) \stackrel{\text{def}}{=} \left\{(x_1, \ldots, x_n),\ \frac{\Pr_{\mathcal{P}_0}[x_1, \ldots, x_n]}{\Pr_{\mathcal{P}_1}[x_1, \ldots, x_n]} \geq t\right\}. \tag{A.2}$$

On note $\alpha(A_n(t))$ et $\beta(A_n(t))$ les probabilités d'erreur de premier et second type obtenues pour la région $A_n(t)$. Alors, pour toute autre région B_n ayant comme probabilités d'erreur associées $\alpha(B_n)$ et $\beta(B_n)$, si $\alpha(B_n) < \alpha(A_n(t))$, alors $\beta(B_n) > \beta(A_n(t))$.

Ce que dit ce résultat est que l'on ne peut pas trouver de région d'acceptation qui améliore les deux probabilités d'erreur par rapport à une région définie par un seuil sur le *quotient de vraisemblance*. De telles régions d'acceptation sont donc optimale en ce sens.

On va terminer cette section en s'intéressant au terme de *vraisemblances* que l'on vient juste d'évoquer. Dans le cadre des tests d'hypothèses, on cherche à valider une hypothèses parmi plusieurs au regard d'un ensemble de réalisations de variables aléatoires. La *vraisemblance* d'une hypothèse est alors la probabilité que celle-ci soit vraie sachant qu'on a obtenu un vecteur d'observations $\boldsymbol{x}$:

$$v(H) \stackrel{\text{def}}{=} \Pr[H|\boldsymbol{X} = \boldsymbol{x}]\,.$$

Si les hypothèses sont équiprobables, la formule de Bayes nous dit que l'hypothèse la plus vraisemblable maximise $\Pr[\boldsymbol{X} = \boldsymbol{x}|H]$. Comme ce sera toujours le cas dans ce document, on définit plutôt

$$v(H) \stackrel{\text{def}}{=} \Pr[\boldsymbol{X} = \boldsymbol{x}|H]\,.$$

Le *quotient de vraisemblance* utilisé dans la définition de la région d'acceptation dans le lemme de Neyman-Pearson est le quotient des vraisemblances des réalisations x_i selon que l'hypothèse H_0 ou H_1 soit considérée vraie. Si ce quotient est supérieur à 1, alors l'événement est plus vraisemblable si H_0 est vraie et vice-versa. Étant donné que les variables sont indépendantes, les probabilités jointes impliquées dans ce rapport peuvent être décomposées en un produit de probabilités invoquant une seule variable. Il peut donc être intéressant de regarder le logarithme de ce quotient de vraisemblance qui sera alors comparé à 0.

Définition A.17. Soient $X_1, \ldots, X_n$ des variables aléatoires indépendantes et identiquement distribuées selon une distribution $\mathcal{Q} \in \{\mathcal{P}_0, \mathcal{P}_1\}$ de support $\mathcal{X}$.

Soit $(x_1, \ldots, x_n)$ un ensemble constitué d'une réalisation de chaque variable. Alors, le logarithme du quotient de vraisemblance de ces réalisations est défini comme

$$\texttt{LLR}(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \ln\left(\frac{\Pr_{\mathcal{P}_0}[x_1, \ldots, x_n]}{\Pr_{\mathcal{P}_1}[x_1, \ldots, x_n]}\right).$$

Comme les variables sont indépendantes, on a aussi

$$\texttt{LLR}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \ln\left(\frac{\Pr_{\mathcal{P}_0}[x_i]}{\Pr_{\mathcal{P}_1}[x_i]}\right).$$

De plus, si on note $N_x \stackrel{\text{def}}{=} \#\{1 \leq i \leq n, x_i = x\}$, alors on a aussi

$$\texttt{LLR}(x_1, \ldots, x_n) = \sum_{x \in \mathcal{X}} N_x \ln\left(\frac{\Pr_{\mathcal{P}_0}[x_i]}{\Pr_{\mathcal{P}_1}[x_i]}\right).$$

Quand le nombre de variables augmente, on a une convergence du LLR vers une loi normale comme rappelé dans [BJV04]. Cette propriété sera utilisée pour l'analyse de la cryptanalyse linéaire.

Théorème A.18. *Soient $X_1, \ldots, X_n$ des variables aléatoires identiquement distribuées selon une distribution $\mathcal{Q} \in \{\mathcal{P}_0, \mathcal{P}_1\}$ de support $\mathcal{X}$ et indépendantes. On note $\boldsymbol{X}^n \stackrel{\text{def}}{=} (X_1, \ldots, X_n)$. Alors, si $\mathcal{Q} = \mathcal{P}_i$,*

$$\Pr\left[\frac{\texttt{LLR}(\boldsymbol{X}^n) - N\mu_i}{\sigma_i\sqrt{n}} < x\right] \underset{N\to\infty}{\longrightarrow} \Phi(x).$$

Avec $\mu_0 \stackrel{\text{def}}{=} D(\mathcal{P}_0||\mathcal{P}_1)$, $\mu_1 \stackrel{\text{def}}{=} -D(\mathcal{P}_1||\mathcal{P}_0)$ et

$$\sigma_i^2 \stackrel{\text{def}}{=} \sum_{\boldsymbol{x} \in \mathcal{X}^n} \Pr_{\mathcal{P}_i}[\boldsymbol{x}] \ln\left(\frac{\Pr_{\mathcal{P}_0}[\boldsymbol{x}]}{\Pr_{\mathcal{P}_1}[\boldsymbol{x}]}\right)^2 - \mu_i^2.$$

Démonstration : Ce résultat est obtenu en appliquant le théorème A.10. On verra, le moment venu, si la convergence est rapide ou non. □

A.3 Statistiques d'ordre

Nous allons maintenant donner quelques notions de bases des statistiques d'ordre qui peuvent être trouvées dans [DN03] par exemple.

Soient, de nouveau, $X_1, \ldots, X_n$, des variables i.i.d. On s'intéresse à l'ordre de ces variables. On trie ces variables par ordre croissant et on obtient n nouvelles variables $X_{(1)}, \ldots X_{(n)}$. On a donc, par exemple, $X_{(1)} = \min_i X_i$ et $X_{(n)} = \max_i X_i$. Ce qui va nous servir par la suite est l'expression des fonctions de répartition de ces variables $X_{(i)}$. On appelle $X_{(i)}$ la i-ème statistique d'ordre des $X_1, \ldots, X_n$.

Proposition A.19. *On définit n variables $X_1, \ldots, X_n$ i.i.d. et G leur fonction de répartition $G(x) \stackrel{\text{def}}{=} \Pr[X_i \leq x]$. Pour un entier $1 \leq r \leq n$, on définit $X_{(r)}$ la r-ième statistique d'ordre et l'on note $G_{(r)}$ sa fonction de répartition. On a alors,*

$$G_{(r)}(x) = \sum_{i=r}^{n} \binom{n}{i} G(x)^i [1 - G(x)]^{n-i}.$$

Démonstration :

$$\begin{aligned} G_{(r)}(x) &= \Pr\left[X_{(r)} \leq x\right] \\ &= \Pr[\text{au moins } r \text{ des } n \ X_i \text{ valent au plus } x] \\ &= \sum_{i=r}^{n} \binom{n}{i} G(x)^i [1 - G(x)]^{n-i}. \end{aligned}$$

□

Une remarque intéressante qui nous sera utile plus tard est la propriété suivante.

Proposition A.20. *Soient $U_1, \ldots, U_n$ de variables indépendantes suivant une loi uniforme sur l'intervalle $[0; 1]$. Alors la distribution de $U_{(r)}$ est une loi Beta$(r, n - r + 1)$.*

Démonstration : La preuve est simple et utilise la proposition A.19.

$$\begin{aligned} G_{(r)}(x) &= \sum_{i=r}^{n} \binom{n}{i} G(x)^i [1 - G(x)]^{n-i} \\ &= \sum_{i=r}^{n} \frac{n!}{i!(n-i)!} x^i [1 - x]^{n-i} \end{aligned}$$

On utilise ensuite la relation vue lors de la présentation de la loi bêta (A.1) qu'on identifie à notre cas. On rappelle que la fonction de répartition d'une variable suivant une loi bêta de paramètres a et b entiers peut s'écrire

$$\sum_{i=a}^{a+b-1} \frac{(a+b-1)!}{i!(a+b-1-i)!} x^i (1-x)^{a+b-1-i}.$$

Par identification, on obtient donc $a = r$ et $b = n - r + 1$. Et donc, $G_{(r)}(x)$ est la fonction de répartition de la loi bêta de paramètres $(r, n - r + 1)$. □

On déduit donc la caractérisation suivante de la fonction de répartition d'une statistique d'ordre.

Lemme A.21. *Soient $X_1, \ldots, X_n$ des variables i.i.d. suivant une distribution ayant G comme fonction de répartition. On note $X_{(1)}, \ldots, X_{(n)}$ les variables correspondant aux variables X_i triées par ordre croissant. Alors,*

$$\Pr\left[X_{(r)} \leq x\right] = B_{r,n-r+1}(G(x)).$$

Démonstration : La preuve est simple et repose sur le fait qu'une fonction de répartition est croissante. Si on note $G(X)_{(r)}$ la r-ième statistique d'ordre des $G(X_i)$, alors, la croissance de G implique $G(X)_{(r)} = G(X_{(r)})$. Or, selon la proposition A.20, comme les $G(X_i)$ suivent une loi uniforme, les $G(X)_{(r)}$ suivent une loi bêta de paramètres $(r, n-r+1)$ et donc

$$\begin{aligned}\Pr\left[X_{(r)} \leq x\right] &= \Pr\left[G(X_{(r)}) \leq G(x)\right] \\ &= \Pr\left[G(X)_{(r)} \leq G(x)\right] \\ &= B_{r,n-r+1}(G(x)).\end{aligned}$$

□

Annexe B

Quelques lemmes utilisés

Sommaire

Cet annexe a pour but de regrouper les preuves relativement longues et assez calculatoires qui risqueraient de perdre le lecteur si elles étaient placées directement après les énoncés correspondants.

B.1 Divergence de Kullback-Leibler

On prouve ici les lemmes relatifs à la dérivation de formules pour approximer la divergence de Kullback-Leibler dans plusieurs cas particuliers.

Lemme 6.16. *Soient* $0 < b < a < 1$ *tels que* $\frac{a-b}{1-b} = \mathcal{O}(a-b)$. *Alors,*

$$D(a|\,|b) = a \cdot \left[\ln\left(\frac{a}{b}\right) - \frac{a-b}{a} + \frac{(a-b)^2}{2a(1-a)}\right] + \mathcal{O}(a-b)^3 . \tag{B.1}$$

Démonstration : En utilisant une série de Taylor, on obtient

$$\begin{aligned}
\ln(1-b) &= \ln(1-a) - (b-a)\frac{1}{1-a} - (b-a)^2\frac{1}{2(1-a)^2} + \mathcal{O}(b-a)^3 \\
\ln\left(\frac{1-a}{1-b}\right) &= (b-a)\frac{1}{1-a} + (b-a)^2\frac{1}{2(1-a)^2} + \mathcal{O}(b-a)^3 \\
(1-a)\ln\left(\frac{1-a}{1-b}\right) &= (b-a) + (b-a)^2\frac{1}{2(1-a)} + \mathcal{O}(b-a)^3 .
\end{aligned}$$

On a donc le développement limité du second terme de la divergence de Kullback-Leibler.

$$\begin{aligned} D\left(a\middle|\middle|b\right) &= a\ln\left(\frac{a}{b}\right)+(1-a)\ln\left(\frac{1-a}{1-b}\right) \\ &= a\ln\left(\frac{a}{b}\right)+(b-a)+(b-a)^2\frac{1}{2(1-a)}+\mathcal{O}\left(b-a\right)^3 \\ &= a\left[\ln\left(\frac{a}{b}\right)-\frac{a-b}{a}+\frac{(a-b)^2}{2a(1-a)}\right]+\mathcal{O}\left(a-b\right)^3. \end{aligned}$$

□

Lemme 6.17. *Soit un réel* $0<a<1$*, et soit* $\epsilon>0$ *un nombre réel tel que* $\frac{\epsilon}{a}=\mathcal{O}\left(\epsilon\right)$ *et* $\frac{\epsilon}{1-a}=\mathcal{O}\left(\epsilon\right)$*. Alors,*

$$D\left(a+\epsilon\middle|\middle|a\right)=\frac{\epsilon^2}{2a(1-a)}+\mathcal{O}\left(\epsilon^3\right).$$

Démonstration : On utilise le développement du lemme 6.16. le terme $\ln\left(\frac{a}{b}\right)$ vaut ici $\ln\left(\frac{a+\epsilon}{a}\right)$ on peut donc développer ce logarithme au voisinage de 1 car $\frac{\epsilon}{a}=\mathcal{O}\left(\epsilon\right)$.

$$\ln\left(\frac{a+\epsilon}{a}\right)=\ln\left(1+\frac{\epsilon}{a}\right)=\frac{\epsilon}{a}-\frac{\epsilon^2}{2a^2}+\mathcal{O}\left(\epsilon^3\right). \tag{B.2}$$

En utilisant le lemme 6.16, on a donc

$$D\left(a+\epsilon\middle|\middle|a\right)=(a+\epsilon)\left[\frac{\epsilon}{a}-\frac{\epsilon^2}{2a^2}-\frac{\epsilon}{a+\epsilon}+\frac{\epsilon^2}{2(a+\epsilon)(1-a-\epsilon)}\right]+\mathcal{O}\left(\epsilon^3\right).$$

On obtient, après une mise au même dénominateur,

$$\begin{aligned} D\left(a+\epsilon\middle|\middle|a\right) &= \frac{(a+\epsilon)a\epsilon^2}{2a^2(a+\epsilon)(1-a-\epsilon)}+\mathcal{O}\left(\epsilon^3\right) \\ &= \frac{\epsilon^2}{2a(1-a-\epsilon)}+\mathcal{O}\left(\epsilon^3\right) \\ &= \frac{\epsilon^2}{2a(1-a)}+\mathcal{O}\left(\epsilon^3\right). \end{aligned}$$

□

Lemme B.1. *Soit un réel* $0<a<1$*, et soit* $\epsilon>0$ *un nombre réel tel que* $\frac{\epsilon}{a}=\mathcal{O}\left(\frac{\epsilon}{1-a}\right)$*. Alors,*

$$\Delta_\epsilon=D\left(a\middle|\middle|a-\epsilon\right)-D\left(a\middle|\middle|a+\epsilon\right)=\frac{2}{3}\epsilon^3\cdot\frac{1-2a}{a^2(1-a)^2}+\mathcal{O}\left(\epsilon^4\right).$$

Démonstration : On sépare Δ_ϵ en deux termes correspondant aux deux coefficients a et $1-a$.

$$\begin{aligned} \Delta_\epsilon &= D\left(a\middle|\middle|a-\epsilon\right)-D\left(a\middle|\middle|a+\epsilon\right) \\ &= a\ln\left(\frac{a+\epsilon}{a-\epsilon}\right)+(1-a)\ln\left(\frac{1-a-\epsilon}{1-a+\epsilon}\right) \\ &= \Delta_{\epsilon,a}+\Delta_{\epsilon,1-a} \end{aligned}$$

Le premier logarithme peut être développé en

$$\begin{aligned}
\Delta_{\epsilon,a} &= a \ln\left[1+\frac{2\epsilon}{a-\epsilon}\right] \\
&= a\left[\frac{2\epsilon}{a-\epsilon} - \frac{2\epsilon^2}{(a-\epsilon)^2} + \frac{8\epsilon^3}{3(a-\epsilon)^3} + \mathcal{O}\left(\epsilon^4\right)\right] \\
&= \frac{1}{1-\frac{\epsilon}{a}}\left[2\epsilon - \frac{2\epsilon^2}{a-\epsilon} + \frac{8\epsilon^3}{3(a-\epsilon)^2} + \mathcal{O}\left(\epsilon^4\right)\right] \\
&= \left[1+\frac{\epsilon}{a}+\frac{\epsilon^2}{a^2}+o\left(\epsilon^2\right)\right]\cdot\left[2\epsilon - \frac{2\epsilon^2}{a-\epsilon} + \frac{8\epsilon^3}{3(a-\epsilon)^2} + \mathcal{O}\left(\epsilon^4\right)\right] \\
&= 2\epsilon - \frac{2\epsilon^2}{(a-\epsilon)} + \frac{8\epsilon^3}{3(a-\epsilon)^2} + \frac{2\epsilon^2}{a} - \frac{2\epsilon^3}{a(a-\epsilon)} + \frac{2\epsilon^3}{a^2} + \mathcal{O}\left(\epsilon^4\right) \\
&= 2\epsilon\left[1+\epsilon\left(\frac{1}{a}-\frac{1}{a-\epsilon}\right)+\epsilon^2\frac{4}{3a^2}+\mathcal{O}\left(\epsilon^3\right)\right].
\end{aligned}$$

Pour le second, on obtient de façon similaire,

$$\Delta_{\epsilon,1-a} = 2\epsilon\left[-1+\epsilon\left(\frac{1}{1-a}-\frac{1}{1-a+\epsilon}\right)-\epsilon^2\frac{4}{3(1-a)^2}+\mathcal{O}\left(\epsilon^3\right)\right].$$

Et donc, en sommant les deux termes,

$$\begin{aligned}
\Delta_\epsilon &= \Delta_{\epsilon,a}+\Delta_{\epsilon,1-a} \\
&= 2\epsilon^2\left[\frac{1}{a}-\frac{1}{a-\epsilon}+\frac{1}{1-a}-\frac{1}{1-a+\epsilon}+\epsilon\left(\frac{4}{3a^2}-\frac{4}{3(1-a)^2}\right)+\mathcal{O}\left(\epsilon^2\right)\right] \\
&= 2\epsilon^2\left[\epsilon\cdot\frac{2a-1}{a^2(1-a)^2}+\epsilon\left(\frac{4}{3a^2}-\frac{4}{3(1-a)^2}\right)+\mathcal{O}\left(\epsilon^2\right)\right].
\end{aligned}$$

Ce qui donne, au final,

$$\Delta_\epsilon = \frac{2}{3}\epsilon^3\cdot\frac{1-2a}{a^2(1-a)^2}+\mathcal{O}\left(\epsilon^4\right).$$

□

B.2 Concentration de la loi bêta

On prouve ici le lemme sur la concentration de la loi bêta utilisé dans la section 6.5.

Lemme 6.19. *Soit b la densité de la distribution bêta ayant pour paramètres $(n-\ell-1,\ell-1)$:*

$$b(t) \stackrel{\text{def}}{=} (n-1)\cdot\binom{n-2}{\ell-1}\cdot t^{n-\ell-1}(1-t)^{\ell-1}.$$

Son maximum est atteint en $t_0 \stackrel{\text{def}}{=} \frac{n-\ell-1}{n-2}$. Soit $\epsilon \stackrel{\text{def}}{=} z\cdot\frac{\sqrt{\ell-1}}{n-2}$, alors, si $z = o\left(\sqrt{\ell}\right)$ et $\ell\in[1,\frac{n}{2}]$, alors,

$$\int_{t_0-\epsilon}^{t_0+\epsilon} b(t)\,dt = 1+\mathcal{O}\left(\frac{1}{\ell^2}+\frac{1}{n}+\frac{e^{-z^2/2}}{z}\right).$$

Avant de prouver le lemme 6.19, l'on va s'intéresser au lemme suivant sur lequel repose totalement sa preuve. Celui-ci est une version un peu plus poussée d'un lemme que l'on peut trouver dans [BH86] et permet d'estimer la valeur de l'intégrale d'une fonction de la forme $e^{-\lambda\phi(t)}$ autour de son maximum atteint en t_0.

Lemme B.2. *Soit $\phi(t)$ une fonction définie sur $]0;1[$ quatre fois différentiable. Supposons que le minimum de ϕ soit 0 et que celui-ci soit atteint en $t_0 \in]\frac{1}{2};1[$ et que $\phi''(t_0) > 0$. Soit λ un nombre réel positif, alors, pour $\epsilon \in]0; 1-t_0[$,*

$$\int_{t_0}^{t_0+\epsilon} e^{-\lambda\phi(t)}\, dt = \int_0^{\phi(t_0+\epsilon)} \left[\frac{1}{\sqrt{2\tau\phi''(t_0)}} - \frac{1}{3}\frac{\phi'''(t_0)}{\phi''^2(t_0)} + A_{t_0}\sqrt{\tau} + o\left(\sqrt{\tau}\right)\right] e^{-\lambda\tau} d\tau,$$

et

$$\int_{t_0-\epsilon}^{t_0} e^{-\lambda\phi(t)}\, dt = \int_0^{\phi(t_0-\epsilon)} \left[\frac{1}{\sqrt{2\tau\phi''(t_0)}} + \frac{1}{3}\frac{\phi'''(t_0)}{\phi''^2(t_0)} + A_{t_0}\sqrt{\tau} + o\left(\sqrt{\tau}\right)\right] e^{-\lambda\tau} d\tau.$$

Où $A_{t_0} \stackrel{\text{def}}{=} \dfrac{\sqrt{2}}{24\phi''(t_0)^{5/2}}\left(\dfrac{5\phi^{(3)}(t_0)^2}{\phi''(t_0)} - 3\phi^{(4)}(t_0)\right)$.

Démonstration : La preuve est assez calculatoire. La première chose est d'effectuer le changement de variable $\tau = \phi(t)$.

$$\int_{t_0}^{t_0\pm\epsilon} e^{-\lambda\phi(t)} dt = \int_0^{\phi(t_0\pm\epsilon)} H(\tau) e^{-\lambda\tau} d\tau$$

avec $H(\tau) = \left.\dfrac{1}{\phi'(t)}\right|_{t=\phi^{-1}(\tau)}$.

Pour pouvoir exprimer H en fonction de τ, il faut trouver d'abord exprimer $t - t_0$ en fonction de τ. On utilise pour cela le développement

$$\phi(t) = \frac{\phi''(t_0)}{2}(t-t_0)^2 + \frac{\phi^{(3)}(t_0)}{6}(t-t_0)^3 + \frac{\phi^{(4)}(t_0)}{24}(t-t_0)^4 + o\left((t-t_0)^4\right).$$

Sans perte de généralité, on suppose t supérieur à t_0 et on obtient.

$$\begin{aligned}(t-t_0)^2 &= \frac{2\phi(t)}{\phi''(t_0)}\left[1 + \frac{1}{3}\frac{\phi^{(3)}(t_0)}{\phi''(t_0)}(t-t_0) + \frac{1}{12}\frac{\phi^{(4)}(t_0)}{\phi''(t_0)}(t-t_0)^2 + o\left((t-t_0)^2\right)\right]^{-1} \\ (t-t_0) &= \sqrt{\frac{2\phi(t)}{\phi''(t_0)}}\left[1 + \frac{1}{6}\frac{\phi^{(3)}(t_0)}{\phi''(t_0)}(t-t_0) + \mathcal{O}\left((t-t_0)^2\right)\right]^{-1}. \qquad \text{(B.3)}\end{aligned}$$

Cela donne le comportement suivant pour $t - t_0$

$$t - t_0 = \sqrt{\frac{2\tau}{\phi''(t_0)}}\left[1 + \mathcal{O}\left(\sqrt{\tau}\right)\right].$$

Si l'on utilise cela dans (B.3), on obtient

$$t - t_0 = \sqrt{\frac{2\tau}{\phi''(t_0)}}\left[1 - \frac{\sqrt{2}}{6}\frac{\phi^{(3)}(t_0)}{\phi''(t_0)^{3/2}}\sqrt{\tau} + o(\sqrt{\tau})\right]. \qquad \text{(B.4)}$$

On va maintenant reprendre l'expression complète de $(t-t_0)^2$ et remplacer $t-t_0$ par l'expression (B.4).

$$\begin{aligned}(t-t_0)^2 &= \frac{2\tau}{\phi''(t_0)}\left[1+\frac{\sqrt{2}}{3}\frac{\phi^{(3)}(t_0)}{\phi''(t_0)^{3/2}}\left[1-\frac{\sqrt{2}}{6}\frac{\phi^{(3)}(t_0)}{\phi''(t_0)^{3/2}}\sqrt{\tau}\right]\sqrt{\tau}+\frac{1}{6}\frac{\phi^{(4)}(t_0)}{\phi''(t_0)^2}\tau+o(\tau)\right]^{-1}\\ t-t_0 &= \sqrt{\frac{2\tau}{\phi''(t_0)}}\left[1+\frac{\sqrt{2}}{3}\frac{\phi^{(3)}(t_0)}{\phi''(t_0)^{3/2}}\sqrt{\tau}+\left(\frac{1}{6}\frac{\phi^{(4)}(t_0)}{\phi''(t_0)^2}-\frac{1}{9}\frac{\phi^{(3)}(t_0)^2}{\phi''(t_0)^3}\right)\tau+o(\tau)\right]^{-1/2}\end{aligned}$$

Et finalement, on obtient

$$t-t_0=\sqrt{\frac{2\tau}{\phi''(t_0)}}\left[1-\frac{\sqrt{2}}{6}\frac{\phi^{(3)}(t_0)}{\phi''(t_0)^{3/2}}\sqrt{\tau}-\left(\frac{1}{12}\frac{\phi^{(4)}(t_0)}{\phi''(t_0)^2}-\frac{5}{36}\frac{\phi^{(3)}(t_0)^2}{\phi''(t_0)^3}\right)\tau+o(\tau)\right]. \quad \text{(B.5)}$$

La même méthode avec $t>t_0$ donne

$$t-t_0=-\sqrt{\frac{2\tau}{\phi''(t_0)}}\left[1+\frac{\sqrt{2}}{6}\frac{\phi^{(3)}(t_0)}{\phi''(t_0)^2}\sqrt{\tau}-\left(\frac{1}{12}\frac{\phi^{(4)}(t_0)}{\phi''(t_0)^2}-\frac{5}{36}\frac{\phi^{(3)}(t_0)^2}{\phi''(t_0)^3}\right)\tau+o(\tau)\right]. \quad \text{(B.6)}$$

Maintenant que l'on a une expression précise pour $t-t_0$ en fonction de τ, on peut retourner au calcul de $H(\tau)$ en fonction de τ. On utilise le développement suivant de ϕ' :

$$\phi'(t)=\phi''(t_0)(t-t_0)+\frac{\phi^{(3)}(t_0)}{2}(t-t_0)^2+\frac{\phi^{(4)}(t_0)}{6}(t-t_0)^3+o\left((t-t_0)^3\right).$$

Ce qui nous donne une expression pour $\frac{1}{\phi'(t)}$

$$\begin{aligned}\frac{1}{\phi'(t)} &= \frac{1}{\phi''(t_0)(t-t_0)+\frac{1}{2}\phi^{(3)}(t_0)(t-t_0)^2+\frac{1}{6}\phi^{(4)}(t_0)(t-t_0)^3+o\left((t-t_0)^3\right)}\\ &= \frac{1}{\phi''(t_0)(t-t_0)}\left[1+\frac{1}{2}\frac{\phi^{(3)}(t_0)}{\phi''(t_0)}(t-t_0)+\frac{1}{6}\frac{\phi^{(4)}(t_0)}{\phi''(t_0)}(t-t_0)^2+o\left((t-t_0)^2\right)\right]^{-1}\\ &= \frac{1-\frac{\phi^{(3)}(t_0)}{2\phi''(t_0)}(t-t_0)+\left(3\frac{\phi^{(3)}(t_0)^2}{\phi''(t_0)}-2\phi^{(4)}(t_0)\right)\frac{(t-t_0)^2}{12\phi''(t_0)}+o\left((t-t_0)^2\right)}{\phi''(t_0)(t-t_0)}\\ &= \frac{1}{\phi''(t_0)(t-t_0)}-\frac{\phi^{(3)}(t_0)}{2\phi''(t_0)^2}+\left(3\frac{\phi^{(3)}(t_0)^2}{\phi''(t_0)}-2\phi^{(4)}(t_0)\right)\frac{t-t_0}{12\phi''(t_0)^2}+o(t-t_0).\end{aligned}$$

On utilise maintenant (B.5) dans cette formule. Le premier terme devient

$$\begin{aligned}\frac{1}{\phi''(t_0)(t-t_0)} &= \frac{1}{\sqrt{2\phi''(t)\tau}}\left[1-\frac{\sqrt{2}}{6}\frac{\phi^{(3)}(t_0)}{\phi''(t_0)^{3/2}}\sqrt{\tau}-\left(\frac{1}{12}\frac{\phi^{(4)}(t_0)}{\phi''(t_0)^2}-\frac{5}{36}\frac{\phi^{(3)}(t_0)^2}{\phi''(t_0)^3}\right)\tau+o(\tau)\right]^{-1}\\ &= \frac{1}{\sqrt{2\phi''(t)\tau}}+\frac{\phi^{(3)}(t_0)}{6\phi''(t_0)^2}+\left(\phi^{(4)}(t_0)-\frac{\phi^{(3)}(t_0)^2}{\phi''(t_0)}\right)\frac{\sqrt{2}}{24\phi''(t_0)^{5/2}}\sqrt{\tau}+o\left(\sqrt{\tau}\right).\end{aligned}$$

Et le terme en $t-t_0$ devient, lui,

$$\left(3\frac{\phi^{(3)}(t_0)^2}{\phi''(t_0)}-2\phi^{(4)}(t_0)\right)\frac{t-t_0}{12\phi''(t_0)^2}=\left(6\frac{\phi^{(3)}(t_0)^2}{\phi''(t_0)}-4\phi^{(4)}(t_0)\right)\frac{\sqrt{2}}{24\phi''(t_0)^{5/2}}\sqrt{\tau}+o(\tau).$$

Ce qui donne, pour $H(\tau)$,

$$H(\tau) = \frac{1}{\sqrt{2\phi''(t_0)\tau}} - \frac{\phi^{(3)}(t_0)}{3\phi''(t_0)^2} + \frac{\sqrt{2}}{24\phi''(t_0)^{5/2}}\left(5\frac{\phi^{(3)}(t_0)^2}{\phi''(t_0)} - 3\phi^{(4)}(t_0)\right)\sqrt{\tau} + o(\tau).$$

Et dans le cas $t < t_0$,

$$-H(\tau) = \frac{1}{\sqrt{2\phi''(t_0)\tau}} + \frac{\phi^{(3)}(t_0)}{3\phi''(t_0)^2} + \frac{\sqrt{2}}{24\phi''(t_0)^{5/2}}\left(5\frac{\phi^{(3)}(t_0)^2}{\phi''(t_0)} - 3\phi^{(4)}(t_0)\right)\sqrt{\tau} + o(\tau).$$

□

On peut donc revenir au lemme 6.19.

Démonstration du lemme 6.19. On rappelle que le but est d'estimer

$$\int_{t_0-\epsilon}^{t_0+\epsilon} (n-1)\cdot\binom{n-2}{\ell-1}\cdot t^{n-\ell-1}(1-t)^{\ell-1}\,dt.$$

On commence d'abord par utiliser l'approximation de Stirling (lemme 6.5) pour le coefficient binomial et, de la même façon que dans la sous-section 6.3.2, on obtient

$$\binom{n-2}{\ell-1} = \sqrt{\frac{1}{2\pi}}\left(\frac{n-2}{n-\ell-1}\right)^{n-\ell-1/2}\left(\frac{n-2}{\ell-1}\right)^{\ell-1/2}\left[1 - \frac{1}{12(\ell-1)} + \mathcal{O}\left(\frac{1}{n}+\frac{1}{\ell^2}\right)\right]$$

et

$$\left(\frac{n-2}{n-\ell-1}\right)^{n-\ell-1}\left(\frac{n-2}{\ell-1}\right)^{\ell-1} t^{n-\ell-1}(1-t)^{\ell-1} = e^{-(n-2)D(t_0||t)}.$$

Et finalement, si on définit l'approximation $\tilde{b}$ de la fonction g par

$$\tilde{b}(t) \stackrel{\text{def}}{=} C_{n,\ell}\cdot e^{-(n-2)D(t_0||t)},$$

avec $C_{n,\ell} \stackrel{\text{def}}{=} (n-1)\cdot\sqrt{\frac{n-2}{2\pi(\ell-1)(n-\ell-1)}}$, on a

$$b(t) = \tilde{b}(t)\left[1 - \frac{1}{12(\ell-1)} + \mathcal{O}\left(\frac{1}{n}+\frac{1}{\ell^2}\right)\right].$$

La fonction $\tilde{b}$ est de la forme des fonctions considérées dans le lemme B.2 et l'on va donc appliquer ce dernier avec $\lambda = n-2$ et $\phi(t) = D(t_0|\,|t)$ et donc

$$\begin{aligned}
\phi''(t_0) = \frac{1}{t_0} + \frac{1}{1-t_0} &= \frac{1}{t_0(1-t_0)} > 0,\\
\phi^{(3)}(t_0) = \frac{2}{(1-t_0)^2} - \frac{2}{t_0^2} &= 2\,\frac{2t_0-1}{t_0^2(1-t_0)^2},\\
\phi^{(4)}(t_0) = \frac{6}{(1-t_0)^3} + \frac{6}{t_0^3} &= 6\,\frac{3t_0^2-3t_0+1}{t_0^3(1-t_0)^3},\\
\text{et } A_{t_0} = \frac{13t_0^2-13t_0+1}{6\sqrt{2t_0(1-t_0)}}&.
\end{aligned}$$

On a bien $\phi''(t_0) > 0$ et $\phi(t_0) = \phi'(t_0) = 0$ et donc on applique le lemme B.2. La restriction $z = o\left(\sqrt{\ell}\right)$ sera remplie par notre choix final de z et la restriction $\ell < n/2$ est supposée être remplie car cela signifie que l'on veut écarter au moins la moitié des clefs. On a donc

$$\int_{t_0}^{t_0+\epsilon} e^{-\lambda\phi(t)}\, dt = \int_0^{\phi(t_0+\epsilon)} \left[\frac{1}{\sqrt{2\tau\phi''(t_0)}} - \frac{1}{3}\frac{\phi'''(t_0)}{\phi''^2(t_0)} + A_{t_0}\sqrt{\tau} + o\left(\sqrt{\tau}\right)\right] e^{-\lambda\tau} d\tau.$$

Il va donc nous falloir calculer trois intégrales ayant les formes suivantes.

Lemme B.3. *Soit $a > 1$ un nombre réel, alors*

1. $\int_0^a e^{-t}\cdot t^{-1/2}\, dt = \sqrt{\pi} - e^{-a}a^{-1/2} + \mathcal{O}\left(e^{-a}a^{-3/2}\right)$.
2. $\int_0^a e^{-t}\, dt = 1 - e^{-a}$.
3. $\int_0^a e^{-t}\cdot t^{1/2}\, dt = \frac{\sqrt{\pi}}{2} - e^{-a}\sqrt{a} + \mathcal{O}\left(e^{-a}a^{-1/2}\right)$.

Démonstration : La preuve est immédiate et résulte simplement de l'intégration par parties de ces intégrales. □

Cela donne donc

$$\begin{aligned}\int_{t_0}^{t_0+\epsilon} e^{-\lambda\phi(t)}\, dt &= \sqrt{\frac{\pi}{2\lambda\phi''(t_0)}} + \mathcal{O}\left(\frac{e^{-\lambda\phi(t_0+\epsilon)}}{\lambda\sqrt{\phi''(t_0)\phi(t_0+\epsilon)}}\right)\\ &- \frac{1}{3\lambda}\frac{\phi^{(3)}(t_0)}{\phi''^2(t_0)} + \mathcal{O}\left(\frac{1}{\lambda}\frac{\phi^{(3)}(t_0)}{\phi''^2(t_0)}e^{-\lambda\phi(t_0+\epsilon)}\right)\\ &+ \frac{A_{t_0}}{2\lambda}\sqrt{\frac{\pi}{\lambda}} + \mathcal{O}\left(A_{t_0}\frac{e^{-\lambda\phi(t_0+\epsilon)}}{\lambda}\sqrt{\phi(t_0+\epsilon)}\right)\end{aligned}$$

et,

$$\begin{aligned}\int_{t_0-\epsilon}^{t_0} e^{-\lambda\phi(t)}\, dt &= \sqrt{\frac{\pi}{2\lambda\phi''(t_0)}} + \mathcal{O}\left(\frac{e^{-\lambda\phi(t_0-\epsilon)}}{\lambda\sqrt{\phi''(t_0)\phi(t_0-\epsilon)}}\right)\\ &+ \frac{1}{3\lambda}\frac{\phi^{(3)}(t_0)}{\phi''^2(t_0)} + \mathcal{O}\left(\frac{1}{\lambda}\frac{\phi^{(3)}(t_0)}{\phi''^2(t_0)}e^{-\lambda\phi(t_0-\epsilon)}\right)\\ &+ \frac{A_{t_0}}{2\lambda}\sqrt{\frac{\pi}{\lambda}} + \mathcal{O}\left(A_{t_0}\frac{e^{-\lambda\phi(t_0-\epsilon)}}{\lambda}\sqrt{\phi(t_0-\epsilon)}\right).\end{aligned}$$

On additionne maintenant les deux intégrales pour obtenir

$$\begin{aligned}
\int_{t_0-\epsilon}^{t_0+\epsilon} e^{-\lambda\phi(t)}\,dt &= \sqrt{\frac{2\pi}{\lambda\phi''(t_0)}} + \mathcal{O}\left(\frac{e^{-\lambda(\phi(t_0-\epsilon)} + e^{-\lambda\phi(t_0+\epsilon))}}{\lambda\epsilon\phi''(t_0)}\right)\\
&+ \mathcal{O}\left(\frac{1}{3\lambda}\frac{\phi^{(3)}(t_0)}{\phi''^2(t_0)}\left(e^{-\lambda\phi(t_0-\epsilon)} + e^{-\lambda\phi(t_0+\epsilon)}\right)\right)\\
&+ \frac{A_{t_0}}{\lambda}\sqrt{\frac{\pi}{\lambda}} + \mathcal{O}\left(\frac{A_{t_0}\epsilon}{\lambda}\sqrt{\phi''(t_0)}\left(e^{-\lambda\phi(t_0-\epsilon)} + e^{-\lambda\phi(t_0+\epsilon)}\right)\right)\\
&= \sqrt{\frac{2\pi}{\lambda\phi''(t_0)}}\cdot\left[1+\sqrt{\phi''(t_0)}\frac{A_{t_0}}{\lambda}\right]\\
&+ \mathcal{O}\left(\frac{1}{\lambda}\left[e^{-\lambda\phi(t_0-\epsilon)} + e^{-\lambda\phi(t_0+\epsilon)}\right]\left[\frac{1}{\epsilon\phi''(t_0)} + \frac{\phi^{(3)}(t_0)}{3\phi''^2(t_0)} + A_{t_0}\epsilon\sqrt{\phi''(t_0)}\right]\right).
\end{aligned}$$

On remplace maintenant λ et les dérivées de ϕ par leurs valeurs en fonction de n, ℓ et t_0

$$\begin{aligned}
\int_{t_0-\epsilon}^{t_0+\epsilon} \tilde{b}(t)\,dt &= \int_{t_0-\epsilon}^{t_0+\epsilon} C_{n,\ell}e^{-(n-2)D(\,t_0||t\,)}\,dt\\
&= C_{n,\ell}\cdot\sqrt{\frac{2\pi t_0(1-t_0)}{n-2}}\cdot\left[1+\frac{13t_0^2-13t_0+1}{12(n-2)t_0(1-t_0)}\right]+R\\
&= \frac{n-1}{n-2}\cdot\left[1+\frac{13t_0^2-13t_0+1}{12(n-2)t_0(1-t_0)}\right]+R\\
&= \left[1+\frac{1}{n-2}\right]\cdot\left[1+\frac{13t_0^2-13t_0+1}{12(n-2)t_0(1-t_0)}\right]+R\\
&= 1+\frac{13t_0^2-13t_0+1}{12(\ell-1)t_0}+\mathcal{O}\left(\frac{1}{n}\right)+R,
\end{aligned}$$

avec,

$$R=\mathcal{O}\left(\frac{C_{n,\ell}}{n}\left(e^{-\lambda\phi(t_0-\epsilon)}+e^{-\lambda\phi(t_0+\epsilon)}\right)\left[\frac{t_0(1-t_0)}{\epsilon}+\frac{2}{3}(2t_0-1)+\frac{13t_0^2-13t_0+1}{12t_0(1-t_0)}\cdot\epsilon\right]\right).$$

Le coefficient entre crochets est dominé par le premier terme qui est d'ordre $\sqrt{\ell}/z$ et donc

$$\begin{aligned}
R &= \mathcal{O}\left(\frac{\sqrt{\ell}C_{n,\ell}}{z\cdot n}\left[e^{-(n-2)D(\,t_0||t_0-\epsilon\,)}+e^{-(n-2)D(\,t_0||t_0+\epsilon\,)}\right]\right)\\
&= \mathcal{O}\left(\frac{\sqrt{\ell}}{z\cdot n}\,C_{n,\ell}e^{-(n-2)D(\,t_0||t_0+\epsilon\,)}\left[1+e^{-(n-2)\Delta_\epsilon}\right]\right)
\end{aligned}$$

où $\Delta_\epsilon \stackrel{\text{def}}{=} D\left(t_0\right||t_0-\epsilon\left.\right) - D\left(t_0\right||t_0+\epsilon\left.\right)$. On va devoir estimer les termes $D\left(t_0\right||t_0-\epsilon\left.\right)$ et Δ_ϵ. C'est le sujet des lemmes 6.17 et B.1. En appliquant le lemme 6.17 on obtient $(n-2)D\left(t_0\right||t_0+\epsilon\left.\right) = \frac{\epsilon^2(n-2)}{2t_0(1-t_0)}$. Ce qui donne, en remplaçant ϵ et t_0 par leurs valeurs, $(n-2)D\left(t_0\right||t_0+\epsilon\left.\right) = \frac{z^2}{2t_0}$. De son côté, le lemme B.1 nous permet de montrer que $e^{-(n-2)\Delta_\epsilon} =$

$o(1)$ ce qui donne, au final, $R = \mathcal{O}\left(\frac{e^{-z^2/2}}{z}\right)$. On termine maintenant la preuve.

$$\begin{aligned}
\int_{t_0-\epsilon}^{t_0+\epsilon} b(t)\,dt &= \int_{t_0-\epsilon}^{t_0+\epsilon} \tilde{b}(t)\left[1 - \frac{1}{12(\ell-1)} + \mathcal{O}\left(\frac{1}{n}+\frac{1}{\ell^2}\right)\right] \\
&= \left[1 + \frac{13t_0^2 - 13t_0 + 1}{12(\ell-1)t_0} + \mathcal{O}\left(\frac{1}{n}+\frac{e^{-z^2/2}}{z}\right)\right] \\
&\cdot \left[1 - \frac{1}{12(\ell-1)} + \mathcal{O}\left(\frac{1}{n}+\frac{1}{\ell^2}\right)\right] \\
&= 1 - (1-t_0)\frac{13t_0-1}{12t_0}\cdot\frac{1}{\ell-1} + \mathcal{O}\left(\frac{1}{\ell^2}+\frac{1}{n}+\frac{e^{-z^2/2}}{z}\right) \\
&= 1 + \mathcal{O}\left(\frac{1}{\ell^2}+\frac{1}{n}+\frac{e^{-z^2/2}}{z}\right).
\end{aligned}$$

□

B.3 Preuve du Théorème 8.3

Dans cette section, on prouve le théorème 8.3 dont on rappelle l'énoncé.

Théorème 8.3. *Supposons que les bits des κ_j soient choisis de façon aléatoire uniforme, et que $\sum_{j=1}^n \mathrm{Cap}\left(\sigma_j^2\right) = d + \delta n$ pour un $\delta > 0$.*

Alors, si on note P_{err} la probabilité que la bonne clef $\boldsymbol{K}$ ne soit pas la plus vraisemblable connaissant les statistiques contenues dans $\boldsymbol{Y}$, il existe une constante A telle que

$$P_{\mathrm{err}} \leq \frac{A}{\delta^2 n} + 2^{-\delta n/2}.$$

Démonstration. Commençons par noter $(\boldsymbol{X}, \boldsymbol{Y})$ un couple de variables aléatoires où $\boldsymbol{X} = (X_j)_{1\leq j\leq n}$ est uniformément distribué sur $\{0,1\}^n$ et $\boldsymbol{Y} = (Y_i)_{1\leq j\leq n}$ est la sortie d'un canal binaire à bruit blanc additif gaussien quand $\boldsymbol{X}$ est envoyé dessus. Autrement dit,

$$Y_j = (-1)^{X_j} + N_j, \tag{B.7}$$

où les N_j sont des variables aléatoires indépendantes suivant un e distribution gaussienne centrée de variance σ_j^2.

Commençons par quelques définitions.

Définition B.4. Pour $\epsilon > 0$, on définit l'ensemble T_ϵ des séquences typiques ϵ-jointes de $\{0,1\}^n \times \mathbb{R}^n$ par $T_\epsilon \stackrel{\text{def}}{=} \bigcup_{\boldsymbol{x}\in\{0,1\}^n} \{\boldsymbol{x}\} \times T_\epsilon(\boldsymbol{x})$ avec

$$\begin{aligned}
T_\epsilon(\boldsymbol{x}) \stackrel{\text{def}}{=} \{&\boldsymbol{y} \in \mathbb{R}^n : |-\log_2(f(\boldsymbol{y})) - H(\boldsymbol{Y})| < n\epsilon && \text{(B.8)}\\
&\left|-\log_2\left(f(\boldsymbol{y}|\boldsymbol{x})2^{-n}\right) - H(\boldsymbol{X},\boldsymbol{Y})\right| < n\epsilon\} && \text{(B.9)}
\end{aligned}$$

où $f(\boldsymbol{y})$ est la densité de la variable$\boldsymbol{Y}$ et $f(\boldsymbol{y}|\boldsymbol{x})$ celle de $\boldsymbol{Y}$ sachant que $\boldsymbol{X}$ vaut $\boldsymbol{x}$.

Le entropies de $\boldsymbol{Y}$ et $(\boldsymbol{X}, \boldsymbol{Y})$ sont données par le lemme suivant.

Lemme B.5.

$$\begin{aligned} H(\boldsymbol{Y}) &= \sum_{j=1}^{n} \mathrm{Cap}\left(\sigma_j^2\right) + \frac{1}{2}\log_2(2\pi e\sigma_j^2) \\ H(\boldsymbol{X},\boldsymbol{Y}) &= n + \sum_{j=1}^{n} \frac{1}{2}\log_2(2\pi e\sigma_j^2) \end{aligned}$$

Démonstration. Tout d'abord, rappelons que les Y_j sont indépendant. C'est pourquoi $H(\boldsymbol{Y}) = \sum_{j=1}^{n} H(Y_j)$. De plus, en utilisant les définitions de l'entropie et de l'information mutuelle, $H(Y_j) = I(X_j;Y_j) + H(Y_j|X_j)$. Comme X_j est distribuée uniformément sur $\{0,1\}$ ce qui maximise l'information mutuelle $I(Y;X)$ entre l'entrée et la sortie d'un canal gaussien, on a $I(X_j;Y_j) = \mathrm{Cap}\left(\sigma_j^2\right)$. En ce qui concerne l'entropie $H(Y_j|X_j)$, il est clair que l'incertitude sur Y_j une fois X_j connu est essentiellement due à N_j et donc $H(Y_j|X_j) = H(N_j)$. La valeur de cette entropie est obtenue par un calcul standard (voir [CT91]) :

$$H(N_j) = \frac{1}{2}\log_2(2\pi e\sigma_j^2). \tag{B.10}$$

Si on rassemble tous ces éléments, on obtient l'expression

$$H(\boldsymbol{Y}) = \sum_{j=1}^{n} \mathrm{Cap}\left(\sigma_j^2\right) + \frac{1}{2}\log_2(2\pi e\sigma_j^2).$$

Pour la seconde entropie, on utilise des arguments similaires :

$$\begin{aligned} H(\boldsymbol{X},\boldsymbol{Y}) &= H(\boldsymbol{X}) + H(\boldsymbol{Y}|\boldsymbol{X}) \\ &= n + \sum_{\boldsymbol{x}\in\{0,1\}^n} \frac{1}{2^n} H(\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x}) \\ &= n + \sum_{\boldsymbol{x}\in\{0,1\}^n} \frac{1}{2^n} H(N_1,\ldots,N_n) \\ &= n + H(N_1,\ldots,N_n) \\ &= n + \sum_{i=1}^{n} H(N_j) \\ &= n + \sum_{i=1}^{n} \frac{1}{2}\log_2(2\pi e\sigma_j^2). \end{aligned}$$

□

On va maintenant justifier le terme d'ensemble typique que l'on va employer pour parler de T_ϵ et donc montrer qu'il est très peu probable qu'une réalisation de $(\boldsymbol{X},\boldsymbol{Y})$ tombe hors de cet ensemble.

Lemme B.6. *Il existe une constante A telle que*

$$\Pr\left[(\boldsymbol{X},\boldsymbol{Y}) \notin T_\epsilon\right] \leq \frac{A}{\epsilon^2 n}.$$

Pour prouver ce lemme on va avoir besoin de quelques lemmes techniques.

Lemme B.7. *Soit* $U_j \stackrel{\text{def}}{=} -\log_2 f_j(Y_j)$ *où* f_j *est donnée par*

$$f_j(y) \stackrel{\text{def}}{=} \frac{1}{2\sqrt{2\pi\sigma_j^2}} \left(e^{-\frac{(y-1)^2}{2\sigma_j^2}} + e^{-\frac{(y+1)^2}{2\sigma_j^2}} \right).$$

On note aussi $V_j \stackrel{\text{def}}{=} -\log_2 \left(\frac{g_j(Y_j-(-1)^{X_j})}{2} \right)$ *avec* g_j *la densité d'une variable gaussienne centrée de variance* σ_j^2*. Alors,*

$$\begin{aligned} -\log_2(f(\boldsymbol{Y})) - H(\boldsymbol{Y}) &= \sum_{j=1}^n U_j - \mathbb{E}\left(\sum_{j=1}^n U_j\right) \\ -\log_2(f(\boldsymbol{Y}|\boldsymbol{X})2^{-n}) - H(\boldsymbol{X},\boldsymbol{Y}) &= \sum_{j=1}^n V_j - \mathbb{E}\left(\sum_{j=1}^n V_j\right). \end{aligned}$$

Démonstration. Pour la première équation il suffit de remarquer que

$$-\log_2(f(\boldsymbol{Y})) = -\log_2\left(\prod_{j=1}^n f_j(Y_j)\right) = -\sum_{j=1}^n \log_2(f_j(Y_j)) = \sum_{j=1}^n U_j$$

et que $H(\boldsymbol{Y}) = \mathbb{E}(-\log_2 f(\boldsymbol{Y}))$. On obtient la seconde de façon similaire. □

Le fait que $(\boldsymbol{X},\boldsymbol{Y})$ tombe hors de notre ensemble typique revient donc à dire que les sommes $\sum_{j=1}^n U_j$ ou $\sum_{j=1}^n V_j$ dévient de leur espérance de plus de $n\epsilon$. Soit X une variable aléatoire réelle de variance $\mathbb{V}(X)$. Alors, Tchebychev nous dit que pour $t > 0$,

$$\Pr[|X - \mathbb{E}(X)| \geq t] \leq \frac{\mathbb{V}(X)}{t^2}. \tag{B.11}$$

Si on veut pouvoir utiliser cette inégalité, il faut pouvoir estimer les variances des U_j et V_j.

Lemme B.8. *Il existe une constante* A *telle que pour tout* j*,*

$$\mathbb{V}(V_j) \leq A \quad \text{et} \quad \mathbb{V}(U_j) \leq A.$$

Démonstration. On va prouver la première assertion. On rappelle que (B.7) nous donne $N_j = Y_j - (-1)^{X_j}$.

$$\begin{aligned} \bar{V_j} \stackrel{\text{def}}{=} V_j - \mathbb{E}(V_j) &= -\log_2\left(\frac{g_j(N_j)}{2}\right) - \mathbb{E}\left(-\log_2\left(\frac{g_j(N_j)}{2}\right)\right) \\ &= -\log_2(g_j(N_j)) - \frac{1}{2}\log_2(2e\pi\sigma_j^2) \end{aligned}$$

$$\begin{aligned}
\bar{V}_j &\stackrel{\text{def}}{=} V_j - \mathbb{E}(V_j) \\
&= -\log_2\left(\frac{g_j(N_j)}{2}\right) - \mathbb{E}\left(\left(-\log_2\left(\frac{N_j}{2}\right)\right)\right) \\
&= -\log_2\left(\frac{g_j(N_j)}{2}\right) - 1 - H(N_j) \\
&= -\log_2((g_j(N_j)) - H(N_j) \\
&= -\log_2(g_j(N_j)) - \frac{1}{2}\log_2(2e\pi\sigma_j^2)
\end{aligned}$$

où la dernière égalité provient de (B.10). On peut donc écrire

$$\bar{V}_j = \log_2(e)\frac{N_j^2}{2\sigma_j^2} + \frac{1}{2}\log_2(2\pi\sigma_j^2) - \frac{1}{2}\log_2(2e\pi\sigma_j^2) = \frac{\log_2(e)}{2}\left(\frac{N_j^2}{\sigma_j^2} - 1\right),$$

$$\begin{aligned}
\bar{V}_j &= \log_2(e)\frac{N_j^2}{2\sigma_j^2} + \frac{1}{2}\log_2(2\pi\sigma_j^2) - \frac{1}{2}\log_2(2e\pi\sigma_j^2) \\
&= \frac{\log_2(e)}{2}\left(\frac{N_j^2}{\sigma_j^2} - 1\right)
\end{aligned}$$

et donc

$$\begin{aligned}
\mathbb{V}(V_j) &= \mathbb{E}\left(\bar{V}_j{}^2\right) \\
&= \mathbb{E}\left(\frac{\log_2(e)^2}{4}\left(\frac{N_j^2}{\sigma_j^2} - 1\right)^2\right) \\
&= \frac{\log_2(e)^2}{4}\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi\sigma_j^2}}\left(\frac{u^2}{\sigma_j^2} - 1\right)^2 e^{-\frac{u^2}{2\sigma_j^2}}du \\
&= \frac{\log_2(e)^2}{4}\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}\left(v^2 - 1\right)^2 e^{-\frac{v^2}{2}}du
\end{aligned}$$

Où la dernière équation est obtenue suite au changement de variable $v = \frac{u}{\sigma_j}$. La variance de V_j est donc constante ce qui prouve la première partie du lemme.

Pour la seconde moitié, on utilisera l'inégalité suivante Pour u positif,

$$\frac{e^{-\frac{(u-1)^2}{2\sigma_j^2}}}{2\sqrt{2\pi\sigma_j^2}} \leq f_j(u) \leq \frac{e^{-\frac{(u-1)^2}{2\sigma_j^2}}}{\sqrt{2\pi\sigma_j^2}}. \tag{B.12}$$

On rappelle que $\mathbb{E}(U_j) = H(Y_j) = H(Y_j|X_j) + I(X_j;Y_j) = H(N_j) + I(X_j;Y_j)$. De plus, les propriétés de l'information mutuelle font que $0 \leq I(X_j;Y_j) \leq H(X_j) = 1$. Et comme $H(N_j) = \frac{1}{2}\log_2(2e\pi\sigma_j^2)$ on obtient l'encadrement suivant.

$$\frac{1}{2}\log_2(2e\pi\sigma_j^2) \leq \mathbb{E}(U_j) \leq \frac{1}{2}\log_2(2e\pi\sigma_j^2) + 1. \tag{B.13}$$

On note $u = Y_j$ afin de simplifier les expressions à venir. Supposons que U_j est supérieur à son espérance et que celle-ci soit positive. Autrement dit, $-\log_2 f_j(u) \geq \mathbb{E}(U_j) \geq 0$. Si on note $\bar{U}_j \stackrel{\text{def}}{=} U_j - \mathbb{E}(U_j)$, alors,

$$\begin{aligned}
\bar{U_j}^2 &= (U_j - \mathbb{E}(U_j))^2 \\
&= (-\log_2(f_j(u)) - \mathbb{E}(U_j))^2 \\
&\leq \left(\log_2(e)\frac{(u-1)^2}{2\sigma_j^2} + 1 + \frac{1}{2}\log_2(2\pi\sigma_j^2) - \frac{1}{2}\log_2(2e\pi\sigma_j^2)\right)^2 \\
&= \left(\log_2(e)\frac{(u-1)^2}{2\sigma_j^2} - \frac{1}{2}\log_2(e/2)\right)^2 \qquad \text{(B.14)}
\end{aligned}$$

En utilisant (B.12) et (B.13), on obtient

$$\begin{aligned}
\mathbb{V}(U_j) = \mathbb{E}\left(\bar{U_j}\right)^2 &= \int_{-\infty}^{\infty} \bar{U_j}^2 f_j(u)du \\
&= \int_{-\infty}^{0} \bar{U_j}^2 f_j(u)du + \int_{0}^{\mathbb{E}(U_j)} \bar{U_j}^2 f_j(u)du + \int_{\mathbb{E}(U_j)}^{\infty} \bar{U_j}^2 f_j(u)du
\end{aligned}$$

En utilisant la borne (B.14), on déduit que

$$\begin{aligned}
\int_{\mathbb{E}(U_j)}^{\infty} \bar{U_j}^2 f_j(u)du &\leq \int_{\mathbb{E}(U_j)}^{\infty} \left(\log_2(e)\frac{(u-1)^2}{2\sigma_j^2} - \frac{1}{2}\log_2(e/2)\right)^2 f_j(u)du \\
&\leq \int_{\mathbb{E}(U_j)}^{\infty} \left(\log_2(e)\frac{(u-1)^2}{2\sigma_j^2} - \frac{1}{2}\log_2(e/2)\right)^2 \frac{e^{-\frac{(u-1)^2}{2\sigma_j^2}}}{\sqrt{2\pi\sigma_j^2}}du \qquad \text{(B.15)} \\
&= \int_{\frac{\mathbb{E}(U_j)-1}{\sigma_j}}^{\infty} \left(\log_2(e)\frac{v^2}{2} - \frac{1}{2}\log_2(e/2)\right)^2 \frac{e^{-\frac{v^2}{2}}}{\sqrt{2\pi}}dv \qquad \text{(B.16)} \\
&\leq \int_{-\infty}^{\infty} \left(\log_2(e)\frac{v^2}{2} - \frac{1}{2}\log_2(e/2)\right)^2 \frac{e^{-\frac{v^2}{2}}}{\sqrt{2\pi}}dv,
\end{aligned}$$

où (B.15) est une conséquence de (B.12) et (B.16) provient du changement de variable $v = \frac{u-1}{\sigma_j}$. Les deux autres intégrales de (B.15) peuvent être traitées de façon similaire en utilisant, à la place de (B.12), l'encadrement suivant pour u négatif.

$$\frac{e^{-\frac{(u+1)^2}{2\sigma_j^2}}}{2\sqrt{2\pi\sigma_j^2}} \leq f_j(u) \leq \frac{e^{-\frac{(u+1)^2}{2\sigma_j^2}}}{\sqrt{2\pi\sigma_j^2}}. \qquad \text{(B.17)}$$

Ce qui nous donne une borne supérieure pour les variances $\mathbb{V}(U_j)$. □

On est enfin prêt à prouver le lemme B.6.

Démonstration. On commence par écrire

$$\begin{aligned}\Pr\left[(\boldsymbol{X},\boldsymbol{Y})\notin T_\epsilon\right] &= \Pr\left[\{|-\log_2(f(\boldsymbol{Y}))-H(\boldsymbol{Y})|\geq n\epsilon\}\cup\left\{\left|-\log_2\left(f(\boldsymbol{Y}|\boldsymbol{X})2^{-n}\right]-H(\boldsymbol{X},\boldsymbol{Y})\right|\geq n\epsilon\right\}\right.\\ &\leq \Pr\left[|-\log_2(f(\boldsymbol{Y}))-H(\boldsymbol{Y})|\geq n\epsilon\right]+\Pr\left[\left|-\log_2\left(f(\boldsymbol{Y}|\boldsymbol{X})2^{-n}\right]-H(\boldsymbol{X},\boldsymbol{Y})\right|\geq n\epsilon\right)\\ &= \Pr\left[|U-\mathbb{E}(U)|\geq n\epsilon\right]+\Pr\left[|V-\mathbb{E}(V)|\geq n\epsilon\right].\end{aligned}$$

avec $U\overset{\text{def}}{=}\sum_{j=1}^n U_j$ et $V\overset{\text{def}}{=}\sum_{j=1}^n V_j$. On utilise alors l'inégalité de Tchebychev avec les bornes sur $\mathbb{V}(U)=\sum_{j=1}^n\mathbb{V}(U_j)\leq n\,A$ et $\mathbb{V}(V)=\sum_{j=1}^n\mathbb{V}(V_j)\leq n\,A$ afin d'obtenir

$$\Pr\left[(\boldsymbol{X},\boldsymbol{Y})\notin T_\epsilon\right]\leq\frac{2A}{n\epsilon^2}. \tag{B.18}$$

□

On a donc montré qu'il est peu probable que $(\boldsymbol{X},\boldsymbol{Y})$ tombe hors de notre ensemble typique. On peut, de plus, montrer que le volume euclidien (noté Vol) de cet ensemble est assez petit.

Lemme B.9.

$$\sum_{\boldsymbol{x}\in\{0,1\}^n}\mathrm{Vol}(\mathrm{T}_\epsilon(\boldsymbol{x}))\leq 2^{\mathrm{H}(\boldsymbol{X},\boldsymbol{Y})+\epsilon\mathrm{n}}.$$

Démonstration. Remarquons que,

$$\begin{aligned}1=\sum_{\boldsymbol{x}\in\{0,1\}^n}\frac{1}{2^n}\int_{\mathbb{R}^n}f(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{y} &\geq \sum_{\boldsymbol{x}\in\{0,1\}^n}\frac{1}{2^n}\int_{T_\epsilon(\boldsymbol{x})}f(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{y}\\ &\geq \sum_{\boldsymbol{x}\in\{0,1\}^n}\mathrm{Vol}(\mathrm{T}_\epsilon(\boldsymbol{x}))2^{-\mathrm{H}(\boldsymbol{X},\boldsymbol{Y})-\epsilon\mathrm{n}}.\end{aligned}$$

La dernière inégalité vient de (B.9). □

Ce résultat va nous servir à démontrer la proposition suivante

Proposition B.10. *Soit $(\tilde{\boldsymbol{X}},\tilde{\boldsymbol{Y}})$ un couple de variables indépendantes où $\tilde{\boldsymbol{X}}$ est distribuée uniformément et $\tilde{\boldsymbol{Y}}$ a la même distribution que $\boldsymbol{Y}$. Alors,*

$$\Pr\left[(\tilde{\boldsymbol{X}},\tilde{\boldsymbol{Y}})\in T_\epsilon\right]\leq 2^{-C+2n\epsilon},$$

avec $C\overset{\text{def}}{=}\sum_{j=1}^n\mathrm{Cap}\left(()\,\sigma_j^2\right)$.

Démonstration. On évalue $\Pr\left[(\tilde{\boldsymbol{X}},\tilde{\boldsymbol{Y}})\in T_\epsilon\right]$,

$$\Pr\left[(\tilde{\boldsymbol{X}},\tilde{\boldsymbol{Y}})\in T_\epsilon\right]=\sum_{\boldsymbol{x}\in\{0,1\}^n}\frac{1}{2^n}\int_{T_\epsilon(\boldsymbol{x})}f(\boldsymbol{y})\leq\sum_{\boldsymbol{x}\in\{0,1\}^n}\frac{1}{2^n}\mathrm{Vol}(\mathrm{T}_\epsilon(\boldsymbol{x}))2^{-\mathrm{H}(\boldsymbol{Y})+\epsilon\mathrm{n}}.$$

La dernière inégalité venant de (B.8). On utilise maintenant lemme B.9 pour obtenir

$$\Pr\left[(\tilde{\boldsymbol{X}},\tilde{\boldsymbol{Y}})\in T_\epsilon\right]\leq\frac{1}{2^n}2^{H(\boldsymbol{X},\boldsymbol{Y})+\epsilon n}2^{-H(\boldsymbol{Y})+\epsilon n}\leq 2^{-n+H(\boldsymbol{X};\boldsymbol{Y})-H(\boldsymbol{Y})+2\epsilon n}.$$

En utilisant les expressions données par le lemme B.5 pour les entropies, on a alors

$$-n+H(\boldsymbol{X},\boldsymbol{Y})-H(\boldsymbol{Y})=-\sum_{j=1}^n\mathrm{Cap}\left(\sigma_j^2\right).$$

□

Tous ces résultats vont maintenant être utilisés afin d'analyser le décodeur par ensembles typiques prenant en entrée un vecteur $\boldsymbol{y}$ de $\mathbb{R}^n$ qui est la sortie d'un canal binaire à bruit blanc additif gaussien, un paramètre réel ϵ, et qui renvoies soit "Échec" ou un vecteur $\tilde{\boldsymbol{k}} \in \{0,1\}^n$.

Algorithme 16 : Décodeur par ensembles typiques.

compteur $\leftarrow 0$;
pour *toutes les valeurs* $\tilde{\boldsymbol{k}}$ *de* $\tilde{\boldsymbol{K}}$ **faire**
  si $\boldsymbol{y} \in T_\epsilon(\tilde{\boldsymbol{k}})$ **alors**
    compteur $\leftarrow$ *compteur* $+ 1$;
    $res = \tilde{\boldsymbol{k}}$;
  fin
fin
si *compteur* $= 1$ **alors**
  retourner *res;*
sinon
  retourner *Échec;*
fin

Cet algorithme renvoie donc la bonne valeur de $\tilde{\boldsymbol{k}}$ si et seulement si $\boldsymbol{y}$ appartient à l'ensemble typique de la bonne clef et uniquement à celui-ci. On va maintenant s'atteler à prouver le théorème 8.3. Soit $\tilde{\boldsymbol{k}}$ la bonne valeur de $\tilde{\boldsymbol{K}}$ et soit $\mathcal{C}$ l'ensemble des valeurs possibles de $\tilde{\boldsymbol{K}}$. La probabilité P_{err} que le décodeur par ensembles typiques échoue est clairement majorée par

$$P_{\text{err}} \leq \Pr_{\boldsymbol{y},\mathcal{C}}\left[\overline{T_\epsilon(\tilde{\boldsymbol{k}})}\right] + \sum_{\tilde{\boldsymbol{k}}' \in \mathcal{C}, \tilde{\boldsymbol{k}}' \neq \tilde{\boldsymbol{k}}} \Pr_{\boldsymbol{y},\mathcal{C}}\left[T_\epsilon(\tilde{\boldsymbol{k}}')\right] \tag{B.19}$$

où $\overline{T_\epsilon(\tilde{\boldsymbol{k}})}$ est l'ensemble complémentaire de $T_\epsilon(\tilde{\boldsymbol{k}})$. Or, le lemme B.6 nous dit que

$$\Pr_{\boldsymbol{y},\mathcal{C}}\left[\overline{T_\epsilon(\tilde{\boldsymbol{k}})}\right] = \Pr\left[(\boldsymbol{X},\boldsymbol{Y}) \notin T_\epsilon\right] \leq \frac{A}{\varepsilon^2 n},$$

et pour $\tilde{\boldsymbol{k}}' \neq \tilde{\boldsymbol{k}}$, proposition B.10 nous dit

$$\begin{aligned}\sum_{\tilde{\boldsymbol{k}}' \in \mathcal{C}, \tilde{\boldsymbol{k}}' \neq \tilde{\boldsymbol{k}}} \Pr_{\boldsymbol{y},\mathcal{C}}\left[T_\epsilon(\tilde{\boldsymbol{k}}')\right] &\leq \sum_{\tilde{\boldsymbol{k}}' \in \mathcal{C}} \Pr_{\boldsymbol{y},\mathcal{C}}\left[T_\epsilon(\tilde{\boldsymbol{k}}')\right] = 2^r \Pr\left[(\tilde{\boldsymbol{X}},\tilde{\boldsymbol{Y}}) \in T_\epsilon\right] \\ &\leq 2^{r - \sum_{j=1}^n \text{Cap}\left(\sigma_j^2\right) + 2\epsilon n}.\end{aligned}$$

En insérant ces deux bornes dans (B.19) on obtient $P_{\text{err}} \leq \frac{A}{\varepsilon^2 n} + 2^{r - \sum_{j=1}^n \text{Cap}\left(\sigma_j^2\right) + 2\epsilon n} \leq \frac{A}{\varepsilon^2 n} + 2^{-\delta n + 2\epsilon n}$. On termine la preuve en choisissant $\epsilon = \frac{\delta}{4}$. □

B.4 Distribution d'une somme de variables discrètes i.i.d.

On va ici prouver le théorème 9.7. On va pour cela introduire quelques notations afin de simplifier les expressions et prouver trois lemmes techniques qui nous seront utiles.

Notations. Pour simplifier, on notera q_i les probabilités $p_0^{(i)}$ ou $p^{(i)}$. On pourra ainsi traiter les deux cas simultanément. De plus, pour alléger les formules, on note

$$\bar{q} \stackrel{\text{def}}{=} \frac{\sum_{i=1}^d q_i}{d} \quad , \quad m_2 \stackrel{\text{def}}{=} \frac{\sum_{i=1}^d q_i^2}{d} \quad \text{et} \quad d \stackrel{\text{def}}{=} |\Delta_0|.$$

Le théorème 9.7 utilise le logarithme de la fonction génératrice des moments des variables $D_{\boldsymbol{p}}(\boldsymbol{k})$ qui sont une somme de d variables de Bernoulli de paramètres q_i.

$$\mu(\eta) = \sum_{i=1}^d \ln\left(1 - q_i + q_i e^{\eta}\right).$$

Ses dérivées sont

$$\mu'(\eta) = \sum_{i=1}^d \frac{q_i e^{\eta}}{1 - q_i + q_i e^{\eta}} \quad \text{et} \quad \mu''(\eta) = \sum_{i=1}^d \frac{q_i e^{\eta}(1-q_i)}{(1 - q_i + q_i e^{\eta})^2}.$$

une valeur joue un rôle particulier dans ce théorème, c'est la valeur s_r de s telle que $\mu'(\eta_r) = d\,\tau$. Le théorème 9.7 dit essentiellement que remplacer η_r par $\eta_0 \stackrel{\text{def}}{=} \ln\left(\frac{\tau(1-\bar{q})}{\bar{q}(1-\tau)}\right)$ donne une bonne estimation de la queue de la distribution de $D(\boldsymbol{k})$.

On va pour cela définir la fonction f telle que $f(\eta_r) = \eta_r$:

$$f \stackrel{\text{def}}{=} \ln(d\tau) - \ln\left(\sum_{i=1}^d \frac{q_i}{1 - q_i + q_i e^{\ ta}}\right).$$

En remarquant que $\mu''(\eta) = \mu'(\eta)(1 - f'(\eta))$, on peut écrire le résultat du théorème 9.6 comme

$$\Pr\left[D(\boldsymbol{k}) \geq d\tau\, N/2\right] = e^{\frac{N}{2}[\mu(\eta_r) - \eta_r\, d\tau]} \left[\frac{1}{|\eta_r|\sqrt{\pi d\tau N(1 - f'(\eta_r))}} + o\left(\frac{1}{\sqrt{N}}\right)\right]. \tag{B.20}$$

La preuve du théorème 9.7 consiste à quantifier l'erreur commise en remplaçant η_r par s_0 dans cette formule. On va pour cela avoir besoin des lemmes suivants.

Lemme B.11. *En utilisant les notations définies plus haut,*

$$f(\eta_0) - \eta_0 = \frac{\tau - \bar{q}}{\bar{q}^2} \cdot (\bar{q}^2 - m_2) + o\left(\frac{\tau - \bar{q}}{\bar{q}^2} \cdot (\bar{q}^2 - m_2)\right).$$

Démonstration : On commence par calculer précisément $f(\eta_0) - \eta_0$.

$$\begin{aligned}
f(\eta_0) &= \ln(d\tau) - \ln\left(\sum_{i=1}^d \frac{q_i}{1 - q_i + q_i e^{\eta_0}}\right) \\
&= \ln\left(\frac{d\tau}{(1-\tau)\bar{q}}\right) - \ln\left(\sum_{i=1}^d \frac{q_i}{(1-q_i)(1-\tau)\bar{q} + q_i(1-\bar{q})\tau}\right) \\
&= \ln\left(\frac{(1-\bar{q})\tau}{(1-\tau)\bar{q}}\right) - \ln\left(\frac{1-\bar{q}}{d}\sum_{i=1}^d \frac{q_i}{(1-q_i)(1-\tau)\bar{q} + q_i(1-\bar{q})\tau}\right) \\
f(\eta_0) &= \eta_0 - \ln\left(\frac{1}{d}\sum_{i=1}^d q_i \cdot \frac{1-\bar{q}}{(1-q_i)(1-\tau)\bar{q} + q_i(1-\bar{q})\tau}\right).
\end{aligned}$$

On quantifie ensuite l'erreur.

$$\begin{aligned} f(\eta_0) - \eta_0 &= -\ln\left(\frac{1}{d}\sum_{i=1}^{d} q_i \cdot \frac{1-\bar{q}}{q_i(\tau-\bar{q}) + \bar{q}(1-\tau)}\right) \\ &= -\ln\left(\frac{1}{d}\sum_{i=1}^{d} \frac{q_i(1-\bar{q})}{\bar{q}(1-\tau)} \cdot \frac{1}{1+\frac{q_i(\tau-\bar{q})}{\bar{q}(1-\tau)}}\right) \\ &= -\ln\left(\frac{1}{d}\sum_{i=1}^{d} \frac{q_i}{\bar{q}}(1-\bar{q})[1+\tau+o(\tau)]\left[1-\frac{q_i(\tau-\bar{q})}{\bar{q}(1-\tau)}+o\left(\frac{q_i(\tau-\bar{q})}{\bar{q}}\right)\right]\right). \end{aligned}$$

$$\begin{aligned} f(\eta_0) - \eta_0 &= -\ln\left(\sum_{i=1}^{d}\frac{q_i}{d\bar{q}} + \sum_{i=1}^{d}\frac{\tau-\bar{q}}{d\bar{q}}\left[q_i - \frac{q_i^2}{\bar{q}}\right] + o\left(\frac{\tau-\bar{q}}{d\bar{q}}\left[q_i - \frac{q_i^2}{\bar{q}}\right]\right)\right) \\ &= -\ln\left(1 + \frac{\tau-\bar{q}}{\bar{q}^2}\cdot(\bar{q}^2 - m_2) + o\left(\frac{\tau-\bar{q}}{\bar{q}^2}\cdot(\bar{q}^2-m_2)\right)\right) \\ &= \frac{\tau-\bar{q}}{\bar{q}^2}\cdot(\bar{q}^2-m_2) + o\left(\frac{\tau-\bar{q}}{\bar{q}^2}\cdot(\bar{q}^2-m_2)\right). \end{aligned}$$

□

Lemme B.12. *En utilisant les notations définies plus haut,*

$$\eta_r = \eta_0 + \mathcal{O}\left(\frac{\tau-\bar{q}}{\bar{q}^2}\cdot(\bar{q}^2-m_2)\right) \quad \text{et} \quad f'(\eta_0) = \tau\frac{m_2}{\bar{q}^2} + o\left(\tau\frac{m_2}{\bar{q}^2}\right).$$

Démonstration : Le développement en série de Taylor de f donne

$$f(\eta_r) = f(\eta_0) + (\eta_r - \eta_0)f'(\eta_0) + \mathcal{O}\left(f''(\eta_0)(\eta_r-\eta_0)^2\right).$$

Comme $f(\eta_r) = \eta_r$, on a alors $\eta_r = \eta_0 + \mathcal{O}\left(\frac{f(\eta_0)-\eta_0}{1-f'(\eta_0)}\right)$.
On a donc besoin d'estimer $f'(\eta_0)$. Or, par définition,

$$f'(\eta_0) = \sum_{i=1}^{d}\frac{q_i^2 e^{\eta_0}}{(1-q_i+q_i e^{\eta_0})^2}\cdot\left[\sum_{i=1}^{d}\frac{q_i}{1-q_i+q_i e^{\eta_0}}\right]^{-1}.$$

De plus, $e^{\eta_0} = \tau/\bar{q} + o(\tau/\bar{q})$, et donc,

$$\begin{aligned} f'(\eta_0) &= \left[\sum_{i=1}^{d} q_i^2 e^{\eta_0}(1+o(1))\right]\cdot\left[\sum_{i=1}^{d} q_i(1-o(1))\right]^{-1} \\ &= \left[\frac{\tau}{\bar{q}}\sum_{i=1}^{d} q_i^2(1+o(1))\right]\cdot\left[d\bar{q} - o(d\bar{q})\right]^{-1} \\ &= \left[\frac{d\tau}{\bar{q}}m_2 + o\left(\frac{d\tau}{\bar{q}}m_2\right)\right]\cdot(d\bar{q})^{-1}\left[1+o(1)\right]. \end{aligned}$$

Ceci mène à l'égalité suivante

$$f'(\eta_0) = \tau\frac{m_2}{\bar{q}^2} + o\left(\tau\frac{m_2}{\bar{q}^2}\right).$$

On utilise alors le fait que $\frac{1}{1-f'(\eta_0)} = \mathcal{O}(1)$ avec lemme B.11, pour obtenir

$$\eta_r = \eta_0 + \mathcal{O}\left(\frac{f(\eta_0)-\eta_0}{1-f'(\eta_0)}\right) = \eta_0 + \mathcal{O}\left(\frac{\tau-\bar{q}}{\bar{q}^2}\cdot(\bar{q}^2-m_2)\right).$$

□

Lemme B.13. *En utilisant les notations définies plus haut,*

$$\begin{aligned}\mu(\eta_r) &= d\ln\left(\frac{1-\bar{q}}{1-\tau}\right) + \mathcal{O}\left(d\frac{(\tau-\bar{q})}{\bar{q}^2}\cdot(\bar{q}^2-m_2)\max(\tau-\bar{q},\tau)\right),\\ 1-f'(\eta_r) &= 1-\tau+\mathcal{O}\left(\frac{\max(\tau-\bar{q},\tau)}{\bar{q}^2}(\bar{q}^2-m_2)\right).\end{aligned}$$

Démonstration : En utilisant le lemme B.12, on obtient $e^{\eta_r} = e^{\eta_0}\times e^{\mathcal{O}\left(\frac{\tau-\bar{q}}{\bar{q}^2}\cdot(\bar{q}^2-m_2)\right)}$ et donc $e^{\eta_r} = e^{\eta_0}\left[1+\mathcal{O}\left(\frac{\tau-\bar{q}}{\bar{q}^2}\cdot(\bar{q}^2-m_2)\right)\right]$. C'est pourquoi

$$\begin{aligned}\mu(\eta_r) &= \sum_{i=1}^{d}\ln\left(1-q_i+q_ie^{\eta_r}\right)\\ &= \sum_{i=1}^{d}\ln\left(1-q_i+q_ie^{\eta_0}\left[1+\mathcal{O}\left(\frac{\tau-\bar{q}}{\bar{q}^2}\cdot(\bar{q}^2-m_2)\right)\right]\right)\\ &= \sum_{i=1}^{d}\ln\left(\left[1-q_i+q_ie^{\eta_0}\right]\left[1+\mathcal{O}\left(\frac{q_i}{\bar{q}}\frac{\tau}{\bar{q}}\frac{\tau-\bar{q}}{\bar{q}^2}\cdot(\bar{q}^2-m_2)\right)\right]\right)\\ &= \sum_{i=1}^{d}\ln\left(1-q_i+q_ie^{\eta_0}\right)+\mathcal{O}\left(d\tau\cdot\frac{\tau-\bar{q}}{\bar{q}^2}\cdot(\bar{q}^2-m_2)\right).\end{aligned}$$

Et finalement,

$$\mu(\eta_r) = \mu(\eta_0) + \mathcal{O}\left(d\tau\cdot\frac{\tau-\bar{q}}{\bar{q}^2}\cdot(\bar{q}^2-m_2)\right).$$

En ce qui concerne $\mu(\eta_0)$,

$$\begin{aligned}\mu(\eta_0) &= \sum_{i=1}^{d}\ln\left(1-q_i+q_ie^{\eta_0}\right)\\ &= \sum_{i=1}^{d}\ln\left((1-q_i)\bar{q}(1-\tau)+q_i(1-\bar{q})\tau\right) - d\ln\left(\bar{q}(1-\tau)\right)\\ &= \sum_{i=1}^{d}\ln\left(\bar{q}-q_i\bar{q}-\bar{q}\tau+q_i\tau\right) - d\ln\left(\bar{q}(1-\tau)\right)\\ &= \sum_{i=1}^{d}\ln\left(\frac{\bar{q}(1-\tau)+q_i(\tau-\bar{q})}{(1-\bar{q})\bar{q}}\right) - d\ln\left(\frac{1-\tau}{1-\bar{q}}\right)\end{aligned}$$

$$\begin{aligned}\mu(\eta_0) &= d\ln\left(\frac{1-\bar{q}}{1-\tau}\right)+\sum_{i=1}^{d}\ln\left(1+\frac{(q_i-\bar{q})(\tau-\bar{q})}{(1-\bar{q})\bar{q}}\right)\\ &= d\ln\left(\frac{1-\bar{q}}{1-\tau}\right)+\sum_{i=1}^{d}\frac{(q_i-\bar{q})(\tau-\bar{q})}{(1-\bar{q})\bar{q}}+\mathcal{O}\left(d\frac{(\tau-\bar{q})^2}{\bar{q}^2}\cdot(\bar{q}^2-m_2)\right)\\ &= d\ln\left(\frac{1-\bar{q}}{1-\tau}\right)+\mathcal{O}\left(d\frac{(\tau-\bar{q})^2}{\bar{q}^2}\cdot(\bar{q}^2-m_2)\right).\end{aligned}$$

Et donc

$$\mu(\eta_r)=d\ln\left(\frac{1-\bar{q}}{1-\tau}\right)+\mathcal{O}\left(d\frac{(\tau-\bar{q})}{\bar{q}^2}\cdot(\bar{q}^2-m_2)\max(\tau-\bar{q},\tau)\right).$$

La seconde partie de ce lemme est donnée par la série de Taylor de $f'(\eta_r)$:

$$\begin{aligned}f'(\eta_r) &= f'(\eta_0)+\mathcal{O}(\eta_0-\eta_r)\\ &= \tau\frac{m_2}{\bar{q}^2}+\mathcal{O}\left(\frac{\tau-\bar{q}}{\bar{q}^2}(\bar{q}^2-m_2)\right).\end{aligned}$$

Ce qui implique que

$$\begin{aligned}1-f'(\eta_r) &= (1-\tau)\left[1+\mathcal{O}\left(\frac{\tau(\bar{q}^2-m_2)}{\bar{q}^2(1-\tau)}\right)\right]+\mathcal{O}\left(\frac{\tau-\bar{q}}{\bar{q}^2}(\bar{q}^2-m_2)\right)\\ &= (1-\tau)+\mathcal{O}\left(\frac{\max(\tau-\bar{q},\tau)}{\bar{q}^2}(\bar{q}^2-m_2)\right).\end{aligned}$$

□

Preuve du théorème 9.7. On revient maintenant à la preuve du théorème 9.7. On utilisera (B.20), lemme B.12 et lemme B.13. On considère maintenant que τ est supérieur à $\bar{q}$ et on cherche à estimer $\Pr[D(\boldsymbol{k})\geq d\tau N/2]$ le même raisonnement mène au résultat voulu pour $\Pr[D(\boldsymbol{k})\leq d\tau N/2]$. On commence par s'occuper du terme exponentiel.

$$\begin{aligned}e^{\frac{N}{2}[\mu(\eta_r)-\eta_r d\tau]} &= \exp\left[\frac{N}{2}d\ln\left(\frac{1-\bar{q}}{1-\tau}\right)-\frac{N}{2}\ln\left(\frac{\tau(1-\bar{q})}{\bar{q}(1-\tau)}\right)d\tau+\mathcal{O}\left(N\,d\tau\frac{\tau-\bar{q}}{\bar{q}^2}(\bar{q}^2-m_2)\right)\right]\\ &= e^{-d\frac{N}{2}D(\tau||\bar{q})}\left[1+\mathcal{O}\left(N\,d\tau\frac{\tau-\bar{q}}{\bar{q}^2}(\bar{q}^2-m_2)\right)\right].\end{aligned}$$

On s'intéresse ensuite au terme polynomial.

$$\begin{aligned}\left[|\eta_r|\sqrt{\pi N\,d\tau(1-f'(\eta_r))}\right]^{-1} &= \left[|\eta_0|+\mathcal{O}\left(\frac{\tau-\bar{q}}{\bar{q}^2}(\bar{q}^2-m_2)\right)\right]^{-1}\\ &\times\left[\sqrt{\pi N\,d\tau}\sqrt{1-\tau+\mathcal{O}\left(\frac{\tau}{\bar{q}^2}(\bar{q}^2-m_2)\right)}\right]^{-1}\\ &= \left[\eta_0\sqrt{\pi N\,d\tau(1-\tau)}\left[1+\mathcal{O}\left(\frac{\tau}{\bar{q}^2}(\bar{q}^2-m_2)\right)\right]\right]^{-1}.\end{aligned}$$

Comme on a supposé τ supérieur à $\bar{q}$, η_0 est positif et,

$$\begin{aligned}\eta_0 &\stackrel{\text{def}}{=} -\ln\left(\frac{\bar{q}(1-\tau)}{\tau(1-\bar{q})}\right) = -\ln\left(1-\frac{\tau-\bar{q}}{\tau(1-\bar{q})}\right)\\ \eta_0 &= \frac{\tau-\bar{q}}{\tau(1-\bar{q})}\left[1+\frac{\tau-\bar{q}}{2\tau(1-\bar{q})}+o\left(\frac{\tau-\bar{q}}{\tau}\right)\right].\end{aligned}$$

Donc,

$$\begin{aligned}\left[\eta_r\sqrt{\pi N\, d\tau(1-f'(\eta_r))}\right]^{-1} &= \frac{1+\mathcal{O}\left(\frac{\tau}{\bar{q}^2}(\bar{q}^2-m_2)\right)}{\eta_0\sqrt{\pi N\, d\tau(1-\tau)}}\\ &= \frac{\tau(1-\bar{q})}{(\tau-\bar{q})\sqrt{\pi N\, d\tau(1-\tau)}}\cdot\left[1+\frac{(\tau-\bar{q})}{2\tau(1-\bar{q})}+o\left(\frac{\tau-\bar{q}}{\tau}\right)\right]\\ &= \frac{\sqrt{\tau}(1-\bar{q})}{(\tau-\bar{q})\sqrt{\pi N\, d(1-\tau)}}+\frac{1}{\sqrt{8\pi t}}+o\left(\frac{1}{\sqrt{t}}\right).\end{aligned}$$

Pour conclure, on injecte ces deux termes dans (B.20) et on obtient

$$\begin{aligned}\Pr\left[D(\boldsymbol{k})\geq d\tau\frac{N}{2}\right] &= \left[\frac{\sqrt{\tau}(1-\bar{q})}{(\tau-\bar{q})\sqrt{\pi N\, d(1-\tau)}}+\frac{1}{\sqrt{8\pi t}}+o\left(\frac{1}{\sqrt{t}}+\frac{1}{\sqrt{N}}\right)\right]\\ &\times\ e^{-d\frac{N}{2}D(\tau||\bar{q})}\cdot\left[1+\mathcal{O}\left(N\, d\tau\frac{\tau-\bar{q}}{\bar{q}^2}(\bar{q}^2-m_2)\right)\right]\\ &= e^{-d\frac{N}{2}D(\tau||\bar{q})}\left[\frac{\sqrt{\tau}(1-\bar{q})}{(\tau-\bar{q})\sqrt{\pi N\, d(1-\tau)}}+\frac{1}{\sqrt{8\pi t}}+o\left(\frac{1}{\sqrt{t}}\right)\right].\end{aligned}$$

□

Bibliographie

[ABK00] Ross J. Anderson, Eli Biham et Lars R. Knudsen : The case for serpent. *In AES Candidate Conference*, pages 349–354, 2000. 1.4.1

[AC09] Martin Albrecht et Carlos Cid : Algebraic techniques in differential cryptanalysis. *In* Orr Dunkelman, éditeur : *Fast Software Encryption - FSE 2009*, volume 5665 de *LNCS*, pages 193–208. Springer, 2009. 2.2.3, 2.2.3

[AG89] R. Arriata et L. Gordon : Tutorial on large deviations for the binomial distribution. *Bulletin of Mathematical Biology*, 51(1):125–131, 1989. 6.3.3, 9.2.1

[BBS99] Eli Biham, Alex Biryukov et Adi Shamir : Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. *In* Jacques Stern, éditeur : *Advances in Cryptology - EUROCRYPT 1999*, volume 1592 de *LNCS*, pages 12–23. Springer, 1999. 5.2.2

[BDCQ04] Alex Biryukov, Christophe De Cannière et Michaël Quisquater : On multiple linear approximations. *In* Yvo Desmedt, éditeur : *Advances in Cryptology - CRYPTO 2004*, volume 3152 de *LNCS*, pages 1–22. Springer, 2004. 1.5, 4.2, 4.2.3, 4.12, 4.2.4, 4.2.4, 4.6, 4.7, 8, 8.1, 8.1.3, 8.1.3, 8.1.3

[BDK02] Eli Biham, Orr Dunkelman et Nathan Keller : Enhancing differential-linear cryptanalysis. *In* Yuliang Zheng, éditeur : *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 de *LNCS*, pages 254–266. Springer, 2002. 5.2.4, 5.2.4

[BDK03] Eli Biham, Orr Dunkelman et Nathan Keller : Differential-linear cryptanalysis of SERPENT. *In* Thomas Johansson, éditeur : *Fast Software Encryption - FSE 2003*, LNCS, pages 9–21. Springer, 2003. 5.2.4, 5.2.4

[BF07] Thomas Baignères et Matthieu Finiasz : Dial `c` for cipher. *In* Eli Biham et Amr M. Youssef, éditeurs : *Selected Areas in Cryptography - SAC 2006*, volume 4356 de *LNCS*, pages 76–95. Springer, 2007. 1.4.1

[BG09a] Céline Blondeau et Benoît Gérard : On the data complexity of statistical attacks against block ciphers. EUROCRYPT 2009 POSTERSESSION, 2009. (document)

[BG09b] Céline Blondeau et Benoît Gérard : On the data complexity of statistical attacks against block ciphers (full version). *In* Alexander Kholosha, Eirik Rosnes et Matthew G. Parker, éditeurs : *Workshop on Coding and Cryptography - WCC 2009*, pages 469–488, 2009. (document), 6

[BG10] Céline Blondeau et Benoît Gérard : Links between theoretical and effective differential probabilities : Experiments on PRESENT. *In TOOLS'10*, 2010. `http://eprint.iacr.org/2010/261`. (document), 2.2.2, 5.3.2, 5.3.3

[BGT10] Céline Blondeau, Benoît Gérard et Jean-Pierre Tillich : Accurate estimates of the data complexity and success probability for various cryptanalyses. *DCC special issue on Coding and Cryptography*, 2010. À paraître. (document), 6, 6.4.3

[BH86] Norman Bleistein et Richard A. Handelsman : *Asymptotic Expansions of Integrals*. Dover Publications, 1986. B.2

[BJV04] Thomas Baignères, Pascal Junod et Serge Vaudenay : How far can we go beyond linear cryptanalysis ? *In* Pil Joong Lee, éditeur : *Advances in Cryptology - ASIACRYPT 2004*, volume 3329 de *LNCS*, pages 432–450. Springer, 2004. 6.2.2, 6.3.1, 6.4.4, 6.4.4, 6.4.4, 6.4.5, 6.4.4, 6.4.4, A.2

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin et Charlotte H. Vikkelsoe : PRESENT : An ultra-lightweight block cipher. *In* Pascal Paillier et Ingrid Verbauwhede, éditeurs : *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 de *LNCS*, pages 450–466. Springer, 2007. 1.4.1, 2.2

[BS91a] Eli Biham et Adi Shamir : Differential cryptanalysis of DES-like cryptosystems. *In* Alfred Menezes et Scott A. Vanstone, éditeurs : *Advances in Cryptology - CRYPTO 1990*, volume 537 de *LNCS*, pages 2–21. Springer, 1991. 2.1.3, 5, 5.1, 5.1.1, 5.1.2, 6.2.1

[BS91b] Eli Biham et Adi Shamir : Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991. 2.1.3, 5, 5.1, 5.1.1, 5.1.3, 6.2.1

[BS93] Eli Biham et Adi Shamir : Differential cryptanalysis of the full 16-round DES. *In* Ernest F. Brickell, éditeur : *Advances in Cryptology - CRYPTO 1992*, volume 740 de *LNCS*, pages 487–496. Springer, 1993. 2.1.3

[BV08] Thomas Baignères et Serge Vaudenay : The complexity of distinguishing distributions. *In* Reihaneh Safavi-Naini, éditeur : *Information Theoretic Security International Conference - ICITS 2008*, volume 5155 de *LNCS*, pages 210–222. Springer, 2008. 6.2.2, 6.3.1, 6.4.5

[Cho10] Joo Yeon Cho : Linear cryptanalysis of reduced-round PRESENT. *In* Josef Pieprzyk, éditeur : *Topics in Cryptology - CT-RSA 2010*, volume 5985 de *LNCS*, pages 302–317. Springer, 2010. 2.2.3, 4.3.2, 4.3.4

[CHW08] Huiju Cheng, Howard M. Heys et Cheng Wang : PUFFIN : A novel compact block cipher targeted to embedded digital systems. *In Proceedings of Euromicro Conference on Digital System Design - DSD 2008*, 2008. 1.4.1

[CS09] Baudoin Collard et François-Xavier Standaert : A statistical saturation attack against the block cipher PRESENT. *In* Marc Fischlin, éditeur : *Topics in Cryptology - CT-RSA 2009*, volume 5473 de *LNCS*, pages 195–210. Springer, 2009. 2.2.3, 2.2.3

[CSQ07] Baudoin Collard, François-Xavier Standaert et Jean-Jacques Quisquater : Improving the time complexity of Matsui's linear cryptanalysis. *In* Kil-Hyun Nam et Gwangsoo Rhee, éditeurs : *Information Security and Cryptology - ICISC 2007*, volume 4817 de *LNCS*, pages 77–88. Springer, 2007. 4.6, 8.2.4

[CT91] Thomas M. Cover et Joy A. Thomas : *Information theory*. Wiley series in communications. Wiley, 1991. 3.12, 3.2.1, 8.2, 8.1.1, 8.1.3, A.2, B.3

[DN03] Herbert A. David et Haikady N. Nagaraja : *Order Statistics (third edition)*. Wiley series in Probability Theory. 2003. A.3

[DR00] Joan Daemen et Vincent Rijmen : Rijndael for AES. *In AES Candidate Conference*, pages 343–348, 2000. 1.4.1

[DR07] Joan Daemen et Vincent Rijmen : Probability distributions of correlation and differentials in block ciphers. *Journal of Mathematical Cryptology*, 1:12–35, 2007. 5.3.3, 5.3.3

[DWS10] Zhenli Dai, Meiqin Wang et Yue Sun : Effect of the dependent paths in linear hull. Cryptology ePrint Archive, Report 2010/325, 2010. 4.5.2

[ER10] Jonathan Etrog et Matthew J. B. Robshaw : On unbiased linear approximations. *In* Ron Steinfeld et Philip Hawkes, éditeurs : *Information Security and Privacy - ACISP 2010*, LNCS. Springer, 2010. 4.4, 4.4.2, 4.4.2, 8, 4.4.3, 4.4.3

[Fel68] William Feller : *An introduction to probability theory and its applications*, volume 1. John Wiley and Sons, 3rd édition, 1968. 8.1.3

[FLT09a] Rafaël Fourquet, Pierre Loidreau et Cédric Tavernier : Finding good linear approximations of bock ciphers and its application to cryptanalysis of reduced round DES. *In* Alexander Kholosha, Eirik Rosnes et Matthew G. Parker, éditeurs : *Workshop on Coding and Cryptography - WCC 2009*, pages 501–515, 2009. 4.7, 4.7.2

[FLT09b] Rafaël Fourquet, Pierre Loidreau et Cédric Tavernier : Finding good linear approximations of bock ciphers and its application to cryptanalysis of reduced round DES. EUROCRYPT 2009 POSTERSESSION, 2009. 4.7, 4.7.2

[Gal68] Robert G. Gallager : *Information Theory and Reliable Communication*. John Wiley and Sons, 1968. 9.2.1, 9.2.1, 9.6

[Gil97] Henri Gilbert : *Cryptanalyse statistique des algorithmes de chiffrement et sécurité des schémas d'authentification*. Thèse de doctorat, Université Paris 11 Orsay, 1997. 5.1.3, 6.2.1

[GT09] Benoît Gérard et Jean-Pierre Tillich : On linear cryptanalysis with many linear approximations. *In* Matthew G. Parker, éditeur : *IMA International Conference, Cryptography and Coding - IMACC 2009*, volume 5921 de *LNCS*, pages 112–132. Springer, 2009. (document), 8

[Gér07] Benoît Gérard : Utilisation de techniques de codage correcteur d'erreurs pour la cryptanalyse de sytèmes de chiffrement à clef secrète. Mémoire de D.E.A., Université de Versailles - Saint-Quentin en Yvelines, Versailles, 2007. (document), 7.2.6

[HCN08] Miia Hermelin, Joo Yeon Cho et Kaisa Nyberg : Multidimensional linear cryptanalysis of reduced round Serpent. *In* Yi Mu, Willy Susilo et Jennifer Seberry, éditeurs : *Information Security and Privacy - ACISP 2008*, volume 5107 de *LNCS*, pages 203–215. Springer, 2008. 2.2.3, 4.3.2, 4.3.3

[HCN09a] Miia Hermelin, Joo Yeon Cho et Kaisa Nyberg : Multidimensional extension of Matsui's algorithm 2. *In* Orr Dunkelman, éditeur : *Fast Software Encryption - FSE 2009*, volume 5665 de *LNCS*, pages 209–227. Springer, 2009. (document), 2.2.3, 4.3.2, 4.3.4, 4.6, 8.1.3, 8.1.3, 8.2, B.4

[HCN09b] Miia Hermelin, Joo Yeon Cho et Kaisa Nyberg : A new technique for multidimensional linear cryptanalysis with applications on reduced round Serpent. *In* Pil Joong Lee et Jung Hee Cheon, éditeurs : *Information Security and Cryptology - ICISC 2008*, volume 5461 de *LNCS*, pages 383–398. Springer, 2009. 2.2.3, 4.3.2

[HCN09c] Miia Hermelin, Joo Yeon Cho et Kaisa Nyberg : Statistical tests for key recovery using multidimensional extension of Matsui's algorithm 1. EUROCRYPT 2009 POSTERSESSION, 2009. 2.2.3, 4.3.2, 4.3.3, 4.6

[Her10] Miia Hermelin : *Multidimensional Linear Cryptanalysis*. Thèse de doctorat, Teknillinen Korkeakoulu, Helsinki, 2010. 4.3.2, 4.3.3, 4.6

[HN08] Miia Hermelin et Kaisa Nyberg : Multidimensional linear distinguishing attacks and boolean functions. *In International Workshop on Boolean Functions : Cryptography and Applications - BFCA 2008*, 2008. 4.3.2

[HN10] Miia Hermelin et Kaisa Nyberg : Dependent linear approximation : The algorithm of Biryukov and others revisited. *In* Josef Pieprzyk, éditeur : *Topics in Cryptology - CT-RSA 2010*, volume 5985 de *LNCS*, pages 318–333. Springer, 2010. 4.3.2, 4.3.3, 4.6

[Jun01] Pascal Junod : On the complexity of Matsui's attack. *In* Serge Vaudenay et Amr M. Youssef, éditeurs : *Selected Areas in Cryptography - SAC 2001*, volume 2259 de *LNCS*, pages 199–211. Springer, 2001. 4.8, 4.2.4, 6.2.2, 8.1, 8.1.3, 8.1.3, 8.2.4

[Jun03] Pascal Junod : On the optimality of linear, differential, and sequential distinguishers. *In* Eli Biham, éditeur : *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 de *LNCS*, pages 17–32. Springer, 2003. 6.2.2, 6.3.1

[JV03] Pascal Junod et Serge Vaudenay : Optimal key ranking procedures in a statistical cryptanalysis. *In* Thomas Johansson, éditeur : *Fast Software Encryption - FSE 2003*, volume 2887 de *LNCS*, pages 235–246. Springer, 2003. 4.2, 6.2.2

[Ker83] Auguste Kerckhoffs : La cryptologie militaire. *Journal des sciences militaires*, IX (Janvier-Février):5–38, 161–191, 1883. 1.3

[Knu95] Lars R. Knudsen : Truncated and higher order differentials. *In* PRENEEL, éditeur : *Fast Software Encryption - FSE 1994*, volume 1008 de *LNCS*, pages 196–211. Springer, 1995. 5.2.1, 5.2.3

[KR94] Burton S. Kaliski et Matthew J. B. Robshaw : Linear cryptanalysis using multiple approximations. *In* Yvo Desmedt, éditeur : *Advances in Cryptology - CRYPTO 1994*, volume 839 de *LNCS*, pages 26–39. Springer, 1994. 4.2

[Lai94] Xuejia Lai : Higher order derivatives and differential cryptanalysis. *In Symposium on Communication, Coding and Cryptography*, 1994. 5.2.3, 5.7

[Lea10] Gregor Leander : Small scale variants of the block cipher PRESENT. Cryptology ePrint Archive, Report 2010/143, 2010. http://eprint.iacr.org/2010/143. 2.2.2, 1, 9.2.3

[LH94] Susan K. Langford et Martin E. Hellman : Differential-linear cryptanalysis. *In* Yvo Desmedt, éditeur : *Advances in Cryptology - CRYPTO 1994*, volume 839 de *LNCS*, pages 17–25. Springer, 1994. 5.2.4, 5.2.4

[LMM91] Xuejia LAI, James L. MASSEY et Sean MURPHY : Markov ciphers and differential cryptanalysis. *In* Donald W. DAVIES, éditeur : *Advances in Cryptology - EUROCRYPT 1991*, volume 547 de *LNCS*, pages 17–38. Springer, 1991. 5.3.2, 5.11, 6.4.5

[Mat94a] Mitsuru MATSUI : The first experimental cryptanalysis of the data encryption standard. *In* Yvo DESMEDT, éditeur : *Advances in Cryptology - CRYPTO 1994*, volume 839 de *LNCS*, pages 1–11. Springer, 1994. 4, 4.2, 6.2.2, 6.4.5, 8.1.3, 8.2, 8.2.4

[Mat94b] Mitsuru MATSUI : Linear cryptanalysis method for DES cipher. *In* Tor HELLESETH, éditeur : *Advances in Cryptology - EUROCRYPT 1993*, volume 765 de *LNCS*, pages 386–397. Springer, 1994. 2.1.3, 4, 2, 4.1.1, 4.1.2, 4.1.3, 4.6, 4.1.4, 4.1.4, 4.5.1, 4.6, 6.2.2, 6.4.5

[Mat95] Mitsuru MATSUI : On correlation between the order of S-boxes and the strength of DES. *In* Alfredo DE SANTIS, éditeur : *Advances in Cryptology - EUROCRYPT 1994*, volume 950 de *LNCS*, pages 366–375. Springer, 1995. 4, 4.7

[Mur06] Sean MURPHY : The independence of linear approximations in symmetric cryptology. *IEEE Transactions on Information Theory*, 52:5510–5518, 2006. 4.4, 4.4.1

[Mur09a] Sean MURPHY : The effectiveness of the linear hull effect. Rapport technique RHUL-MA-2009-19, Royal Holloway, 2009. 4.5.1, 4.5.3

[Mur09b] Sean MURPHY : Overestimates for the gain of multiple linear approximations. Rapport technique RHUL-MA-2009-21, Royal Holloway, 2009. 4.2.4

[NH07] Kaisa NYBERG et Miia HERMELIN : Multidimensional Walsh transform and a characterization of bent functions. *In* P. Vijay KUMAR, Tor HELLESETH et Oyvind YTREHUS, éditeurs : *IEEE Information Theory Workshop - ITW 2007*, pages 83–86, 2007. 4.3.2, 4.19

[NSZW09] Jorge NAKAHARA, Pouyan SEPEHRDAD, Bingsheng ZHANG et Meiqin WANG : Linear (hull) and algebraic cryptanalysis of the block cipher PRESENT. *In* Akira OTSUKA, éditeur : *International Conference on Cryptology And Network Security - CANS 2009*, volume 5888 de *LNCS*, pages 58–75. Springer, 2009. 2.2.3, 2.2.3, 4, 4.5.3, B.4

[Nyb95] Kaisa NYBERG : Linear approximation of block ciphers. *In* Alfredo DE SANTIS, éditeur : *Advances in Cryptology - EUROCRYPT 1994*, volume 950 de *LNCS*. Springer, 1995. 4.5.1

[Nyb96] Kaisa NYBERG : Generalized feistel networks. *In* Kwangjo KIM et Tsutomu MATSUMOTO, éditeurs : *Advances in Cryptology - ASIACRYPT 1996*, volume 1163 de *LNCS*, pages 91–104. Springer, 1996. 1.4.2

[Nyb01] Kaisa NYBERG : Correlation theorems in cryptanalysis. *Discrete Applied Mathematics*, 111(1-2):177–188, 2001. 4.5.1, 4.24, 4.5.1

[Ohk09] Kenji OHKUMA : Weak keys of reduced-round PRESENT for linear cryptanalysis. *In* Michael J. Jacobson JR., Vincent RIJMEN et Reihaneh SAFAVI-NAINI, éditeurs : *Selected Areas in Cryptography - SAC 2009*, volume 5867 de *LNCS*, pages 249–265. Springer, 2009. 2.2.3, 2.2.3, 4, 4.5.2, B.4

[ÖVTK09] Onur ÖZEN, Kerem VARICI, Cihangir TEZCAN et Çelebi KOCAIR : Lightweight block ciphers revisited : Cryptanalysis of reduced round PRESENT and HIGHT.

In Colin Boyd et Juan Manuel Gonzáles Nieto, éditeurs : *Information Security and Privacy - ACISP 2009*, volume 5594 de *LNCS*, pages 90–107. Springer, 2009. 2.2.3, 2.2.3

[Pat04] Jacques Patarin : Security of random feistel schemes with 5 or more rounds. *In* Yvo Desmedt, éditeur : *Advances in Cryptology - CRYPTO 2004*, volume 3152 de *LNCS*, pages 106–122. Springer, 2004. 1.4.2

[Roc10] Thomas Roche : *Dimensionnement et intégration d'un chiffre symétrique dans le contexte d'un système d'information distribué de grande taille*. Thèse de doctorat, Université de Grenoble, 2010. 4.7.2

[RT09] Thomas Roche et Cédric Tavernier : Side-channel attacks based on linear approximations. *In International Alternative Workshop on Aggressive Computing and Security - iAWACS 2009*. ISBN Presses Techniques ESIEA, 2009. 4.7, 4.7.2

[RU08] Thomas Joseph Richardson et Rüdiger Leo Urbanke : *Modern Coding Theory*. Cambridge University Press, 2008. 3.1.3, 3.14

[Rén07] Alfréd Rényi : *Probability Theory*. Dover ed edition édition, 2007. 4.10

[Sel08] Ali Aydın Selçuk : On probability of success in linear and differential cryptanalysis. *Journal of Cryptology*, 21(1):131–147, 2008. (document), 1.5, 4.9, 4.6, 5.1.3, 6.2.1, 6.2.2, 6.2.3, 6.4.4, 6.4.4, 6.4.4, 6.4.4, 6.5, 6.5, 6.5.3, 6.2, 5, 9.1.3, 9.2.3, B.4

[She06] Irina Gennad'evna Shevtsova : Sharpening of the upper bound of the absolute constant in the Berry-Esséen inequality. *Theory of Probability and its Applications*, 51:549–553, 2006. A.1.6

[SK96] Bruce Schneier et J. Kelsey : Unbalanced feistel networks and block cipher design. *In* Dieter Gollmann, éditeur : *Fast Software Encryption - FSE 1996*, volume 1039 de *LNCS*, pages 121–144. Springer, 1996. 1.4.2

[Tav04] Cédric Tavernier : *Testeurs, problèmes de reconstruction univariés et multivariés, et application à la cryptanalyse du DES*. Thèse de doctorat, Ecole Polytechnique, 2004. 4.7.2

[TCG92] Anne Tardy-Corfdir et Henri Gilbert : A known plaintext attack of FEAL-4 and FEAL-6. *In* Joan Feigenbaum, éditeur : *Advances in Cryptology - CRYPTO 1991*, volume 576 de *LNCS*, pages 172–181. Springer, 1992. 2.1.3, 4

[Val00a] Antoine Valembois : *Détection, Reconnaissance et Décodage des Codes Linéaires Binaires*. Thèse de doctorat, Université de Limoges, 2000. 7, 7.2.5

[Val00b] Antoine Valembois : Fast soft-decision decoding of linear codes, stochastic resonance in algorithms. *In IEEE International Symposium on Information Theory - ISIT 2000*, page 91, 2000. 7, 7.2.5

[Vid05] Marion Videau : *Critères de Sécurité des algorithmes de Chiffrement à Clé Secrète*. Thèse de doctorat, Université Pierre et Marie Curie (Paris VI), 2005. 5.2

[Wan08] Meiqin Wang : Differential cryptanalysis of reduced-round PRESENT. *In* Serge Vaudenay, éditeur : *Progress in Cryptology - AFRICACRYPT 2008*, volume 5023 de *LNCS*, pages 40–49. Springer, 2008. 2.2.3, 2.2.3, 5.1, 5.3.1, 5.2, 9.1.3, 9.2.3, 2, B.4

[ZRHD08] Muhammad R. Z'aba, Håvard Raddum, Matthew Henricksen et Ed Dawson : Bit-pattern based integral attack. *In* Kaisa Nyberg, éditeur : *Fast Software*

Encryption - FSE 2008, volume 5086 de *LNCS*, pages 363–381. Springer, 2008. 2.2.3, 2.2.3

Index

Table des figures

Liste des Algorithmes

Liste des tableaux

Table des matières