



Analyse de canaux de communication dans un contexte non coopératif.

Johann Barbier

► To cite this version:

Johann Barbier. Analyse de canaux de communication dans un contexte non coopératif.. Informatique [cs]. Ecole Polytechnique X, 2007. Français. NNT: . pastel-00003711

HAL Id: pastel-00003711

<https://pastel.hal.science/pastel-00003711>

Submitted on 22 Jul 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



THÈSE

présentée à
L'ÉCOLE POLYTECHNIQUE

pour obtenir le titre de
DOCTEUR ÈS SCIENCES

Spécialité : Informatique

soutenue par
Johann BARBIER

le 28 novembre 2007

Titre :
**Analyse de canaux de communication dans un
contexte non coopératif**

**-Application aux codes correcteurs d'erreurs -
et à la stéganalyse**

Directeurs de thèse : Robert Cori et Éric Filiol

Jury

Président	M.	Jean-Marc	STEYAERT	École Polytechnique
Rapporteurs	M.	Christian	CACHIN	IBM Research (Suisse)
	M.	David	HACCOUN	École Polytechnique de Montréal (Canada)
	M.	Basel	SOLAIMAN	ENST Brest
Examineurs	M.	Claude	BERROU	ENST Brest
	M ^{lle}	Caroline	FONTAINE	IRISA Rennes
	M.	Pierre	LOIDREAU	DGA Celar
	M.	Andreas	WESTFELD	Université de Dresden (Allemagne)

version finale

*À ma petite Lucie, source intarissable d'inspiration,
et mon épouse Katya ; mon pilier de toujours*

Remerciements

« La valeur d'un homme tient dans sa capacité à donner et non dans sa capacité à recevoir. »

Albert Einstein

AU-DELÀ de la thèse en elle-même, ces dernières années auront été pour moi une véritable aventure humaine, tant éprouvante que palpitante, qui s'est inscrite dans une mutation profonde de mon parcours professionnel. Échanger mes rangers et mon treillis contre un bureau spacieux et un PC nouvelle génération de coloration kaki à peine visible, fut un défi exaltant mais tout aussi effrayant. Loin du grand air et du commandement de terrain édifié en véritable sacerdoce, promis à la délivrance de Saint-Cyr (ou *Pékin de Bahut* (PDB) dans notre vocabulaire traditionnel), je retrouve alors les bancs de l'école pour y préparer un doctorat. Ce manuscrit m'apparaît aujourd'hui comme l'une des dernières étapes d'un parcours du combattant un peu différent que celui pour lequel j'ai été formé. Au regard des obstacles franchis, je prends maintenant la pleine mesure du soutien que chacun a pu m'apporter sur un chemin qui peut paraître sinueux à la lumière de mon parcours. C'est avec sincérité que je veux dire MERCI à toutes ces personnes qui ont contribué, de près ou de loin, à cette expérience très personnelle que peut représenter une thèse. Loin de moi l'idée de hiérarchiser mes remerciements, je suivrai donc un fil chronologique.

Tout d'abord, je voudrais remercier Éric Filiol. Plus qu'un directeur de thèse, il été d'abord mon professeur mais aussi un guide avisé dans les méandres de l'administration militaire. Ses conseils judicieux m'ont permis de prendre du recul pour faire les bons choix professionnels. Dans les moments de doute, il a toujours pris sur son temps précieux pour m'écouter amicalement et m'aider à trouver ma place entre une communauté qui n'était plus la mienne et une autre qui ne l'était pas encore. Son investissement personnel et sa capacité de travail sont pour moi un véritable modèle. Je voudrais aussi le remercier pour sa franchise, ou ce qu'il aime appeler son « mauvais caractère », plaçant ainsi nos rapports dans un contexte sain, sans ambiguïté et sans non-dit. Sa passion, son pragmatisme et ses idées originales rendent nos discussions fructueuses et c'est un réel plaisir de collaborer avec Éric.

Je voudrais ensuite remercier Basel Solaiman, tout d'abord pour avoir pris le temps de rapporter ma thèse malgré un emploi du temps extrêmement chargé. Son charisme, sa

pédagogie et sa passion pour la Science, ont érigé son cours de *Traitement de l'Information* à Saint-Cyr en modèle de présentation scientifique. Je le remercie pour m'avoir accueilli à l'ENSTB pendant mes permissions et surtout initié aux joies de la recherche. Sa bienveillance et sa simplicité stimulent des discussions passionnantes et accessibles. Favorisant toujours la mise en commun de volontés plus que de compétences, il m'a convaincu de la puissance de l'interdisciplinarité.

Je remercie aussi Jean-Marc Steyaert et François Morain pour m'avoir ouvert les portes du Lix et Robert Cori pour avoir co-encadré ma thèse. Ce projet n'aurait pas vu le jour sans la confiance qu'ils ont placée en moi. Encadrants exigeants lors de mes stages de recherche, ils illustrent de mon point de vue, l'excellence scientifique. Ils m'ont apporté les bases et la méthodologie nécessaires à tout travaux de recherche. Je remercie notamment Jean-Marc de m'avoir fait l'honneur de faire partie de mon jury de thèse.

Je remercie aussi Thomas Sirvent, camarade de DEA hier, étudiant, collègue et ami aujourd'hui. C'est un des rares à avoir du subir mes états d'âmes. Toujours compatissant et neutre à la fois, ses conseils sont toujours empreints d'une grande maturité et m'ont ainsi permis d'avancer avec un regard neuf. Ses remarques averties et son sens de la pédagogie rendent les discussions tant passionnantes qu'instructives. Son secours m'a été maintes fois précieux notamment pour me sortir de mauvais pas.

Je remercie Reynald Lercier pour m'avoir permis d'inscrire cette thèse au sein du département de cryptologie du Centre d'ELectronique de l'ARmement, tout en me laissant le temps et l'autonomie nécessaires. Exigeant, il a grandement contribué à me canaliser sur mon objectif. Il m'a par ailleurs, enseigné la rigueur scientifique et inculqué l'art du détail. Je remercie aussi André Parriel et Olivier Mangeot pour m'avoir donné ma chance et avoir porté haut le résultat de mes travaux tant au CELAR qu'au sein de la Délégation Générale pour l'Armement. Je suis particulièrement sensible à leur investissement.

Je voudrais aussi remercier tout particulièrement Emmanuel Mayer, mon camarade de bureau qui doit me supporter à longueur de journée. À la fois conseiller scientifique et technique, il endosse aussi le rôle de psychologue à ses heures perdues. Grand sensei, il prend patiemment le temps de répondre à toutes les questions existentielles que je me pose toutes les dix minutes environ. Sa pédagogie hors paire et sa rigueur me permettent d'apprendre un peu plus chaque jour les rudiments du métier. Il a joué un rôle majeur dans la réussite de cette thèse.

Bien que personnelle, une thèse est avant tout un travail qui s'inscrit au sein d'une équipe. Je remercie donc mes co-auteurs, Kichenakoumar Mayoura, Sébastien Houcke et Guillaume Sicot avec lesquels j'ai grand plaisir à collaborer et surtout à partager. Je remercie aussi l'ensemble du laboratoire EC, riche de sa diversité, il offre un cadre idéal pour se réaliser pleinement. Je remercie le Centre de Recherche de l'École Supérieure et d'Application des Transmissions pour son accueil chaleureux ainsi que monsieur Tatania pour son investissement.

Enfin, je remercie Christian Cachin et David Haccoun de l'intérêt qu'ils portent à mes travaux et de l'honneur qu'ils m'ont fait en acceptant de les rapporter. Je remercie aussi notamment Caroline Fontaine, Claude Berrou, Pierre Loidreau et Andreas Westfeld pour

l'honneur qu'ils m'ont fait en participant à mon jury de thèse.

Mais aussi un grand merci à tous ceux que j'ai malencontreusement oubliés ; ils se reconnaîtront et je l'espère me pardonneront. Je ne pouvais bien évidemment pas conclure mes remerciements sans saluer la compréhension de ma famille, ma belle-famille et de mes amis qui ont accepté mon manque de disponibilité et mes retards latents dans ma correspondance. Enfin, les militaires ont coutume de dire que « la logistique est le nerf de la guerre ». C'est donc tout naturellement un merci particulier que j'adresse à mon épouse, exemple vivant de *logistique qui vous font gagner une guerre*. J'admire son courage et son dévouement et je mesure pleinement les sacrifices, plus grands que les miens encore, auxquels elle a consenti. Enfin, un « *ssii* » à ma petite Lucie, qui donne tout son sens à ce travail.

Contributions

« Don't bother about genius. Don't worry about being clever. Trust to hard work, perseverance and determination. And the best motto for a long march is "Don't grumble. Plug on!" »

Sir Frederick Treves

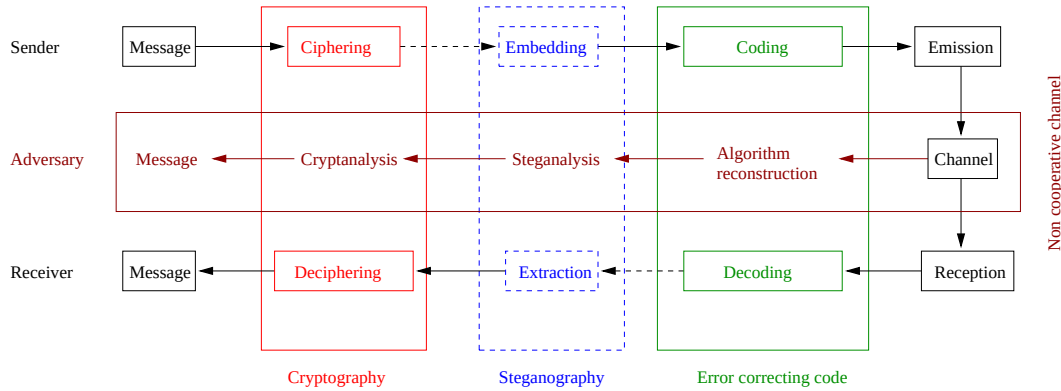


FIG. 1 – Communication scheme

IN this thesis, we play the role of a passive adversary who wants to have access to the information exchanged between two legal users, Alice and Bob. In order to protect themselves against such an eavesdropper, they use cryptography to insure communication security (COMSEC) but also error correcting codes and steganography to insure transmission security (TRANSEC) as illustrated in figure 1. In this context, the adversary has no knowledge about the communication scheme used by Alice and Bob. He has then to “guess” all the parameters of the intercepted communication in order to demodulate the intercepted signal, to decode the binary stream, to detect and extract the hidden message and finally to decrypt it. Such a context of attack, which considers a very constrained adversary, is called a *non cooperative context*. On the contrary, the context is said to be *cooperative* when users share known parameters, that is the classical way for communication. By analogy with digital communication, steganography can be considered as particular way of digital communication. In this perspective, the cover medium is the channel, the embedding algorithm is the coding and emission step, and the extraction algorithm is the

reception and decoding step. In that scenario, the adversary does not control neither the intercepted media nor the parameters used for the embedding process. Then, the medium can be considered as a non cooperative channel. The adversary has to blindly extract the hidden message before cryptanalysis it. We now make the hypothesis that the adversary has already recovered the demodulation parameters and has access to the encoded binary stream. In this thesis, we focus on two main steps of his attack which are *error correcting code reconstruction* and *steganalysis*. First, we design algorithms to retrieve the parameters of linear block codes, convolutional and turbo codes from the intercepted binary stream with only the knowledge of the type of the code. Such data processing is called *blind algorithms*. Then, we elaborate classifiers to distinguish between innocuous media, also called the *cover media* and media which hold hidden information, also called the *stego media*. If we consider cover media as transmission channels in which we “emit” hidden messages, steganalysis is also a blind algorithm. These steps are crucial before cryptanalysis, as it cannot be done before the message is extracted and decoded and the redundancy is withdrawn.

In the context of error correcting code reconstruction, the eavesdropper has access to the noisy coded binary stream. From his point of view, the modulation, the emission, the physical channel, the reception and the demodulation can be modelled by a binary symmetric channel (BSC). In the case of the observer trying to retrieve the parameters of error correcting codes which are implemented inside an equipment, the BSC is noiseless and the equipment is evaluated as a black box. In the case of interception, the observation is noisy. The noise introduced in the intercepted sequence depends on the channel but also of the modulation. It implies that we need to model accurately the transmission channel to evaluate error correcting code reconstruction algorithms. Nevertheless, we experimentally notice that the most restricting channel for such algorithms is the BSC. We analyse our algorithms under this hypothesis. Because of very specific applications, error correcting code reconstruction techniques applied to a communication scheme in a non cooperative context is not widespread in the literature. G. Planquette is perhaps the first one who dealt with binary stream analysis [154]. He explains how to detect block codes, convolutional codes and linear scramblers using two types of approaches. The first one, consists in estimating correlations found in output signals using statistical tests and hypothesis testing. The second one is algebraic and consists in looking for a variation of the rank of a matrix build with the intercepted bits. This approach is made complete with a low Hamming weight search using Leon algorithm [129] or the Canteaut-Chabaud one [42]. G. Planquette has then set up the basis of the binary stream analysis. In the same direction, A. Valembois [195, 196] formalized the problems of detection and reconstruction of the block codes following the same approaches. His results have recently been improved by M. Cluzeau [50, 51] by using an iterative decoding algorithm to correct some errors from beginning to end of the reconstruction process. In the same time, G. Burel and R. Gautier [37], for a noiseless channel, but also G. Sicot and S. Houcke [173, 174, 175], for the general case have obtained similar results by taking advantage of the resistance of the Gauss elimination algorithm when equations are noisy. Moreover, B. Rice [162] and É. Filiol [69, 70, 71] got interested more particularly in the reconstruction of convolutional codes.

In this thesis, we first improve É. Filiol’s results and then generalized them to turbo codes. In a second time, we express the algorithms of convolutional and block codes reconstruction with the same formalism. This formalism is based on linear algebra which

allows us to consider block codes as particular cases of convolutional codes [143]. We recall first the basis of this formalism for convolutional codes in chapter 1. Then, we define the linear block codes reconstruction problem using the same formalism and analyse the Sicot-Houcke algorithm in the chapter 2. We chose to adopt G. Sicot and S. Houcke's strategy rather than A. Valembois' one for two main reasons : Firstly, G. Planquette and A. Valembois' approach has recently been revisited by M. Cluzeau and secondly, the analysis of Sicot-Houcke algorithm has never been done through the scope of linear algebra before. Nevertheless, the comparison between both approaches remains and is the central point of current researches. The algorithms that we designed allows us to retrieve a basis of parity checks of the code, *i.e.* the dual code. Unfortunately, without any else hypothesis, finding a decoder is equivalent to a random code decoding problem which is known to be NP-complete [29]. The complexity of such algorithms is exponential on the Hamming weight of the parity check of highest Hamming weight but also on the length of the code and on the bit error rate (BER). The size of interleavers is then a crucial limiting factor. The results that we obtain go in the same direction as the intuitive hypothesis made in section 2.1.2 and so justify the choice of a BSC. The theoretical and experimental results illustrate that increasing the number of iterations of the Sicot-Houcke algorithm or the number of intercepted bits, increases the probability of detection. The results of this analysis has been published at the Conference on Cryptography, Coding and Information Security (CCIS'06), [19]. Moreover, an extended version will be submitted to IEEE Transactions on Computers and a synthesis has been submitted to the journal of Signal Processing.

The formalism introduced in chapter 1 leads us to discover new equations for the reconstruction of convolutional codes and then to significantly improve the theoretical complexity of É. Filiol's algorithm for both noisy and noiseless channels, but also to greatly decrease the number of intercepted bits needed for the reconstruction. Moreover, the analysis of the Sicot-Houcke algorithm made us understand why his experimental results were better than those announced by his theoretical analysis. Indeed, we start with this algorithm and take advantage of the strong algebraic structure of convolutional codes to obtain new equations to retrieve the parity checks. We noticed that these parity checks can be written as minors of given matrices that we pointed out. From that central remark, we complete É. Filiol's reconstruction technique into a completely automatic process. For a noiseless channel, we also improve B. Rice's technique to reconstruct $(n, 1)$ -convolutional codes by using the Berlekamp-Massey algorithm [28, 140]. These improvements are described in chapter 3. We give a detailed analysis of the proposed algorithms in section 3.3 and give a proof of É. Filiol's conjecture [71, p. 170] : *the output of the proposed algorithm is canonical matrices*. As for block codes reconstruction, the algorithms only allow to recover the code but not the decoder. The remaining indetermination is due to the intrinsic nature of convolutional codes. Nevertheless, in the particular case of systematic codes, the systematic bits give us an access to a decoder. In section 3.5, we explain how to reconstruct recursive or punctured convolutional codes. We consider the case of a convolutional code followed by an interleaver as a block code reconstruction problem because the interleaver "breaks" the Hankel structure of the interception matrix. We show that convolutional codes reconstruction algorithms can be considered as particular cases of algorithms to reconstruct linear block codes as opposed to linear block codes which are particular cases of convolutional codes. We published these new algorithms in SPIE Security and Defense conference in 2005 [9] and a synthesis is planned to be submitted to IEEE Transactions on Computers.

In the last chapter dedicated to error correcting code reconstruction, we generalize the former techniques to reconstruct the turbo codes. To achieve this, we design algorithms to reconstruct a block interleaver knowing its inputs and outputs. For noiseless channels, the proposed algorithm is proved to be optimal and for noisy channels, the algorithm is adapted to a BER higher than those usually considered for most transmission channels. We analyse these algorithms in section 4.3 and detail in section 4.4 the experimental results that we obtained. They have been published in MAJECSTIC'03 [14]. Then, using convolutional code reconstruction algorithms, we are able to find a decoder for the systematic convolutional code. Practically, the second convolutional coder is most of time equal to the first one. If it is not the case, we need to solve a linear system to find the appropriate decoder. After that, we decode the outputs of the convolutional encoders to obtain some noisy pairs of inputs and outputs for recovering the interleaver. The reconstruction of the interleaver is all the more efficient as some errors are corrected when decoding the outputs of the convolutional coders. Contrary to the reconstruction of block or convolutional codes, we are able to exhibit a decoder. The entire technique has been presented at SPIE Security and Defense conference in 2005 [9] and an extended version is prepared to be submitted to IEEE Transactions on Computers. In the context of this survey, we take the most restricting point of view for an observer, *i.e.* the non cooperative context. It appears that it is not always possible to find an equivalent decoder without any *a priori* knowledge. Now, the indetermination is often very easy to clear up with a small amount of information. So, it is quite natural to adapt reconstruction techniques to the cooperative context. For instance, one application consists in not sending the new coding scheme parameters when both the sender and the receiver auto adapt themselves to the channel. These new parameters are then retrieved using reconstruction algorithms. Such algorithms are then a compromise between the bandwidth and payload. Moreover, when latency is not crucial one can imagine to encode additional information within the choice of given parameters of the coding scheme, such as the interleaver in a turbo code. In that cases, reconstruction algorithms can be considered as decoding algorithms and can also be integrated inside an error correcting code scheme. These adaptations are detailed in the patents that we have taken out [12, 13].

According to the figure 1, the next step that the adversary has to overcome is to detect if the decoded message holds hidden information or not. Indeed, steganography is more and more widespread in addition to cryptology. The plaintext is first ciphered and then hidden into a cover medium with is send through the communication channel. In a second part of this thesis, we present techniques to detect digital images which hold hidden information. First, we present in chapter 5 non compressed and JPEG formats and give a description of the steganographic algorithms for which we propose an attack. Then, in chapter 6, we are interested in classical models of security in steganography. We formalize the real-life adversary and propose two new models of security. This new models of security are connected to the classical ones but also to the performances of practical steganalysis. All the attacks presented in this thesis are evaluated in these new models. Then, we detail an efficient steganalysis against the steganographic scheme *Multi Bit Plane Image Steganography* (MBPIS), designed by B.C. Nguyen *et al.* at IWDW'06 (International Workshop on Digital Watermarking) [147]. The authors claim that their algorithm is robust against RS steganalysis [76, 77] by excluding the areas for which the RS analysis is the most sensitive. A straightforward adaptation of the RS analysis leads us to efficiently detect stego

media generated by MBPIS. As JPEG is perhaps one of the most widespread format to exchange photos, we got interested in steganographic schemes which are dedicated to this format. Finally, we describe in chapter 8 a novel approach which consists in looking for a deviation of the binary entropy into the compressed frequency domain (DFC). This approach leads us to point out a novel class of functions which makes possible the design of detectors the performances of which are quasi-independent of the payload. This technique is illustrated by two steganalysis schemes. The first one is an universal one and is able to detect some unknown steganographic algorithms. The second one is a specific one and is dedicated to the detection of *Outguess*, *F5* and *JPHide* and *JPSeek*, three standard steganographic algorithms.

There are two aims in the chapter 6. The first one is to rigorously model the real-life steganographic adversary which is commonly used. The second one is to point out a lower bound on the insecurity of the attacked schemes directly from practical steganalysis performances. The traditional approach in security proofs is to take the designer's point of view and prove with a strong adversary, the security of a designed scheme, in a given model and using reductions to hard problems. Our approach is a little bit different. We deal with a steganalysis of a given steganography scheme and we want to study the insecurity of the scheme in a model that fits the best the real-life attacker. As the real-life adversary is very weak, it is never taken into account in the design of security models. This implies that effective steganalysis and more precisely those presented in chapters 7 and 8 point out some bounds on the insecurity if the analysed scheme in the proposed models but also in stronger ones. First, using models proposed by C. Cachin [39], S. Katzenbeisser and F. Petitcolas [115] and N. Hopper [99], we formalize the concepts of *discrimination attacks*, classifier and *discriminant steganalysis*; then we summarize the last one as a statistical discrimination problem. The discrimination attacks formalize the *features extraction*. Then, we recall the classical indistinguishability-based security models and weaken them to be closer to the real-life adversary. We propose two new models of security, IND-SSA and IND-USA and link them to the hierarchy of classical ones. The IND-SSA stands for an Specific Steganalysis Adversary model and the IND-USA for the Universal Steganalysis Adversary model. Finally, we lower bound the insecurity using the probabilities of false positive and false negative rates of effective steganalysis and show how this bound also holds for classical models. In conclusion, we discuss about the need to have a common methodology to evaluate the security of a given steganography algorithm or to compare the performances of steganalysis altogether. Finally, we connect our models to classical ones and deduce that a steganographic scheme which is not secure in our models are also not secure in the classical ones. A recent work of A.D. Ker [119] go in the same direction. He proposes a common methodology to reduce the gap between theoretical and practical models. We also detail in section 6.3 the Fisher analysis for classification which is the base of the linear classifiers that we conceive for our attacks. The results of this chapter will be presented at the 10th international workshop on Information Hiding (IH'08) [11].

The last chapters of this thesis are dedicated to practical steganalysis. We first present an attack against *Multi Bit Plane Image Steganography* (MBPIS) proposed by B.C. Nguyen *et al.* at IWDW'06 (International Workshop on Digital Watermarking) [147]. Two effects are expected by using MBPIS; first to avoid the human visual analysis and then the non-random changes of pixels values. One of its most important properties consists in locating the non-noisy areas of bit planes also called *flat areas*. In smooth regions of the cover image,

pixels have similar values. The embedding process may add noise to non-noisy areas and therefore some steganalysis may succeed. Hence the flat areas are isolated and are not modified during the embedding process. Another feature of the designed algorithm is to embed data into the Canonical Gray Coding (CGC). The authors claim that their data hiding method is secure against classical steganalysis like RS analysis [77], χ^2 attacks [204] and Pairs Analysis [59, 134]. But the immunity of MPBIS against some other steganalysis techniques [137, 118] has not been tested so far. A similar approach has been developed by Agaian, Rodriguez and Perez [3]. The main common characteristics to such techniques are to embed information in multi bit planes, to change the initial coding domain and to take advantage of non-informative areas of the image. This algorithm is detailed in section 5.1.2. The discrimination function of the RS analysis is more sensitive to embedding in smooth areas, since changing the value of a pixel in such an area increases the discontinuity of values inside a group of pixels. That is why non-flat areas have very small impact on RS analysis compared to the flat ones. Since MBPIS does not embed data into flat areas, the distortions introduced can not be detected with RS analysis if we consider all the areas. So, we adapt the RS analysis by also excluding flat areas and define a RS window which fits the sliding window used by MBPIS and the discriminating function associated with. In this way, we have take advantage of the counter-measures introduced by the authors to protect MBPIS against RS analysis. Unfortunately, we are not able to estimate the embedding rate as in the classical RS analysis. Actually, the capacity of an uncompressed image according to MBPIS is message dependent as some of the flat areas of a given the bit plane become flat after embedding in higher bit planes. So, it is impossible to compute the coefficients needed for the quadratic interpolation and then, the length estimation. Nevertheless, we also design a classifier based on the classical RS analysis and show that is much less accurate than the classifier based on the adapted RS analysis. One main conclusion we can draw regarding the presented analysis is the following one. We suggest that making steganographic algorithm robust against a steganalysis by avoiding to embed into parts of the image which are significant for the considered analysis is absolutely not secure. In the same way, one strategy which can be performed by the analyser is to focus only on areas effectively used by the steganographic algorithm and then adapt classical analysis without taking into account the avoided parts. The results of that work will be presented at the International Workshop on Digital Watermarking (IWDW'07) [18].

We also get interested in steganography dedicated to JPEG format. We propose in the last chapter a novel approach in steganalysis for such a format. We first notice that it is difficult to maintain in the same time the statistics in the spatial domain, in the frequency domain and in the compressed frequency domain (DFC). Then, we propose to point out statistical deviations in the DFC since the designers only try to keep the statistics in the spatial domain or the frequency domain unchanged. This approach seems counter-intuitive as the DFC is composed of DCT and AC coefficients coded by a Run Length Encoding (RLE) and compressed by Huffman. In a first approximation this binary stream is close to random. Nevertheless, standard JPEG implementation has two factors where a bias may be introduced. First, the JPEG norm proposes to use pre-calculated Huffman tables. These tables are efficient in average and allow to gain the complexity related to their computation and some space in the header. The use of such tables makes Huffman compression under-optimal in that context. Moreover, the coefficient V of the RLE is not compressed and his Hamming weight is not random at all as we explain it in this thesis. So, the binary stream has a binary entropy deviation that we exploit. In the same time, we

show that the *avalanche criteria* [68] of lossless compression step is close to 0.5, *i.e.* only few changes on the input bits of such functions imply a flip of half the bits of the outputs. Using the deviation and this property, we design an universal and a specific steganalysis schemes against *Outguess*, *F5* and *JPHide* and *JPSeek*. For the universal scheme, we cut the binary stream into blocks and we evaluate the distribution of their Hamming weight within the stream. We also introduce as feature the Kulbak-Liebler distance between this distribution and a reference one. For the specific steganalysis, we randomly embed several times the image to be analysed and measure the variation of average number of 0 in the stream. Exploring the compressed frequency domain completes the traditional detection schemes and reveals a new class of good functions for steganalysis. These functions are required to be bijective and have an avalanche criterion close to 0.5. So, whatever the number of changes on the inputs may be, the number of changes on the outputs is always the same. Under this hypothesis, designing steganography classifiers which the accuracies do not depend in practice on the payload may be possible. If the function is well chosen, it could reveal and even magnify statistical deviations which are not visible in its input domain. Looking for such functions appears to be promising. Such functions are not only a theoretical view ; one of them, defined by the RLE and Huffman compression, has been evaluated. The avalanche criterion of the JPEG lossless compression step makes this deviation quasi-independent of the embedding rate and so, makes possible the design of steganographic detectors which the efficiencies do not depend on the payload. We design such classifiers with very high and constant detection rates. The experimental results show that our steganalysis schemes are able to efficiently detect the use of new algorithms which are not used during the training step, even if the embedding rate is very low ($\approx 10^{-6}$). The result of that work has been published at Conference on Cryptography, Coding and Information Security (CCIS'06) [15], at International Workshop on Digital Watermarking (IWDW'06) and in the Journal of Multimedia [17].

Introduction

« Internet est le produit d'une combinaison unique de stratégie militaire, de coopération scientifique et d'innovation contestataire. »

Manuel Castells (La société en réseau)

L'AVÈNEMENT de l'ère du « tout numérique » a radicalement transformé nos modes de communication et a ainsi réduit l'espace-temps de la diffusion d'information à sa plus simple expression. Par le passé, la diffusion d'information était majoritairement unidirectionnelle et sa durée de vie était de l'ordre du mois, voire de l'année ; aujourd'hui, elle se diffuse largement et instantanément. Hier, restreintes à une élite, les Technologies de l'Information de la Communication (TIC), sont maintenant accessibles au plus grand nombre et communiquer avec n'importe qui, n'importe où, et de manière instantanée fait partie intégrante du mode de vie de l'homme moderne, ce, de façon quasi-indépendante du contexte culturel. Un des grands challenges de ces technologies est alors de concevoir des mesures de protection de l'information adaptées à ce nouvel environnement. Des protections tout d'abord pour compenser l'erreur introduite par les canaux de plus en plus variés mais aussi des protections contre l'indiscrétion d'un tiers. Pour ce faire, elles intègrent notamment dans la chaîne de communication des mécanismes assurant la confidentialité des données transmises, tels la cryptographie, mais aussi des mécanismes de correction des erreurs, tels le codage de canal. Un des défis majeurs qu'il reste encore aujourd'hui à relever est la garantie du respect de la vie privée des utilisateurs finaux, face à des personnes indiscrètes dont les motivations peuvent être nombreuses : commerciales, professionnelles, personnelles ou mêmes illégales. Afin de juger la sécurité mise en place dans les systèmes de communication, deux stratégies peuvent être adoptées. La première consiste à prendre la place de l'attaquant et de confronter ces systèmes à l'état de l'art des attaques aussi bien pratiques que théoriques ; la seconde, à prouver formellement la sécurité dans des modèles donnés. La cryptographie offre bien sûr un cadre d'étude idéal d'évaluation de la sécurité, et les attaquants considérés sont en général très forts. On peut alors s'interroger sur la puissance réelle de l'attaquant effectif. En effet, les mécanismes de protection s'intègrent au sein d'un système qu'il faut attaquer étape par étape. Par exemple, l'obtention de messages chiffrés nécessite un certain nombre de traitements préliminaires plus ou moins complexes, qui ne sont pas pris en compte dans les modèles d'attaquants classiques. Dans le cadre d'une interception sans connaissance *a priori*, l'adversaire doit notamment être capable d'enlever toute redondance introduite par le codage correcteur d'erreurs.

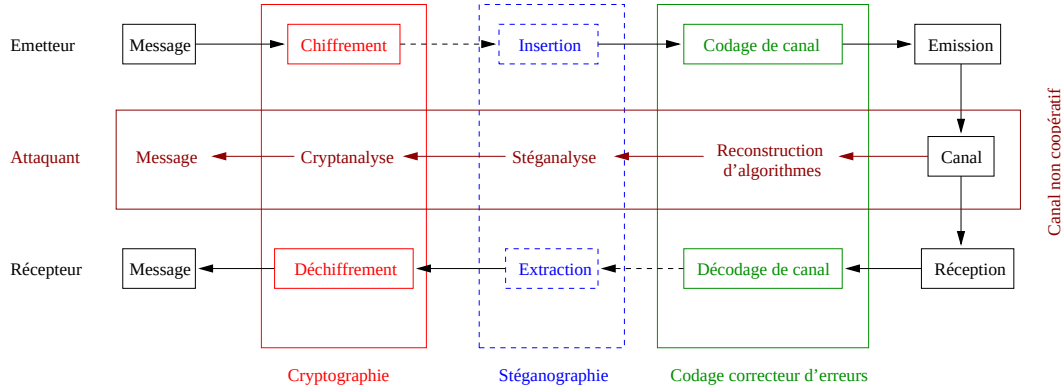


FIG. 2 – Schéma synthétique d'un système de communication

L'objectif de cette thèse est de faire le lien entre l'adversaire théorique traditionnellement utilisé en cryptographie, évoluant dans un monde idéal et l'adversaire réel. Celui-ci essaie d'avoir accès de manière illégitime à l'information échangée entre un émetteur et un récepteur *via* un système de communication représenté schématiquement par la figure 2. Pour ce faire, dans le contexte le plus défavorable, sa connaissance du système qu'il veut écouter est minimale. En faisant l'hypothèse que cet attaquant est passif et possède des capacités d'interception et de traitement du signal qui lui permettent de remonter aux trames binaires codées, cette thèse présente des techniques dont l'objet est de retrouver, à partir de ces trames, le code correcteur d'erreurs, voire le codeur dans certains cas, initialement utilisé. Ces techniques sont une étape incontournable avant l'obtention d'un message cryptanalysable. En effet, aucune cryptanalyse ne semble envisageable en présence des bits de redondances introduits par le codeur correcteur d'erreurs. La recherche des paramètres du codeur s'effectue avec la seule connaissance de la nature du code employé ; ces traitements sont alors qualifiés d'aveugles et le canal est appelé canal non coopératif. Par opposition, le canal est dit coopératif si les paramètres du codeur sont connus ; c'est le cas nominal d'une communication classique dans laquelle, à la fois l'émetteur et le récepteur connaissent les paramètres du codeur.

Parallèlement à l'emploi de la cryptographie pour protéger les communications, on voit apparaître un usage grandissant de la stéganographie. Le message chiffré est alors dissimulé dans un support numérique anodin et c'est ce support qui est envoyé sur le canal de transmission. Alors que l'utilisation de moyens cryptographiques est encore contrôlée, voire interdite dans beaucoup de pays et les réseaux parfois surveillés, la stéganographie s'impose comme un complément pour communiquer plus librement mais surtout de manière furtive. L'attaquant qui veut avoir accès à l'information échangée de manière illégitime doit en plus des traitements énoncés précédemment, détecter quels sont les supports numériques qui contiennent de l'information cachée et en extraire ensuite le message chiffré. Dans cet esprit, le support numérique peut être considéré comme un canal de transmission. Par analogie, la dissimulation correspond à la partie codage et émission de la chaîne de communication classique et l'extraction correspond à la réception et au décodage. Dans ce contexte, l'adversaire ne maîtrise ni les supports numériques qu'il intercepte, ni même les paramètres utilisés pendant la dissimulation ; le support numérique peut alors être vu comme un canal non coopératif. L'attaquant doit alors être capable, en aveugle, d'extraire le message chiffré avant même de pouvoir envisager sa cryptanalyse. Bien que les

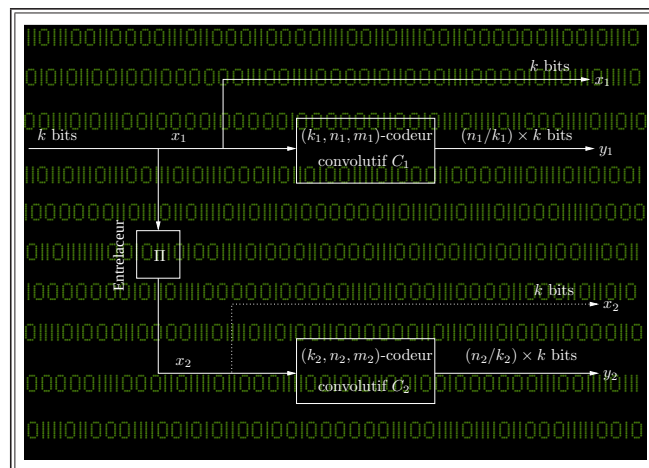
mécanismes de stéganographie et de codage correcteur d'erreurs soient intrinsèquement différents, du point de vue de l'attaquant, retrouver leurs paramètres est une étape incontournable précédant la cryptanalyse. Dans cette thèse l'étude des canaux non coopératifs est abordée sous ces deux angles ; celui de la reconstruction des codes correcteurs d'erreurs et celui de la stéganalyse, ou analyse stéganographique.

Nous présentons dans une première partie, des techniques d'analyse du train binaire, consistant à retrouver en aveugle des paramètres des codes correcteurs utilisés. Les codes étudiés possèdent une structure algébrique forte qui s'exprime simplement dans le formalisme algébrique proposé par G.D. Forney [87] et repris par R.J. McEliece [143]. Tout au long de cette partie, nous nous sommes attachés à mettre en œuvre ce formalisme afin de mettre en évidence une hiérarchie sur les algorithmes de reconstruction proposés. Plus précisément, nous nous intéressons tout d'abord à la reconstruction des codes en blocs linéaires et présentons une analyse très fine de l'algorithme de Sicot et Houcke [175]. Cet algorithme prend en entrée une trame de bits bruitée par le canal et renvoie une estimation du code dual associé au codeur utilisé. Il constitue alors la brique de base de nos techniques de reconstruction. Nous traitons ensuite le cas de codes convolutifs dans la lignée des travaux d'É. Filiol [69, 70, 71] et mettons en évidence de nouvelles équations. Enfin, en concevant un algorithme qui reconstruit un entrelaceur à partir de ses entrées et sorties bruitées, nous généralisons les techniques de reconstruction des chapitres précédents à la reconstruction des turbo-codes. Dans ce cas favorable, nous montrons comment retrouver un décodeur équivalent.

Dans une seconde partie, nous présentons des techniques pour détecter des images fixes contenant de l'information cachée. Nous nous appliquons tout d'abord, à définir précisément l'attaquant réel et proposons deux nouveaux modèles de sécurité qui en découlent. Nous détaillons ensuite une stéganalyse efficace de l'algorithme de stéganographie adapté aux images fixes non compressées, *Multi Bit Plane Image Steganography*, spécifié par B.C. Nguyen *et al.* à IWDW'06 (International Workshop on Digital Watermarking) [147]. Enfin, le JPEG étant l'un des formats d'échange d'images les plus répandus, nous nous sommes intéressés à la stéganographie dédiée à ce format. Nous avons développé une approche nouvelle qui consiste à évaluer une déviation de l'entropie binaire dans le domaine fréquentiel compressé. Cette approche a ensuite été déclinée en deux schémas de stéganalyses. Le premier, qualifié d'universel, permet de détecter l'utilisation d'algorithmes de stéganographie qui sont potentiellement inconnus. Le second, qualifié de spécifique, est dédié à la détection d'un algorithme donné. Nous avons particularisé ce dernier pour détecter *Outguess*, *F5* et *JPHide and JPSeek*, trois algorithmes de référence.

Première partie

Techniques de reconstructions de codes correcteurs d'erreurs



Introduction

« Soignez le commencement, pensez à la fin,
la fin viendra sans fatigue. Si vous oubliez le
but, vous succomberez avant la fin. »

Chou King (Philosophe chinois)

CETTE partie est consacrée à l'étude des techniques de *reconstruction d'algorithmes* appliquées aux codes correcteurs d'erreurs et plus particulièrement aux codes en blocs, codes convolutifs et turbo-codes. Une *technique de reconstruction d'algorithme* $\mathcal{R}$ consiste à retrouver un algorithme inconnu $\mathcal{A}$, ou un algorithme $\mathcal{A}'$ équivalent à $\mathcal{A}$, à partir de la connaissance éventuellement partielle de ses entrées et/ou sorties. Dans le cas d'algorithmes non randomisés, $\mathcal{A}'$ est *équivalent* à $\mathcal{A}$ si et seulement si $\mathcal{A}$ et $\mathcal{A}'$ renvoient des sorties identiques pour des entrées identiques. Cela revient à considérer l'algorithme $\mathcal{A}$ comme une boîte noire et à observer certaines de ses entrées et/ou sorties afin d'obtenir de l'information sur $\mathcal{A}$ ou sur n'importe quel algorithme équivalent.

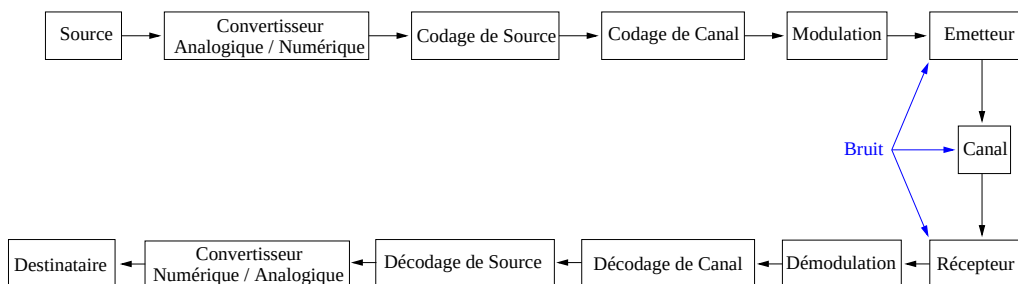


FIG. 3 – Chaîne de communication simplifiée

Dans le cadre de cette étude, nous nous intéressons à une chaîne de communication classique comme décrite par la figure 3, et plus particulièrement à la partie codage de canal. $\mathcal{A}$ est donc un algorithme de codage correcteur d'erreurs qui prend en entrée des trains binaires générés par une source aléatoire sans mémoire selon un processus de Markov d'ordre 0 et renvoie un ensemble $\mathcal{T}$ de trains de bits dont certains vérifient une équation donnée. En général, le message est compressé et éventuellement chiffré avant le codage de canal, ce qui justifie pleinement la modélisation du message par un processus de Markov d'ordre 0. Chaque élément t_i de $\mathcal{T}$ est modulé, émis sur le canal et démodulé pour donner $\tilde{t}_i$ à l'entrée du décodeur de canal. $\tilde{t}_i$ correspond au train binaire auquel l'émetteur, le canal

et le récepteur ont ajouté du bruit, *i.e.* $\tilde{t}_i = t_i + e_i$ où e_i est un vecteur d'erreur ajouté par la chaîne de communication. Seuls, dans cette étude, sont étudiés les canaux additifs, c'est-à-dire pour lesquels le bruit est ajouté ; les canaux à effacement, pour lesquels certains bits disparaissent lors de la traversée du canal, sont hors contexte. Nous nous plaçons du point de vue d'un observateur qui écoute le canal et qui est capable de simuler la réception, la démodulation et éventuellement tous les traitements en amont du décodeur de canal. Cet observateur a donc accès à $\tilde{\mathcal{T}} = \{\tilde{t}_i\}$ et son objectif est de retrouver $\mathcal{A}$ (ou un algorithme équivalent), qu'il ne connaît pas, afin de construire un décodeur de canal équivalent à celui de la chaîne de communication. Malheureusement, dans certains cas, des limitations théoriques ne permettent que de retrouver le code et ses paramètres et redent la détermination du codeur équivalente au problème de décodage d'un code aléatoire, dont on sait que c'est un problème NP-complet [29]. Il nous faut maintenant définir précisément les deux contextes distincts dans lesquels l'observateur peut s'inscrire.

Le contexte *coopératif*, est le contexte le plus favorable. L'observateur connaît toute la chaîne de communication exceptés la source, le codeur et le décodeur de canal. Suivant les schémas étudiés, l'observateur peut aussi avoir accès à quelques paramètres du codeur et éventuellement à certaines de ses entrées. Les applications des techniques de reconstruction dans ce contexte sont nombreuses et sont utilisées le plus souvent dans le cadre de canaux qui varient beaucoup au cours du temps et où la bande passante est faible. L'émetteur et le récepteur doivent alors faire évoluer dynamiquement les paramètres du codeur de canal afin qu'il soit à chaque instant le plus adapté au canal de transmission. L'émetteur change les paramètres du codeur et laisse le soin au récepteur de les « deviner » grâce à une technique de reconstruction [12, 13].

Le contexte *non-coopératif* est beaucoup plus contraignant. L'observateur ne connaît rien de la chaîne de transmission ; il doit d'abord intercepter la transmission sur le canal et reconstruire tous les algorithmes de traitement en aval du codeur de canal. Dans ce contexte, l'observateur modélise généralement un attaquant qui essaie d'avoir accès à de l'information échangée sur un canal de communication pour lequel il n'est pas un utilisateur légitime.

Dans le cadre de cette étude nous prenons la place de l'observateur qui essaie de « remonter » la chaîne de communication en aveugle, *i.e.* dans un canal non coopératif. Nous supposons que notre observateur dispose des matériels d'interception et de traitement de signal qui lui permettent de démoduler le signal intercepté. Il a alors accès au train de bits éventuellement bruité. Sous ces hypothèses, les parties *modulation*, *émetteur*, *canal*, *récepteur* et *démodulation* de la figure 3 page précédente peuvent être considérées, de son point de vue, comme un *canal binaire*. Dans le cas où l'observateur essaie de retrouver les paramètres des codeurs implantés dans un équipement qu'il possède, la canal sera considéré sans erreur et l'équipement stimulé en boîte noire. Dans le cas d'une interception, l'observation se fait sur un canal bruité. Le bruit introduit dépend du canal et de la modulation utilisée ; il convient donc, avant toute étude de modéliser ce canal et surtout le bruit introduit dans les séquences binaires observées. Dans les deux cas, la chaîne de communication, du point de vue de notre observateur et sous les hypothèses précédentes, peut se résumer à la figure 4 page suivante. Différents types de canaux doivent être envisagés pour modéliser l'ensemble des canaux physiques de transmission. Pour évaluer les algorithmes que nous avons développés, nous nous sommes intéressés aux canaux les

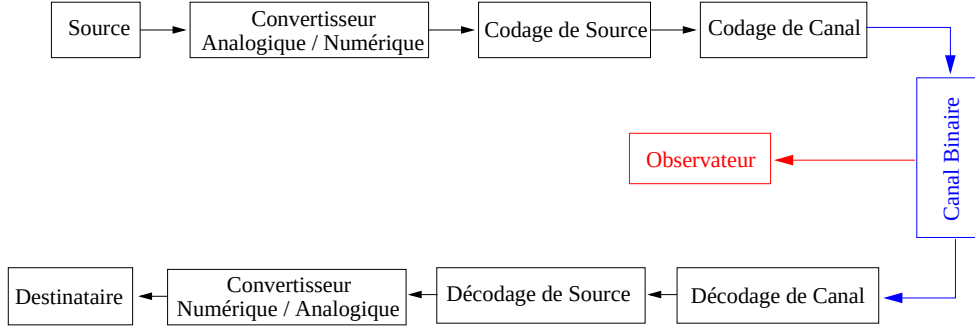


FIG. 4 – Chaîne de communication du point de vue de l'observateur

plus classiques, c'est-à-dire au *canal binaire symétrique* (CBS), au *canal additif gaussien*, au *canal à évanouissement* et enfin au *canal de type burst* afin de couvrir le plus grand éventail de canaux réels. Le bruit ajouté par le canal de transmission est représenté dans le cas d'un canal binaire par un vecteur binaire d'erreur $e = (e_i)$. Si nous notons (y_i) les bits en sortie du codeur de canal (cf. Fig. 4), $(\tilde{y}_i)$ les bits (y_i) en sortie du canal binaire et interceptés par l'observateur, nous avons alors la relation

$$\tilde{y}_i = y_i \oplus e_i \quad \forall i,$$

où $\oplus$ est le *ou-exclusif*. Le cas le plus simple est celui du canal binaire symétrique. Les e_i sont des variables aléatoires indépendantes qui suivent chacune, une distribution uniforme de paramètre p , *i.e.*

$$\Pr(e_i = 1) = p \quad \text{et} \quad \Pr(e_i = 0) = q = 1 - p.$$

Le *taux d'erreur binaire* (TEB) ε est donc constant et égal à p . Le canal additif gaussien, quant à lui, ajoute au signal émis un signal d'amplitude B , variable aléatoire suivant une distribution gaussienne de paramètres 0 et $\frac{N_0}{2}$ où N_0 est la *densité spectrale monolatérale de puissance du bruit*, donnée physique du canal. Grâce à des calculs classiques en traitement du signal [8], le canal binaire de la figure 4 se ramène à un canal binaire symétrique dont le taux d'erreur binaire ε dépend du canal et du type de modulation utilisée. De même, le canal à évanouissement ajoute au signal émis un signal d'amplitude B , variable aléatoire suivant une distribution de Rayleigh-Rice de paramètres 0 et $\frac{N_0}{2}$ et se ramène à un canal binaire symétrique dont le taux d'erreur binaire ε dépend du canal et du type de modulation utilisée. Les canaux étudiés jusqu'ici présentaient un TEB constant au cours du temps. Pour certains canaux (canal satellite par exemple), cette hypothèse ne tient pas et l'on est obligé de considérer un modèle un peu plus fin pour représenter des erreurs qui arrivent par paquets, ou *salves*. La variation au cours du temps de ε peut être approximée par un processus de Markov d'ordre 1. Ces canaux appelés « *de type burst* », peuvent être représentés par un automate fini à deux états comme illustré sur la figure 5 page suivante. Ce modèle est appelé *modèle de Gilbert-Elliot* [91].

L'automate de Gilbert-Elliot (G-E) est un automate fini à deux états, l'état « Good » correspond à des plages sans erreur et l'état « Bad » correspond à des plages avec erreurs. Les probabilités de transitions $\Pr(B|G)$ et $\Pr(G|B)$ sont respectivement b et g . Dans l'état B la probabilité d'erreur est P_b et dans l'état G , P_g . Le TEB moyen est donné par la formule

$$\varepsilon = \frac{1}{g + b} [bP_b + gP_g].$$

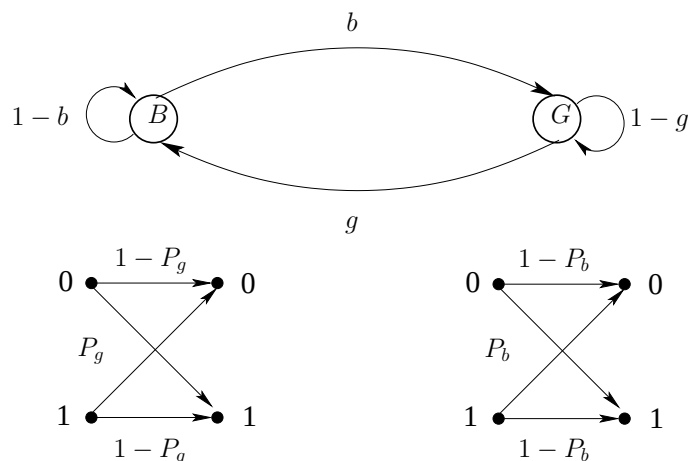


FIG. 5 – Diagramme d'état-transition du modèle de Gilbert-Elliot

De plus, le processus peut-être rendu stationnaire en prenant comme probabilités initiales

$$\mathcal{Pr}(G) = \frac{g}{b+g} \quad , \quad \mathcal{Pr}(B) = \frac{b}{b+g}.$$

Il est aussi utile d'introduire le coefficient μ défini par $\mu = 1 - g - b$. μ prend des valeurs comprises entre -1 et 1 et sert à mesurer l'intensité de la mémoire du canal. Si μ est proche de 1, le canal tend à générer de longues salves d'erreurs. La valeur -1, un peu moins réaliste, correspond à une alternance rapide entre les états. En général, plus g (resp. b) est faible, plus les salves d'erreurs (resp. plages sans erreur) sont longues. Traditionnellement, P_g est choisi beaucoup plus petit que P_b . D'autre part, en fixant les bons paramètres de G-E, on obtient une très bonne approximation du canal de Rayleigh. Les paramètres de G-E à choisir pour simuler plusieurs types de canaux de Rayleigh sont indiqués dans [182]. Le tableau 1 donne des valeurs possibles des paramètres à choisir pour un TEB fixé. De plus, en

TEB moyen	paramètres (g, b, P_g, P_b)
0.1	(0.017,0.003,0.03,0.5)
0.2	(0.014,0.06,0.071,0.5)
0.3	(0.014,0.06,0.214,0.5)
0.4	(0.014,0.06,0.357,0.5)
0.42	(0.014,0.06,0.386,0.5)
0.44	(0.014,0.06,0.414,0.5)
0.46	(0.014,0.06,0.443,0.5)
0.49	(0.014,0.06,0.486,0.5)

TAB. 1 – Paramètres G-E pour un TEB donné

choisissant $g = b = P_g = 0$, $P_b = \varepsilon$, on retrouve la définition du canal binaire symétrique de TEB ε . Le modèle de Gilbert-Elliot possède un pouvoir de description puissant et c'est naturellement celui que nous avons adopté pour approximer les canaux binaires.

De par leurs applications très spécifiques, les techniques de reconstruction appliquées à une chaîne de communication dans un contexte non-coopératif, ne font pas l'objet d'une littérature très fournie. Néanmoins, G. Planquette est l'un des précurseurs de l'analyse du train binaire [154]. Celui-ci montre comment détecter l'utilisation de codes en blocs, de codes convolutifs et des brasseurs auto-synchronisants en utilisant deux types d'approches distinctes. La première consiste à estimer des corrélations sur des signaux de sortie, en utilisant des tests statistiques et d'hypothèses. La seconde approche est plutôt de type algébrique et consiste à détecter une variation de rang dans une matrice composée des bits interceptés. Celle-ci est complétée par une recherche de relations de parité de poids faible à l'aide des algorithmes de Leon [129] et Canteaut-Chabaud [42]. G. Planquette a alors posé les bases de ces deux approches dont sont issus tous les résultats du domaine de l'analyse du train binaire. Dans le même esprit, A. Valembois [195, 196] a formalisé les problèmes de détection et reconnaissance des codes linéaires en suivant les deux approches précédemment définies. Ses résultats ont été récemment améliorés par M. Cluzeau [50, 51] en utilisant un algorithme de décodage itératif pour corriger des erreurs avant même la fin du processus de reconstruction. Parallèlement, G. Burel et R. Gautier [37], pour un canal sans bruit, ainsi que G. Sicot et S. Houcke [173, 174, 175], dans le cas général, ont obtenus des résultats similaires en tirant au maximum parti de la résistance au bruit de l'algorithme de Gauss pour reconnaître des codes en blocs. D'autre part, B. Rice [162] et É. Filiol [69, 70, 71] se sont intéressés plus particulièrement à la reconstruction des codes convolutifs.

Dans cette étude, nous nous sommes tout d'abord attachés à améliorer les résultats d'É. Filiol puis à les généraliser aux turbo-codes. Dans un deuxième temps, nous avons regroupé sous un formalisme général les algorithmes de reconstruction des codes convolutifs et des codes en blocs. Ce formalisme s'appuie sur l'approche algébrique qui permet notamment de voir les codes en blocs comme des cas particuliers des codes convolutifs [143]. Nous rappelons tout d'abord les fondements de l'approche algébrique des codes convolutifs au chapitre 1. Nous formalisons ensuite le problème de reconstruction des codes en blocs linéaires et analysons sous cet angle l'algorithme proposé par G. Sicot et S. Houcke dans le chapitre 2. Le formalisme ainsi introduit nous a permis d'aller plus loin dans l'automatisation du processus de reconstruction proposé par É. Filiol et ainsi d'en améliorer significativement la complexité. De plus, l'analyse de l'algorithme de G. Sicot et S. Houcke a permis d'expliquer pourquoi les résultats expérimentaux observés par É. Filiol étaient meilleurs que ceux attendus en théorie. Cette amélioration est décrite dans le chapitre 3. Enfin, au chapitre 4 nous généralisons ces algorithmes en une technique efficace pour reconstruire les turbo-codes.

Représentation algébrique des codes convolutifs et linéaires

« N'admettez rien a priori si vous pouvez le vérifier. »

Rudyard Kipling (Souvenir)

Sommaire

1.1 L'approche algébrique	26
1.1.1 Codeurs et codes convolutifs	26
1.1.2 Fonctions génératrices	28
1.1.3 Définition d'un code convolutif	29
1.1.4 Codeurs convolutifs rékursifs	31
1.2 Propriétés de codeurs convolutifs	32
1.2.1 La notion de degré	32
1.2.2 Matrices génératrices	33
1.2.3 Matrices catastrophiques	36
1.2.4 Les codes optimaux	37
1.3 Les codes convolutifs poinçonnés	39
1.3.1 Définition des codes poinçonnés	39
1.3.2 Construction par regroupement	42
1.3.3 Approche algébrique du regroupement	43
1.3.4 Poinçonnage	45
1.3.5 Les « bons » codes poinçonnés	45

LA théorie des codes convolutifs a été introduite par P. Elias [64] en 1955. Sans algorithme effectif de décodage, celle-ci est restée longtemps sans application pratique. En 1959, les codes convolutifs sont remis au bout du jour par D.W. Hagelbarger [96] sous le nom de *codes récurrents*. L'intérêt naissant de la communauté pour ces codes a permis, dès 1961, aux premiers algorithmes de décodage de voir le jour. Les performances de ce nouveau type de code correcteur d'erreurs dépassent de loin celles des codes classiques, notamment celles de codes en blocs. Les algorithmes de décodage séquentiel [207, 66, 108],

de décodage à seuil [139] mais surtout l'algorithme de Viterbi [197, 151, 89] ont rendu les codes convolutifs populaires et répandus. Ceux-ci sont particulièrement adaptés aux communications spatiales ; ils occupent alors une place privilégiée dans la grande histoire de l'aventure spatiale. En effet, la NASA adopte comme standard le code découvert par J.P. Odenwalder [148] en 1970. Ce même code est utilisé par la navette *Voyager* pour coder les photos de Jupiter, Saturne, Uranus et Neptune, au début des années 80.

Ce chapitre, essentiellement issu de l'article de R.J. McEliece [143], présente en détails la théorie de code convolutifs en utilisant l'approche algébrique, nécessaire à la compréhension des algorithmes de reconstruction.

1.1 L'approche algébrique

À l'opposé des codes en blocs dont la théorie est parfaitement comprise, les codes convolutifs sont plus difficilement appréhendables car la théorie sous-jacente est beaucoup plus complexe. La théorie de G.D. Forney [87, 88, 90] fournit néanmoins des éléments de comparaison entre les codes convolutifs et les codes en blocs. L'approche adoptée par G. D. Forney est algébrique et permet d'utiliser un formalisme commun entre ces deux familles de codes correcteurs et de considérer ainsi les codes en blocs linéaires comme des codes convolutifs particuliers. Si n est la dimension de l'espace des mots du code, k la dimension de l'espace des mots d'information et m le degré du code alors l'étude des (n, k) -codes en blocs linéaires aura tendance à considérer n, k grands et $m = 0$ tandis que l'étude des (n, k, m) -codes convolutifs considérera n, k fixés, petits et m grand.

1.1.1 Codeurs et codes convolutifs

Notons F le corps des symboles, utilisé pour coder l'information ; nous prendrons $F = GF(2)$. Nous appellerons *mot d'information*, un vecteur de F^k et *mot de code*, un vecteur de F^n . Un (n, k, m) -codeur convolutif peut alors être vu comme une application ϕ de $(F^k)^*$ dans $(F^n)^*$, avec $n > k$, telle que ϕ associe à une séquence de mots d'information, $(x(i))_{i \geq 0}$ une séquence de mots de code $(y(i))_{i \geq 0}$, i.e.

$$\phi : x(1), x(2), \dots, \longrightarrow y(1), y(2), \dots,$$

telle que le mot de code $y(i)$ dépende de $x(i-1), \dots, x(i-M)$, où M est appelé *mémoire du codeur*. Sous cet angle, un code en bloc peut être considéré comme un code convolutif de mémoire nulle. Le codeur convolutionnel possède alors une mémoire interne, qui peut être représentée par un *vecteur d'état*, $s(i)$, de m coordonnées à valeurs dans F^k . Le $i^{\text{ème}}$ mot de code $y(i)$ est une fonction linéaire de $x(i)$ et de $s(i)$. Le codeur convolutif est alors entièrement défini par la donnée des matrices¹ à coefficients dans $GF(2)$,

$$\begin{aligned} \mathcal{A} &: m \times m, \\ \mathcal{B} &: k \times m, \\ \mathcal{C} &: m \times n, \\ \mathcal{D} &: k \times n, \end{aligned}$$

vérifiant la relation $s(0) = 0$ et $\forall i \geq 0$,

$$\begin{cases} s(i+1) &= s(i)\mathcal{A} + x(i)\mathcal{B}, \\ y(i) &= s(i)\mathcal{C} + x(i)\mathcal{D}. \end{cases} \quad (1.1)$$

¹Nous utiliserons tout au long de ce chapitre la convention d'écriture matricielle anglo-saxonne.

L'entier m est appelé *degré du codeur convolutif*. C'est aussi le nombre de registres utiles pour garder en mémoire les mots d'information nécessaires au calcul du mot de code courant. Nous retrouvons ainsi la définition d'un code en bloc lorsque m est égal à 0 ; *i.e.* un codeur convolutif de degré 0, ou *sans mémoire*², est alors un codeur en blocs linéaire défini par l'équation

$$y(i) = x(i)\mathcal{D}.$$

Définition 1.1 On appelle *code convolutif associé au codeur convolutif* $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$, l'ensemble de toutes les séquences possibles produites par le codeur.

Exemple suivi 1.1

Nous prendrons pour la suite, en exemple guide, le $(2, 3, 5)$ -codeur convolutif, $\mathcal{C}_1$,

$$\begin{cases} y_1(i) = x_1(i) + x_1(i-1) + x_2(i-2) + x_2(i-3), \\ y_2(i) = x_1(i-1) + x_1(i-2) + x_2(i), \\ y_3(i) = x_1(i) + x_1(i-1) + x_1(i-2) + x_2(i), \\ \quad + x_2(i-1) + x_2(i-2) + x_2(i-3). \end{cases} \quad (1.2)$$

Le codeur $\mathcal{C}_1$ transforme la séquence de mots d'information $((x_1(i), x_2(i)))_{i \geq 0}$ en la séquence de mots de code $((y_1(i), y_2(i), y_3(i)))_{i \geq 0}$, et son état interne est une séquence de vecteurs d'état

$$s(i) = (s_1(i), s_2(i), s_3(i), s_4(i), s_5(i))_{i \geq 0}$$

permettant de garder en mémoire les valeurs respectives de

$$x_1(i-1), x_1(i-2), x_2(i-1), x_2(i-2), x_2(i-3).$$

La définition de l'état interne implique

$$\begin{cases} s_1(i+1) = x_1(i), \\ s_2(i+1) = x_1(i-1) = s_1(i), \\ s_3(i+1) = x_2(i), \\ s_4(i+1) = x_2(i-1) = s_3(i), \\ s_5(i+1) = x_2(i-2) = s_4(i). \end{cases} \quad (1.3)$$

Les équations (1.2) et (1.3) peuvent être réécrites sous la forme d'un système matriciel

$$\begin{cases} s(i+1) = s(i) \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + x(i) \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \\ y(i+1) = s(i) \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} + x(i) \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}. \end{cases} \quad (1.4)$$

²Degré et mémoire sont des notions distinctes (cf. §1.2.1).

1.1.2 Fonctions génératrices

Ce paragraphe introduit la notion de transformation en z à partir de *fonctions génératrices*. Cette transformation permet d'associer à une suite d'éléments, une série formelle ; elle est le cœur de *l'approche algébrique* des codes convolutifs.

Définition 1.2 On appelle *fonction génératrice*, ou *transformation en z* , une application qui, à une suite $(a(i))_{i \geq 0}$ d'éléments de F , associe la série formelle

$$A(Z) = \sum_{i \geq 0} a(i)Z^i.$$

L'ensemble des séries formelles à coefficients sur F constitue un anneau commutatif, noté $F[[Z]]$.

Remarque 1.1 :

L'indéterminée en Z permet de représenter un décalage d'indice entre les termes de la suite $(a(i))_{i \geq 0}$. Dans le cadre de notre étude, les éléments de la suite sont émis « au cours du temps » ; l'indéterminée sera alors notée par convention D , pour « *delay* ». L'anneau $F[[D]]$ peut être prolongé en un corps $F((D))$, corps des séries formelles de Laurent, de la forme

$$A(D) = \sum_{i \geq -m} a(i)D^i,$$

où m est un entier positif.

Définition 1.3 On appelle *valuation d'une série de Laurent* $A(D)$, l'entier $v(A(D))$, défini par

$$v(A(D)) = \min\{i | a(i) \neq 0\}.$$

Définition 1.4 On appelle *poids d'une série de Laurent* $A(D)$ le nombre de ses coefficients non nuls, noté $w(A(D))$.

Nous faisons référence, par la suite, à cinq catégories de séries particulières.

- Les séries de Laurent de la forme $A(D) = a(0) + \dots + a(L)D^L$ sont appelées *polynômes*. L'ensemble des polynômes est noté $F[D]$.
- Les séries de Laurent de la forme de la définition 1.2 sont dites *causales*. L'ensemble des séries causales est noté $F[[D]]$.
- Chaque fonction rationnelle $P(D)/Q(D)$, avec $P(D), Q(D)$ polynômes et $Q(D) \neq 0$, possède un unique développement en série de Laurent qui est appelé *série de Laurent rationnelle*. L'ensemble des séries de Laurent rationnelles est noté $F(D)$.
- Une série de Laurent causale et rationnelle est dite *réalisable*.
- Les séries de Laurent de poids fini. Elles sont de la forme $P(D)/D^L$, avec $P(D)$ polynôme et L entier positif.

Remarque 1.2 :

Une série de Laurent causale et de poids fini est un polynôme et toute série de poids fini est rationnelle. $F(D)$ est appelé le *sous-corps rationnel* de $F((D))$. On peut montrer que c'est le plus petit corps contenant F et D . Une série de Laurent est réalisable si et seulement si elle est de la forme $P(D)/Q(D)$, avec $P(D), Q(D)$ polynômes et $Q(0) \neq 0$.

La transformation en z se généralise, de façon naturelle, aux matrices à coefficients dans F et aux F -espaces vectoriels de dimension finie en appliquant la transformation précédemment définie aux coefficients des matrices et, respectivement, aux composantes des vecteurs. Ainsi, la transformée en z de la séquence $x = (x(i))_{i \geq 0}$, où $x(i) \in F^k$, sera

$$X(D) = \sum_{i \geq 0} x(i)D^i = \left(\sum_{i \geq 0} x_1(i)D^i, \dots, \sum_{i \geq 0} x_k(i)D^i \right).$$

En considérant que $\forall i < 0, x(i), y(i), s(i)$ valent 0, nous pouvons multiplier chaque membre de la relation (1.1) page 26, par D^i et sommer sur i . Nous obtenons alors la transformée en z du système (1.1),

$$\begin{cases} S(D)D^{-1} &= S(D)\mathcal{A} + X(D)\mathcal{B}, \\ Y(D) &= S(D)\mathcal{C} + X(D)\mathcal{D}. \end{cases} \quad (1.5)$$

Nous pouvons ainsi obtenir une expression simple de $S(D)$ et $Y(D)$ en fonction de $X(D)$ en résolvant le système (1.5) :

$$\begin{cases} S(D) &= X(D)E(D), \\ Y(D) &= X(D)G(D), \end{cases} \quad (1.6)$$

où les matrices $E(D)$, de taille $k \times m$ et $G(D)$, de taille $k \times n$, sont données par la formule

$$E(D) = \mathcal{B}(D^{-1}I_m - \mathcal{A})^{-1}, \quad (1.7)$$

$$G(D) = \mathcal{D} + E(D)\mathcal{C}. \quad (1.8)$$

où I_m est la matrice identité $m \times m$.

Définition 1.5 Dans les conditions précédentes, la matrice G est appelée *matrice génératrice du (n, k, m) -codeur convolutif*.

Nous ferons l'hypothèse, sans perte de généralité, que la matrice $G(D)$ est de rang k .

1.1.3 Définition d'un code convolutif

L'ensemble des séquences d'information possibles, $F[[D]]^k$, peut être élargi à $F((D))^k$ si l'on considère que le codeur peut commencer à coder à partir de n'importe quel moment non restreint à $i = 0$. Dans le même esprit que celui de la définition 1.1 page 27, un code convolutif peut être vu comme étant l'ensemble des fonctions génératrices, $Y(D)$, produites par le codeur, lorsque $X(D)$ parcourt $F((D))^k$.

Par définition, les coefficients de G , sont rationnels, donc, d'après la relation 1.8, un (n, k) -code convolutif est un sous-espace de dimension k de $F((D))^n$, inclus dans $F(D)^n$.

Réciproquement, toute matrice $G(D)$ de dimension $k \times n$ à coefficients dans $F(D)$ est $F(D)$ -équivalente à une matrice $G'(D)$ causale. De plus, on peut montrer qu'il existe $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$ (cf. [194]) qui réalise $G'(D)$. On peut donc en conclure que l'ensemble des (n, k) -codes convolutifs et l'ensemble des sous-espaces de dimension k de $F((D))^n$ et inclus dans $F(D)^n$ sont égaux. Ceci permet de définir un code convolutif sous l'angle algébrique par

Définition 1.6 *Un (n, k) -code convolutif est un sous-espace de dimension k de $F((D))^n$, inclus dans $F(D)^n$.*

Remarque 1.3 :

Un code convolutif est entièrement défini par la donnée d'un espace vectoriel ; en revanche plusieurs matrices génératrices permettent de générer le même espace vectoriel et donc le même code. Le choix d'une matrice génératrice détermine un codeur convolutif particulier. En pratique, les mots de code sont de poids finis et les seules matrices qui correspondent aux codeurs réalisables sont celles dont les entrées sont causales et rationnelles. Sans perte de généralité, nous pouvons adopter la définition suivante pour les codes convolutifs.

Définition 1.7 *Un (n, k) -code convolutif est un sous-espace de dimension k de $F(D)^n$.*

Exemple suivi 1.2

Reprenons l'exemple précédent du $(2, 3, 5)$ -codeur convolutif $\mathcal{C}_1$. Ce codeur est défini de façon unique par le quadruplet

$$(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}) = \left(\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \right),$$

mis en évidence par le système (1.4) page 27. Les relations (1.7) page précédente et (1.8) page précédente, permettent d'obtenir les expressions de $E(D)$ et $G(D)$.

$$\begin{aligned} E(D) &= \mathcal{B}(D^{-1}I_m - \mathcal{A})^{-1} = \begin{bmatrix} D & D^2 & 0 & 0 & 0 \\ 0 & 0 & D & D^2 & D^3 \end{bmatrix}, \\ G(D) &= \mathcal{D} + E(D)\mathcal{C} = \begin{bmatrix} 1 + D & D + D^2 & 1 + D + D^2 \\ D^2 + D^3 & 1 & 1 + D + D^2 + D^3 \end{bmatrix}. \end{aligned}$$

La matrice $G(D)$ définit de façon unique le codeur $\mathcal{C}_1$; c'est une des matrices génératrices qui engendrent le code convolutif, que nous noterons $\mathcal{C}$. Le code $\mathcal{C}$ est engendré par toutes les matrices génératrices $F(D)$ -équivalentes à $G(D)$. Par exemple, la matrice

$$G'(D) = G(D)/(1 + D) = \begin{bmatrix} 1 & D & \frac{1+D+D^2}{1+D} \\ D^2 & \frac{1}{1+D} & 1 + D^2 \end{bmatrix},$$

définit un codeur $\mathcal{C}_2$ qui engendre $\mathcal{C}$. Le codeur $\mathcal{C}_2$ est de degré 5 mais de mémoire infinie.

1.1.4 Codeurs convolutifs récurrents

Les codeurs convolutifs *récurrents* sont les codeurs qui possèdent des *registres avec rétroaction*. Ces registres avec rétroaction correspondent à des coefficients rationnels dans la matrice génératrice du codeur. Nous représentons un registre par une case associée à l'opérateur *delay* D . Dans le cas où $F = GF(2)$, le registre avec rétroaction simple de la figure 1.1 permet d'écrire les équations suivantes :

$$\begin{aligned} y(i) &= s(i-1), \\ s(i) &= x(i) + y(i) = x(i) + s(i-1). \end{aligned}$$

Nous pouvons en déduire facilement

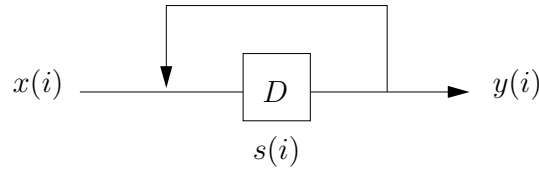


FIG. 1.1 – Registre avec rétroaction

$$y(i) = x(i-1) + x(i-2) + \dots,$$

et en appliquant la transformée en z ,

$$Y(D) = DX(D) + D^2X(D) + D^3X(D) + \dots = \frac{D}{1+D}X(D).$$

En choisissant un polynôme de rétroaction primitif, les codes récurrents ainsi obtenus possèdent de bonnes performances [21]. De plus, le taux d'erreur résiduelle après décodage pour un codeur récurrent est plus petit que celui du codeur convolutif non récurrent correspondant pour un canal faiblement bruité [27, 31]. De par leur nature, les codeurs récurrents ont une mémoire infinie et leur matrice génératrice possède au moins un coefficient non polynomial.

Exemple suivi 1.3

Le codeur associé à la matrice génératrice $G'(D) = G(D)/(1+D) = \begin{bmatrix} 1 & D & \frac{1+D+D^2}{1+D} \\ D^2 & \frac{1}{1+D} & 1+D^2 \end{bmatrix}$ est récurrent et se représente physiquement par le circuit de la figure 1.2 page suivante où $\bullet$ représente le ou exclusif. Pour s'en convaincre, numérotions les fils de 1 à 6 de haut en bas et de a à c de gauche à droite. La correspondance est la suivante :

$$\begin{array}{ll} 1. & X_1(D) \quad 4. & X_2(D) \\ 2. & DX_1(D) \quad 5. & D^2X_2(D) \\ 3. & \frac{D}{1+D}X_1(D) \quad 6. & \frac{D}{1+D}X_2(D) \end{array}$$

Pour chaque fil de a à c , on ne compte que les fils horizontaux marqués par $\bullet$.

Nous obtenons

- a. $X_1(D) + D^2X_2(D)$,
- b. $DX_1(D) + X_2(D) + \frac{D}{1+D}X_2(D)$
 $= DX_1(D) + \frac{1}{1+D}X_2(D)$,
- c. $X_1(D) + DX_1(D) + \frac{D}{1+D}X_1(D) + X_2(D) + D^2X_2(D)$
 $= \frac{1+D+D^2}{1+D}X_1(D) + (1+D^2)X_2(D)$.

Nous retrouvons bien notre matrice $G'(D)$.

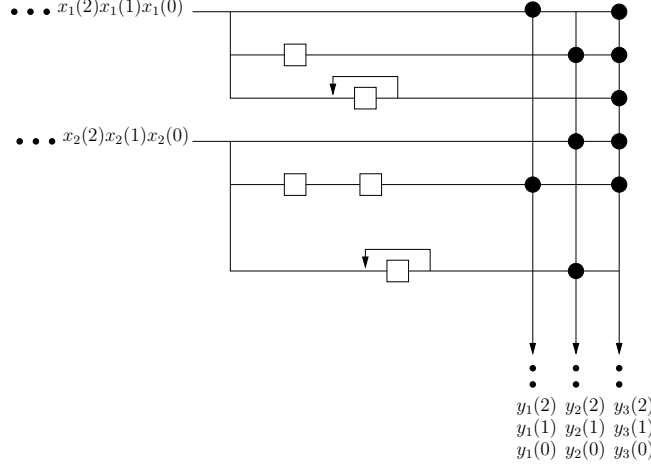


FIG. 1.2 – Représentation physique du codeur récursif de matrice génératrice G'

1.2 Propriétés de codeurs convolutifs

L'approche algébrique fournit une théorie puissante, basée sur l'algèbre des matrices à coefficients dans $F(D)$. Elle va nous permettre, à partir d'observations simples sur des matrices génératrices de déduire des propriétés sur les codeurs convolutifs mais aussi sur les codes qu'ils engendrent.

1.2.1 La notion de degré

Nous avons défini précédemment la notion de *degré* pour un codeur convolutif comme étant le nombre de coordonnées à valeur dans F^k du vecteur d'état interne, s . Dans le cadre de l'étude algébrique, nous nous intéresserons aux codeurs possédant une *Matrice Génératrice Polynomiale* (MGP), *i.e.* dont les coefficients sont des polynômes. Chaque code convolutif possède une MGP ; en effet, n'importe quelle matrice génératrice du code dont on multiplie chaque ligne par le plus petit commun multiple (PPCM) des dénominateurs de ses coefficients, est une MGP.

Définition 1.8 Soit $X(D)$, un vecteur de $F[D]^L$, L entier positif. On appelle *degré* de $X(D)$, l'entier $\deg X(D)$ défini par

$$\deg X(D) = \max_{i=1 \dots L} \{\deg X_i(D)\}.$$

De même, nous noterons $G_i(D)$ la $i^{\text{ème}}$ ligne de $G(D)$. La définition précédente se généralise aux MGP de k lignes de façon naturelle en prenant

$$\deg G(D) = \max_{i=1\dots k} \{\deg G_i(D)\}.$$

Le degré d'une MGP est aussi appelé *mémoire du codeur*.

Définition 1.9 Soit une matrice de dimension $k \times n$. On appelle mineurs d'ordre i , avec $i \leq \min(k, n)$, les déterminants de ses sous-matrices carrées de taille $i \times i$.

Définition 1.10 Soit $G(D)$ une MGP et $\Delta = (\Delta_i)_{i=1\dots n-k}$ où les Δ_i sont les $\binom{n}{k}$ mineurs d'ordre k de $G(D)$. On appelle degré interne, l'entier positif noté $\text{intdeg } G(D)$, défini par

$$\text{intdeg } G(D) = \max_{i=1\dots n-k} \{\deg \Delta_i\}.$$

Définition 1.11 Soit $G(D)$ une MGP, on appelle degré externe, l'entier positif noté $\text{extdeg } G(D)$, défini par

$$\text{extdeg } G(D) = \sum_{i=1}^k \deg G_i(D).$$

Des considérations sur les degrés interne et externe d'une MGP mettent en évidence des propriétés intéressantes pour les codeurs. Nous pouvons finalement définir le *degré d'un code convolutif*.

Définition 1.12 On appelle degré du code convolutif $\mathcal{C}$, noté $\deg \mathcal{C}$, le degré externe minimal obtenu sur l'ensemble des MGP qui l'engendrent. Le degré de $\mathcal{C}$ est aussi appelé longueur de contrainte de $\mathcal{C}$.

1.2.2 Matrices génératrices

Nous mettons en évidence dans ce paragraphe différentes catégories de MGP qui possèdent des propriétés très fortes. Ces propriétés expliquent la préférence accordée pour l'implémentation de certains codeurs.

Définition 1.13 Une MGP, $G(D)$, est dite systématique si et seulement si elle est de la forme

$$G(D) = \begin{bmatrix} I_k | G'(D) \end{bmatrix},$$

où I_k est la matrice identité d'ordre k et $G'(D)$ une MGP de dimension $k \times (n - k)$. Le codeur associé est appelé codeur systématique.

Définition 1.14 Une MGP, $G(D)$, de taille $k \times n$ est dite basique si et seulement si pour toute MGP de la forme $T(D)G(D)$ où $T(D)$ est non singulière de dimension $k \times k$ et à coefficients dans $F(D)$, la relation suivante est vérifiée.

$$\text{intdeg } G(D) \leq \text{intdeg } T(D)G(D).$$

Remarque 1.4 :

Nous rapellons qu'une matrice *unimodulaire* est une matrice carrée à coefficients dans $F(D)$ et de déterminant appartenant à F^* .

Définition 1.15 Une MGP, $G(D)$, de taille $k \times n$ est dite *réduite* si et seulement si pour toute MGP de la forme $T(D)G(D)$ où $T(D)$ est unimodulaire, la relation suivante est vérifiée.

$$\text{extdeg } G(D) \leq \text{extdeg } T(D)G(D).$$

Définition 1.16 Dans l'ensemble des MGP qui engendrent un même code $\mathcal{C}$, celles qui sont de degré externe minimal sont appelées *matrices génératrices canoniques* de $\mathcal{C}$. Une matrice canonique $G(D)$ vérifie donc

$$\text{extdeg } G(D) = \deg \mathcal{C},$$

d'après la définition 1.12.

Remarque 1.5 :

Une matrice canonique d'un code convolutif n'est pas unique. Pour s'en convaincre, considérons une matrice canonique $G_0(D)$ d'un code convolutif $\mathcal{C}$. Alors toute matrice $G(D)$, obtenue par permutation des lignes de $G_0(D)$, engendre $\mathcal{C}$ et $\text{extdeg } G(D) = \text{extdeg } G_0(D)$ est donc minimal. Pour un (n, k) -code convolutif $\mathcal{C}$, il existe au moins $(k!)$ matrices génératrices canoniques engendrant $\mathcal{C}$.

Définition 1.17 Soit $G(D)$, une MGP. On appelle *indices de Forney*, l'ensemble, noté $\{e_1, \dots, e_k\}$ des degrés des lignes de $G(D)$.

$$\{e_1, \dots, e_k\} = \{\deg G_1(D), \dots, \deg G_k(D)\}.$$

On peut montrer assez facilement que les indices de Forney sont invariants pour l'ensemble des MGP canoniques engendrant le même code convolutif [143]. Il en est de même pour $\sum_{i=1}^k e_i = m$, le degré externe de $G(D)$ et $\max_i \{e_i\} = \deg G(D)$, la mémoire du codeur. De plus, nous avons la propriété suivante.

Proposition 1.1 Soit $G(D)$ une MGP canonique de degré m engendrant un (n, k) -code convolutif $\mathcal{C}$ alors

$$\deg \mathcal{C} = m.$$

Remarque 1.6 :

À partir de maintenant, nous notons m le degré du code et nous parlons de (n, k, m) -code convolutif. La notation classique dans la littérature engendre souvent une confusion avec m le degré d'un codeur convolutif qui désigne en fait le nombre de registres pour garder en mémoire l'information nécessaire au calcul des mots de code.

Exemple suivi 1.4

Le codeur $\mathcal{C}_1$ de notre exemple guide est représenté par une MGP, $G(D) = \begin{bmatrix} 1+D & D+D^2 & 1+D+D^2 \\ D^2+D^3 & 1 & 1+D+D^2+D^3 \end{bmatrix}$.

Nous en déduisons aisément

$$\begin{aligned} \deg G(D) &= 3, \\ \text{intdeg } G(D) &= \max\{\deg(1+D^2+D^5), \deg(1+D^2+D^4+D^5), \deg(1+D+D^3+D^5)\}, \\ &= 5, \\ \text{extdeg } G(D) &= 2+3=5, \\ \text{Indices de Forney} &= (2,3), \\ \deg \mathcal{C} &= 2+3=5. \end{aligned}$$

Nous avons $\deg \mathcal{C} = m$, mais à ce stade, la proposition 1.1 page ci-contre ne nous permet pas de conclure que $G(D)$ est canonique. Cette condition est en effet nécessaire mais pas suffisante.

Définition 1.18 [178] Soient $G(D)$ une MGP de taille $k \times n$ avec $k < n$ et Λ_i le PGCD des mineurs d'ordre i de $G(D)$. Par convention, on posera $\Lambda_0 = 1$. Soit $\gamma_i = \left(\frac{\Lambda_i}{\Lambda_{i-1}}\right)$. Les γ_i pour i de 1 à k sont appelés *facteurs invariants* des $G(D)$.

Nous présentons maintenant des théorèmes qui permettent de reconnaître et de caractériser des MGP basiques, réduites et canoniques. Des démonstrations claires peuvent être trouvées dans [143]. Pour faciliter la compréhension, nous rappelons tout d'abord que toute matrice $H(D)$ de taille $n \times k$, à coefficients dans $F(D)$, vérifiant $G(D)H(D) = I_k$, est appelée *inverse à droite* de $G(D)$.

Théorème 1.1 Une MGP, $G(D)$, de taille $k \times n$ est basique si et seulement si une des conditions suivantes est vérifiée.

1. Les facteurs invariants de $G(D)$ valent 1.
2. Le PGCD des mineurs d'ordre k de $G(D)$ vaut 1.
3. $G(a)$ est de rang k pour tout a dans la clôture algébrique de F .
4. $G(D)$ possède un inverse à droite à coefficients dans $F[D]$.
5. Si $y(D) = x(D)G(D)$ avec $y(D) \in F[D]^n$, alors $x(D) \in F[D]^k$ (« une sortie polynomiale implique une entrée polynomiale »).
6. $G(D)$ est une sous-matrice d'une matrice unimodulaire.

Théorème 1.2 Une MGP, $G(D)$, de taille $k \times n$ est réduite si et seulement si une des conditions suivantes est vérifiée.

1. Si l'on définit la matrice indicatrice de plus haut degré de chaque ligne, $\overline{G}$, par $\overline{G}_{ij}$ comme étant le coefficient de D^{e_i} de $G_{ij}(D)$ où $e_i = \deg G_i(D)$. Alors $\overline{G}$ est de rang k .
2. $\text{extdeg } G(D) = \text{intdeg } G(D)$.
3. $\forall X(D) \in F[D]^k$

$$\deg X(D)G(D) = \max_{1 \leq i \leq k} (\deg X_i(D) + \deg G_i(D)).$$

Théorème 1.3 Une MGP, $G(D)$, est canonique si et seulement si elle est à la fois basique et réduite.

Remarque 1.7 :

Une façon efficace de calculer le PGCD des mineurs d'ordre k d'une MGP, $G(D)$, est de calculer les facteurs invariants de $G(D)$ par l'algorithme de Smith [143, annexe B], [178, §12.2].

Exemple suivi 1.5

Reprenons la matrice $G(D)$ de notre exemple guide. $G(D) = \begin{bmatrix} 1+D & D+D^2 & 1+D+D^2 \\ D^2+D^3 & 1 & 1+D+D^2+D^3 \end{bmatrix}$. Ses mineurs d'ordre $k = 2$ sont $\{1 + D^2 + D^5, 1 + D^2 + D^4 + D^5, 1 + D + D^3 + D^5\}$ et leur PGCD vaut 1. D'autre part, $\text{intdeg } G(D) = \text{extdeg } G(D) = 5$. Nous pouvons en conclure que $G(D)$ possède les propriétés suivantes.

- $G(D)$ est basique.
- $G(D)$ est réduite.
- $G(D)$ est donc canonique.
- Ceci justifie $\deg \mathcal{C} = \text{extdeg } G(D) = 5$.

1.2.3 Matrices catastrophiques

Parmi l'ensemble des MGP, seul un certain nombre permet de corriger des erreurs ; d'autres, les MGP *catastrophiques* propagent l'erreur à *l'infini* au moment du décodage. Ces matrices sont donc à proscrire pour des réalisations physiques de codeurs convolutifs.

Soient $\mathcal{C}$ un (n, k, m) -code convolutif engendré par une MGP $G(D)$, $X(D)$ un mot d'information et $Y(D) = X(D)G(D)$ le mot de code associé. Ce mot de code va être modulé et transmis sur un canal bruité, puis démodulé. Dans un deuxième temps, l'algorithme de décodage va estimer le mot de code le plus près du signal démodulé, $\tilde{Y}(D)$. Enfin, dans un troisième temps, on va décoder le mot de code estimé et retrouver un mot d'information estimé $\tilde{X}(D)$. L'objectif étant, lors du décodage, de retrouver un mot d'information estimé le plus proche du mot d'information initial. Le décodage s'effectue par la relation

$$\tilde{X}(D) = \tilde{Y}(D)K(D), \quad (1.9)$$

où $K(D)$ est l'inverse à droite de $G(D)$ ³. Notons $E_c(D)$ l'erreur sur le mot de code et $E_i(D)$ l'erreur sur le mot d'information. Par définition

$$\begin{aligned} E_c(D) &= \tilde{Y}(D) - Y(D), \\ E_i(D) &= \tilde{X}(D) - X(D). \end{aligned}$$

D'après la relation (1.9), nous obtenons

$$E_i(D) = E_c(D)K(D). \quad (1.10)$$

³La matrice $K(D)$ existe et est définie de manière unique car nous avons fait l'hypothèse au paragraphe 1.1.2 que $G(D)$ est de rang k .

Un décodage est dit *catastrophique* si pour une erreur de poids fini sur le mot de code, on obtient une erreur de poids infini sur le mot d'information. La relation (1.10) page ci-contre peut s'écrire $E_c(D) = E_i(D)G(D)$. Un décodage est donc catastrophique s'il existe un mot d'information de poids infini codé en un mot de poids fini. Ceci nous amène naturellement à la définition d'une matrice catastrophique.

Définition 1.19 Une matrice génératrice $G(D)$, de taille $k \times n$, est dite *catastrophique* s'il existe un vecteur $X(D)$ de $F(D)^k$, de poids infini, tel que

$$Y(D) = X(D)G(D)$$

soit de poids fini.

En 1968, J.L. Massey et M.K. Sain [141] démontrent le théorème suivant qui permet de tester si une matrice génératrice est catastrophique ou non.

Théorème 1.4 Soit $G(D)$, une MGP d'un (n, k, m) -code convolutif. $G(D)$ est non catastrophique si l'une des conditions suivantes est vérifiée.

1. Aucune entrée $X(D)$ de poids fini ne peut produire une sortie $Y(D)$ de poids infini.
2. Le PGCD des mineurs d'ordre k de $G(D)$ est une puissance de D .
3. $G(D)$ possède un inverse à droite dont les coefficients sont de poids fini.

Bien que ce théorème ne s'applique qu'aux MGP, [143] cite un théorème un peu plus général et adapté aux matrices génératrices dont les coefficients sont rationnels. Grâce à ce théorème et aux propriétés des matrices génératrices énoncées précédemment, on montre assez facilement que

- les matrices génératrices basiques,
- les matrices génératrices systématiques,

ne sont pas catastrophiques.

Exemple suivi 1.6

La matrice $G(D) = \begin{bmatrix} 1+D & D+D^2 & 1+D+D^2 \\ D^2+D^3 & 1 & 1+D+D^2+D^3 \end{bmatrix}$ est basique donc non catastrophique. En revanche, la matrice $G'(D) = (1+D)G(D) = \begin{bmatrix} 1+D^2 & D+D^3 & 1+D^3 \\ D^2+D^4 & 1+D & 1+D^4 \end{bmatrix}$ engendre le même code convolutif $\mathcal{C}$ mais est catastrophique. En effet, le PGCD de ses mineurs d'ordre $k = 2$ vaut $1+D^2$.

1.2.4 Les codes optimaux

Nous avons vu dans les paragraphes précédents comment choisir de « bonnes » matrices génératrices pour engendrer un code convolutif donné. Nous nous intéressons maintenant aux critères de choix d'un code convolutif. Le premier paramètre à prendre en compte est le rendement du code, donné par le rapport $R = k/n$. Il correspond au rapport du nombre de bits d'information sur le nombre de bits transmis. Pour des raisons de gain de bande passante, l'objectif est de choisir R le plus proche de 1, en revanche, plus le canal est bruité et plus R diminuera pour obtenir des performances acceptables en termes de correction. D'autre part, la complexité de l'algorithme de décodage, l'algorithme de

Viterbi [197], impose un choix de k et n petits.

L'approche algébrique permet d'unifier les codes convolutifs et les codes en blocs afin de pouvoir les comparer avec les mêmes outils. Pour un rendement donné, k/n , les codes le plus performants sont ceux qui possèdent la *distance minimale*, *i.e.* la plus petite distance entre deux mots du code, la plus grande. Dans le cas des codes convolutifs, nous devons introduire la notion de *distance libre*.

Définition 1.20 On appelle *distance libre* d'un (n, k) -code convolutif $\mathcal{C}$, l'entier, $d_{\text{libre}}(\mathcal{C})$ défini par

$$d_{\text{libre}}(\mathcal{C}) = \min_{X(D) \in \mathcal{C}} \left\{ \sum_{i=1}^n w(X_i(D)) \right\}.$$

L'approche algébrique généralise ainsi la notion de distance minimale pour les codes convolutifs. De plus, le codage bénéficie de la linéarité de la matrice génératrice ; la capacité de correction du code est d'autant plus grande que la distance entre deux mots de code est grande. Comme pour les codes en blocs, on considère la règle suivante : « plus grande est la distance libre, meilleures sont les performances du code ». On peut noter par ailleurs que si l'on considère les codes en blocs comme des codes convolutifs particuliers, la définition de la distance libre appliquée aux codes en blocs est identique à celle de distance minimale. Malheureusement, il n'y a pas de formule explicite donnant la distance libre d'un code convolutif, elle doit être déterminée de façon expérimentale. Néanmoins, il existe des bornes théoriques sur la distance libre. Une approche à base de série de Hilbert permet, de mettre en évidence une borne sur la distance libre d'un (n, k, m) -code convolutif $\mathcal{C}$ [143].

$$d_{\text{libre}}(\mathcal{C}) \leq \min_{L \geq 0} \Delta_q(n(L+1), k(L+1) - m),$$

où $\Delta_q(n, k)$ est la plus grande distance minimale d'un (n, k) -code en bloc linéaire sur $GF(q)$.

Définition 1.21 Un (n, k, m) -code convolutif $\mathcal{C}$ dont la distance libre atteint la plus grande distance libre pour un (n, k, m) -code convolutif est dit *optimal*.

D'autre part nous noterons $\Delta(n, k, m)$ la plus grande distance libre possible pour un (n, k, m) -codeur convolutif, *i.e.*

$$\Delta(n, k, m) = \max_{\mathcal{C}} \{d_{\text{libre}}(\mathcal{C}) \mid \mathcal{C} \text{ } (n, k, m)\text{-code convolutif}\}.$$

Nous rappelons maintenant quelques codes convolutifs optimaux qui nous ont servi pour tester les algorithmes de reconstruction. Les polynômes des matrices génératrices seront exprimés en notation octale, *i.e.* le polynôme $D^6 + D^5 + D^3 + D + 1$ sera noté 153 en octal. Cette liste, non exhaustive, peut être complétée par [109, 123, 126, 152].

Remarque 1.8 :

Le code convolutif (133 171) mis en évidence par J.P. Odenwalder [148] en 1970 est le code correcteur utilisé par la navette Voyager dans les années 80 pour coder ses transmissions vers la Terre.

m	$\Delta(2, 1, m)$	$G(D)$	m	$\Delta(3, 1, m)$	$G(D)$
0	2	(1 1)	0	3	(1 1 1)
1	3	(1 3)	1	5	(1 3 3)
2	5	(5 7)	2	8	(5 7 7)
3	6	(13 17)	3	10	(13 15 17)
4	7	(23 35)	4	12	(25 33 37)
5	8	(53 75)	5	13	(47 53 75)
6	10	(133 171)	6	15	(133 145 175)
8	12	(561 753)	7	16	(225 331 367)
10	14	(2355 3661)	8	18	(557 663 711)
			9	20	(1117 1365 1633)
			10	22	(2353 2671 3175)

TAB. 1.1 – $(2, 1, m)$ et $(3, 1, m)$ -codes et convolutifs optimaux**Exemple suivi 1.7**

Le $(3, 2, 5)$ -codeur convolutif $\mathcal{C}_1$ a pour matrice génératrice codée en octal $\begin{pmatrix} 3 & 6 & 7 \\ 14 & 1 & 17 \end{pmatrix}$; le code convolutif engendré $\mathcal{C}$ est optimal et possède une distance libre de 6.

1.3 Les codes convolutifs poinçonnés

Nous allons maintenant introduire les *codes convolutifs poinçonnés*. Ces codes ont été introduits par J.B. Cain, G.C. Clark et J.M. Geist [41] en 1979. Ils sont très faciles à construire à partir de codes, appelés *codes parents* et offrent des distances libres plus grande à rendement égal. De plus, de par leur structure particulière, ils sont plus faciles à implémenter et moins coûteux à décoder par l'algorithme de Viterbi [197] ou par décodage séquentiel [23, 24, 208]. Les codes poinçonnés sont donc très répandus et sont plus attractifs que les codes « non poinçonnés ».

1.3.1 Définition des codes poinçonnés

Jusqu'à maintenant, nous avons considéré les codeurs convolutifs comme une application qui transforme globalement une séquence de mots d'information en une séquence de mots de code. Localement, on peut voir un codeur convolutif comme une machine cadencée, qui prend à chaque « top » d'horloge, un mot d'information de k bits en entrée et qui sort un mot de code de n bits. Si l'on ralentit d'un facteur M la cadence de la machine, elle prend alors en entrée M mots d'information de k bits et sort M mots de code de n bits. Le (n, k, m) -code convolutif $\mathcal{C}$ engendré par le codeur est appelé *code parent*. Le codeur « ralenti », engendre un code convolutif de paramètres (nM, kM, m) , de même rendement et de même distance libre que le code parent. Nous avons alors effectué un *regroupement de profondeur* M .

m	$\Delta(4, 1, m)$	$G(D)$
0	4	(1 1 1 1)
1	7	(1 3 3 3)
2	10	(5 7 7 7)
3	13	(13 15 15 17)
4	16	(25 27 33 37)
5	18	(53 67 71 75)
6	20	(135 135 147 163)
7	22	(235 275 313 357)
8	24	(463 535 733 745)
9	27	(1117 1365 1633 1653)
10	29	(2387 2353 2671 3175)

m	$\Delta(3, 2, m)$	$G(D)$
0	2	$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$
2	3	$\begin{pmatrix} 3 & 2 & 3 \\ 2 & 1 & 1 \end{pmatrix}$
3	4	$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 1 & 7 \end{pmatrix}$
4	5	$\begin{pmatrix} 7 & 4 & 1 \\ 2 & 5 & 7 \end{pmatrix}$
5	6	$\begin{pmatrix} 3 & 6 & 7 \\ 14 & 1 & 17 \end{pmatrix}$
6	7	$\begin{pmatrix} 13 & 6 & 13 \\ 6 & 13 & 17 \end{pmatrix}$
7	8	$\begin{pmatrix} 3 & 6 & 15 \\ 34 & 31 & 17 \end{pmatrix}$
9	9	$\begin{pmatrix} 25 & 30 & 17 \\ 50 & 7 & 65 \end{pmatrix}$
10	10	$\begin{pmatrix} 63 & 54 & 31 \\ 26 & 53 & 43 \end{pmatrix}$

m	$\Delta(4, 3, m)$	$G(D)$
0	2	$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$
2	3	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 \\ 0 & 3 & 1 & 0 \end{pmatrix}$
3	4	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 3 & 2 & 1 \\ 0 & 2 & 5 & 5 \\ 3 & 2 & 2 & 3 \end{pmatrix}$
5	5	$\begin{pmatrix} 4 & 3 & 0 & 7 \\ 0 & 2 & 5 & 5 \\ 3 & 4 & 0 & 7 \end{pmatrix}$
6	6	$\begin{pmatrix} 6 & 1 & 4 & 3 \\ 2 & 6 & 7 & 1 \end{pmatrix}$
8	7	$\begin{pmatrix} 7 & 6 & 2 & 1 \\ 14 & 5 & 0 & 15 \\ 10 & 4 & 17 & 1 \end{pmatrix}$
9	8	$\begin{pmatrix} 1 & 14 & 16 & 3 \\ 10 & 13 & 2 & 7 \\ 16 & 0 & 3 & 13 \end{pmatrix}$

TAB. 1.2 – $(4, 1, m)$, $(3, 2, m)$ et $(4, 3, m)$ -codes convolutifs optimaux

Exemple suivi 1.8

En divisant la fréquence du codeur $\mathcal{C}_1$ par 2, *i.e.* en effectuant un regroupement de profondeur 2, nous appliquons la transformation suivante

$$\begin{pmatrix} x_1(0) & x_1(2) & x_1(4) & \dots \\ x_2(0) & x_2(2) & x_2(4) & \dots \\ x_1(1) & x_1(3) & x_1(5) & \dots \\ x_2(1) & x_2(3) & x_2(5) & \dots \end{pmatrix} \Rightarrow \begin{pmatrix} y_1(0) & y_1(2) & y_1(4) & \dots \\ y_2(0) & y_2(2) & y_2(4) & \dots \\ y_3(0) & y_3(2) & y_3(4) & \dots \\ y_1(1) & y_1(3) & y_1(5) & \dots \\ y_2(1) & y_2(3) & y_2(5) & \dots \\ y_3(1) & y_3(3) & y_3(5) & \dots \end{pmatrix}.$$

Le *poinçonnage* consiste à ne pas pas envoyer tous les bits consistuant les mots de codes, faisant ainsi augmenter artificiellement le rendement du code initial. Un poinçonnage est défini de manière unique par le choix d'un *motif de poinçonnage*, c'est-à-dire une matrice P de taille $n \times M$ à coefficients dans $GF(2)$. Nous noterons N le nombre de coefficients de P à 1 ; N est aussi appelé *densité* de la matrice de poinçonnage. Les coefficients à 1 dénotent les bits qui seront envoyés et ceux à 0 dénotent les bits qui ne seront pas envoyés.

Par exemple, en prenant $M = 2$ et le codeur $\mathcal{C}_1$, la matrice de poinçonnage

$$P = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \end{pmatrix},$$

effectue la transformation

$$\begin{pmatrix} y_1(0) & y_1(1) & y_1(2) & y_1(3) & \dots \\ y_2(0) & y_2(1) & y_2(2) & y_2(3) & \dots \\ y_3(0) & y_3(1) & y_3(2) & y_3(3) & \dots \end{pmatrix} \Rightarrow \begin{pmatrix} y_1(0) & & y_1(2) & & \dots \\ y_2(0) & y_2(1) & y_2(2) & y_2(3) & \dots \\ y_3(0) & y_3(1) & y_3(2) & y_3(3) & \dots \end{pmatrix}.$$

En résumant le regroupement et le poinçonnage défini par la matrice P , nous obtenons la transformation

$$\begin{pmatrix} x_1(0) & x_1(2) & x_1(4) & \dots \\ x_2(0) & x_2(2) & x_2(4) & \dots \\ x_1(1) & x_1(3) & x_1(5) & \dots \\ x_2(1) & x_2(3) & x_2(5) & \dots \end{pmatrix} \Rightarrow \begin{pmatrix} y_1(0) & y_1(2) & y_1(4) & \dots \\ y_2(0) & y_2(2) & y_2(4) & \dots \\ y_3(0) & y_3(2) & y_3(4) & \dots \\ & y_2(1) & y_2(3) & y_2(5) & \dots \\ & y_3(1) & y_3(3) & y_3(5) & \dots \end{pmatrix}.$$

Cette transformation correspond à un $(4, 5, 5)$ -code convolutif.

Le poinçonnage peut se résumer à deux étapes. La première est un regroupement de profondeur M à partir d'un code (n, k, m) -code parent. À l'issue, nous obtenons un (nM, kM, m) -code convolutif de même distance libre que le code parent. La deuxième étape est un poinçonnage à partir d'un motif de poinçonnage P , matrice de dimensions $n \times M$ et de densité N . En pratique, nous observons que la sortie du poinçonnage est un (N, kM, m) -code convolutif de distance libre souvent très proche ou égale à $\Delta(N, kM, m)$ et en moyenne 2^{M-1} fois moins coûteux pour le décodage avec l'algorithme de Viterbi [197] qu'un (N, kM, m) -code convolutif. Nous voyons maintenant voir comment obtenir la matrice génératrice d'un code poinçonné à partir de la matrice génératrice du code parent.

1.3.2 Construction par regroupement

Dans ce paragraphe, nous expliquons comment construire la matrice génératrice d'un code poinçonné à partir celle du code initial à l'aide de notre exemple suivi. Pour ce faire, nous procédons en deux étapes : nous construisons tout d'abord une matrice infinie, puis nous en prenons ensuite une sous-matrice de taille désirée.

Pour construire la matrice après regroupement, revenons à la définition d'un mot de code (cf. relation 1.5 page 29),

$$Y(D) = X(D)G(D) = X(0)G(D) + X(1)DG(D) + X(2)D^2G(D) + \dots$$

Une définition d'un code convolutif (cf. définition 1.1 page 27) est l'ensemble des mots de code possibles générés par l'ensemble des mots d'information ; c'est aussi l'ensemble de toutes les combinaisons possibles des lignes de la matrice définie par

$$\overline{G}(D) = \begin{pmatrix} G(D) \\ DG(D) \\ D^2G(D) \\ D^3G(D) \\ \vdots \end{pmatrix}. \quad (1.11)$$

Récrivons la matrice génératrice du codeur $\mathcal{C}_1$ sous la forme appelée *décomposition polynomiale*,

$$G(D) = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} D + \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} D^2 + \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} D^3. \quad (1.12)$$

En combinant les relations (1.11) et (1.12), nous obtenons pour la matrice génératrice du codeur $\mathcal{C}_1$,

$$\overline{G}(D) = \begin{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} D & \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} D^2 & \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} D^3 & & \\ & \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} D & \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} D^2 & \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} D^3 & \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} D^4 & \\ & & \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} D^2 & \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} D^3 & \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} D^4 & \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} D^5 \\ & \vdots & & & & \ddots \end{pmatrix}.$$

La $i^{\text{ème}}$ colonne de $G(D)$ correspond au processus de codage au $i^{\text{ème}}$ top d'horloge, c'est-à-dire à l'entrée du codeur. De même, la $i^{\text{ème}}$ ligne de $G(D)$ correspond à la sortie du codeur au $i^{\text{ème}}$ top d'horloge. Si l'on divise la cadence de l'horloge par M , les colonnes j et les lignes $G_j(D)$ de $G(D)$ pour $j \in [(i-1)M \dots iM[$, représentent respectivement l'entrée et la sortie du codeur au $i^{\text{ème}}$ top d'horloge. Nous noterons $G^{[M]}(D)$ la matrice polynomiale de taille $nM \times kM$ formée par les lignes de $G(D)$ représentant le top d'horloge « 1 ». Dans le cas $M = 2$, nous avons

$$G^{[2]}(D) = \begin{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} D & \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} D \\ \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} D & \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} D & \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} D & \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} D^2 \end{pmatrix}.$$

$G^{[2]}(D)$ peut s'exprimer en fonction de sa décomposition polynomiale par

$$G^{[2]}(D) = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} D + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} D^2,$$

ou encore par

$$G^{[2]}(D) = \begin{pmatrix} 1 & D & 1+D & 1 & 1 & 1 \\ D & 1 & 1+D & D & 0 & 1+D \\ D & D & D & 1 & D & 1+D \\ D^2 & 0 & D+D^2 & D & 1 & 1+D \end{pmatrix}. \quad (1.13)$$

Cette méthode nous permet d'évaluer n'importe quelle matrice regroupée d'ordre M , $G^{[M]}$, quel que soit M .

1.3.3 Approche algébrique du regroupement

Nous présentons maintenant une technique plus directe pour construire les matrices $G^{[M]}$ en utilisant la $M^{\text{ième}}$ *décomposition polyphase* des séries formelles.

Définition 1.22 [143, p. 1118] *Soit*

$$f(D) = a(0) + a(1)D + a(2)D^2 + \dots,$$

une série causale d'indéterminée D . On appelle $M^{\text{ième}}$ décomposition polyphase de $f(D)$, la liste $(f_{0,M}(D), f_{1,M}(D), \dots, f_{M-1,M}(D))$ de longueur M définie par

$$\left(\sum_{k \geq 0} a(kM)D^k, \sum_{k \geq 0} a(kM+1)D^k, \dots, \sum_{k \geq 0} a(kM+(M-1))D^k \right).$$

$f_{j,M}(D)$ est appelé $(j, M)^{\text{ième}}$ *composante polyphase* de $f(D)$.

La $(j, M)^{\text{ième}}$ composante polyphase de $f(D)$ est une série dont les coefficients sont obtenus en commençant au $j^{\text{ième}}$ coefficient de $f(D)$ et en comptant de M en M .

Exemple 1.9

Soit la série formelle

$$f(D) = 3 + D + 4D^2 + D^3 + 5D^4 + 9D^5 + 2D^6 + 6D^7,$$

alors la troisième décomposition polyphase de $f(D)$ est

$$(3 + D + 2D^2, 1 + 5D + 6D^2, 4 + 9D).$$

Quand le contexte est clair nous abrégons $f_{j,n}(D)$ en $f(j)$.

Définition 1.23 *Soit $f(D) = a(0) + a(1)D + a(2)D^2 + \dots$, une série causale d'indéterminée D et $(f(0), f(1), \dots, f(M-1))$ la $M^{\text{ième}}$ décomposition polyphase de $f(D)$. La matrice polynomiale de taille $M \times M$, $f^{[M]}(D)$, définie par*

$$f^{[M]}(D) = \begin{pmatrix} f(0) & f(1) & \dots & f(M-1) \\ Df(M-1) & f(0) & \dots & f(M-2) \\ Df(M-2) & Df(M-1) & \dots & f(M-3) \\ \vdots & \vdots & \ddots & \vdots \\ Df(1) & Df(2) & \dots & f(0) \end{pmatrix},$$

est appelée $M^{\text{ième}}$ matrice polycyclique pseudocirculante (PCPC) associée à $f(D)$.

Soit $\mathcal{C}$ un (n, k) -code convolutif, $\mathcal{C}^{[M]}$, le (nM, kM) -code convolutif construit par regroupement de profondeur M à partir de $\mathcal{C}$. Le théorème suivant permet de construire une matrice génératrice $G^{[M]}(D)$ pour $\mathcal{C}^{[M]}$ à partir d'une matrice génératrice $G(D)$ de $\mathcal{C}$.

Nous devons maintenant définir la notion d'entrelaceur. Un *entrelacement de profondeur* M est une permutation sur les colonnes dont les indices sont pris en comptant de M en M . Par exemple, l'entrelacement de profondeur 3 de 6 colonnes est

$$(1, 2, 3, 4, 5, 6) \implies (1, 4, 2, 5, 3, 6).$$

Théorème 1.5 [143, p. 1120] *Dans les conditions précédentes, si $G(D) = (G_{ij}(D))$ alors une matrice génératrice pour $\mathcal{C}^{[M]}$, $G^{[M]}(D)$, peut être obtenue en remplaçant les $G_{ij}(D)$ par leur $M^{\text{ième}}$ PCPC et en entrelaçant les lignes et les colonnes à la profondeur M .*

Exemple suivi 1.10

En appliquant le théorème 1.5 à notre exemple guide, nous obtenons

$G_{ij}(D)$	2 ^{ième} décomposition polyphase	PCPC
$G_{11} = 1 + D$	$(1, 1)$	$\begin{pmatrix} 1 & 1 \\ D & 1 \end{pmatrix}$
$G_{12} = D + D^2$	$(D, 1)$	$\begin{pmatrix} D & 1 \\ D & D \end{pmatrix}$
$G_{13} = 1 + D + D^2$	$(1 + D, 1)$	$\begin{pmatrix} 1+D & 1 \\ D & 1+D \end{pmatrix}$
$G_{21} = D^2 + D^3$	(D, D)	$\begin{pmatrix} D & D \\ D^2 & D \end{pmatrix}$
$G_{22} = 1$	$(1, 0)$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
$G_{23} = 1 + D + D^2 + D^3$	$(1 + D, 1 + D)$	$\begin{pmatrix} 1+D & 1+D \\ D+D^2 & 1+D \end{pmatrix}$

Avant entrelacement nous avons

$$G' = \begin{pmatrix} 1 & 1 & D & 1 & 1+D & 1 \\ D & 1 & D & D & D & 1+D \\ D & D & 1 & 0 & 1+D & 1+D \\ D^2 & D & 0 & 1 & D+D^2 & 1+D \end{pmatrix}.$$

Entrelacement de profondeur 2 sur les lignes : $(1, 2, 3, 4) \implies (1, 3, 2, 4)$.

Entrelacement de profondeur 2 sur les colonnes : $(1, 2, 3, 4, 5, 6) \implies (1, 3, 5, 2, 4, 6)$.

Après entrelacement

$$G^{[2]}(D) = \begin{pmatrix} 1 & D & 1+D & 1 & 1 & 1 \\ D & 1 & 1+D & D & 0 & 1+D \\ D & D & D & 1 & D & 1+D \\ D^2 & 0 & D+D^2 & D & 1 & 1+D \end{pmatrix}. \quad (1.14)$$

Nous retrouvons bien la matrice (1.13) page précédente obtenue avec la méthode de regroupement.

Nous pouvons montrer le théorème suivant [194].

Théorème 1.6 *Si $G(D)$ est une matrice génératrice canonique pour $\mathcal{C}$, alors $G^{[M]}(D)$ est une matrice génératrice canonique pour $\mathcal{C}_1^{[M]}$. $\mathcal{C}^{[M]}$ et $G^{[M]}(D)$ ont donc même degré.*

Pareillement, quelle que soit la valeur de M , les codes regroupés ont tous le même ensemble de mots de code que le code parent et donc la même distance libre.

1.3.4 Poinçonnage

Tandis que les codes regroupés permettent de conserver de « bonnes » propriétés des codes parents, le transfert de ces mêmes propriétés n'est pas immédiat pour les codes poinçonnés. Chaque code poinçonné doit faire l'objet d'une étude particulière. Néanmoins, l'étude des codeurs poinçonnés catastrophique a été envisagée [149].

Soient $G(D)$ la matrice génératrice d'un (n, k) -code convolutif $\mathcal{C}$, P une matrice de poinçonnement de taille $n \times M$ et $G^{[M]}(D)$ une matrice de code regroupé à la profondeur M ayant $\mathcal{C}$ comme code parent. $G^{[M]}(D)$ est de taille $kM \times nM$; il existe donc une correspondance naturelle entre les coefficients de P et les colonnes de $G^{[M]}(D)$. Soit σ la bijection définie par

$$(i, j) \longrightarrow \sigma(i, j) = i + n(j - 1).$$

On peut alors associer à chaque coefficient P_{ij} la colonne $\sigma(i, j)$ de $G^{[M]}(D)$. Cette correspondance définit une matrice génératrice $G_p(D)$ du code poinçonné associé au motif P ayant $\mathcal{C}$ comme code parent.

Définition 1.24 Dans les conditions précédentes, $G_p(D)$ est construite à partir de $G^{[M]}(D)$ en effaçant ses colonnes $\sigma(i, j)$ lorsque les coefficients $P_{ij} = 0$. $G_p(D)$ est appelée version P -poinçonnée de $\mathcal{C}$.

Exemple suivi 1.11

Pour obtenir la matrice génératrice du code P -poinçonné du code parent $\mathcal{C}$, on efface la colonne $\sigma(1, 2) = 4$ de la matrice (1.14) page précédente.

$$G_P^{[2]}(D) = \begin{pmatrix} 1 & D & 1+D & 1 & 1 \\ D & 1 & 1+D & 0 & 1+D \\ D & D & D & D & 1+D \\ D^2 & 0 & D+D^2 & 1 & 1+D \end{pmatrix}.$$

1.3.5 Les « bons » codes poinçonnés

La recherche exhaustive des codes poinçonnés a été initiée par J.B. Cain et *al.* [41] et développée par Y. Yasuda et *al.* [208] et J. Hagenbauer [97] pour des codes de faible mémoire. Ces résultats ont été étendus par G. Bégin et D. Haccoun [23, 24] pour des codes de mémoire plus grande. Les codes poinçonnés étant déterminés par recherche exhaustive, nous regroupons ici quelques « bons » codes poinçonnés. Nous reprenons la notation octale comme expliquée au paragraphe 1.2.4. Les codes donnés ici sont ceux recensés par É. Filiol dans sa thèse [71, annexe C], issus des travaux de G. Bégin et D. Haccoun [23, 24].

Code parent		Code poinçonné		Code parent		Code poinçonné	
m	$G(D)$	P	d_{libre}	m	$G(D)$	P	d_{libre}
2	(5 7)	$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$	3	14	(55367 63121)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	10
3	(15 17)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	4	15	(111653 145665)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	10
4	(23 35)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	4	16	(347241 246277)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	12
5	(53 75)	$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$	6	17	(506477 673711)	$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$	12
6	(133 171)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	6	18	(1352755 1771563)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	12
7	(247 371)	$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$	7	19	(2451321 3546713)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	12
8	(561 753)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	7	19	(2142513 3276177)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	13
9	(1167 1545)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	7	20	(6567413 5322305)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	12
10	(2335 3661)	$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$	8	21	(15724153 12076311)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	13
11	(4335 5723)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	9	22	(33455341 24247063)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	14
12	(10533 17661)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	9	23	(55076157 75501351)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	15
13	(21675 27123)	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	10				

TAB. 1.3 – Codes poinçonnées de taux $\frac{2}{3}$

Code parent		Code poinçonné		Code parent		Code poinçonné	
m	$G(D)$	P	d_{libre}	m	$G(D)$	P	d_{libre}
2	(5 7)	$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$	3	14	(5536763121)	$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$	8
3	(15 17)	$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$	4	15	(111653 145665)	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	8
4	(23 35)	$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$	3	16	(347241 246277)	$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$	8
5	(53 75)	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	4	17	(506477 673711)	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	9
6	(133 171)	$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$	5	18	(1352755 1771563)	$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$	10
7	(247 371)	$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$	6	19	(2451321 3546713)	$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$	10
8	(561 753)	$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$	6	19	(2142513 3276177)	$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$	10
9	(1167 1545)	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	6	20	(6567413 5322305)	$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$	10
10	(2335 3661)	$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$	6	21	(15724153 12076311)	$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$	11
11	(4335 5723)	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	7	22	(33455341 24247063)	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	12
12	(10533 17661)	$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$	7	23	(55076157 75501351)	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	13
13	(21675 27123)	$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$	7				

TAB. 1.4 – Codes poinçonnés de taux $\frac{3}{4}$

Code parent		Code poinçonné		Code parent		Code poinçonné	
m	$G(D)$	P	d_{libre}	m	$G(D)$	P	d_{libre}
2	(5 7)	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	2	12	(10533 17661)	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	6
3	(15 17)	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	3	13	(21675 27123)	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	7
4	(2335)	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$	3	14	(5536763121)	$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$	7
5	(53 75)	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$	4	15	(111653 145665)	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$	8
6	(133 171)	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$	4	16	(347241 246277)	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$	8
7	(247 371)	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$	5	17	(506477 673711)	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$	8
8	(561 753)	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$	5	18	(1352755 1771563)	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	8
9	(1167 1545)	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$	4	19	(2451321 3546713)	$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$	9
10	(2335 3661)	$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$	5	19	(2142513 3276177)	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	9
11	(4335 5723)	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	6				

TAB. 1.5 – Codes poinçonnés de taux $\frac{4}{5}$

Chapitre 2

Reconstruction des codes linéaires

« Toute connaissance dégénère en probabilité. »

David Hume (*Traité de la nature humaine*)

Sommaire

2.1 Étude algébrique	51
2.1.1 Formalisation du problème de reconstruction	51
2.1.2 Algorithme de Gauss randomisé	57
2.2 Algorithmes de reconstruction	60
2.2.1 Canal sans erreur	60
2.2.2 Canal avec erreur	63
2.3 Analyse des algorithmes	65
2.3.1 Canal sans erreur	67
2.3.2 Canal avec erreur	69
2.4 Résultats expérimentaux	77
2.4.1 Impact du poids d'une relation de parité	77
2.4.2 Impact du nombre de mots de code interceptés	78
2.4.3 Impact de l'erreur	79
2.4.4 Des codes bien réels	80
2.5 Conclusion et perspectives	82

DANS ce chapitre, nous nous intéressons à la reconstruction des codes en blocs. Ceux-ci ont été présentés comme des cas particuliers de codes convolutifs au regard du formalisme introduit au chapitre 1. Sous cet angle, ils apparaissent comme plus simples à reconstruire que les codes convolutifs ; en fait, les techniques de reconstruction des codes en blocs linéaires nous servent de briques de base pour reconstruire les codes convolutifs et les turbo-codes. Notons $F = GF(2)$, $\mathcal{C}$ un code en bloc linéaire de dimension k dans F^n de rendement $\frac{k}{n}$ et de matrice génératrice¹ G de dimension $n \times k$ à coefficients dans F . Soient $(y_i)_{i=1\dots N}$ les mots de codes transmis sur le canal binaire, $(e_i)_{i=1\dots N}$ l'erreur binaire

¹Nous utiliserons à partir de maintenant la convention d'écriture matricielle française.

introduite par ce canal et $(\tilde{y}_i)_{i=0\dots N}$ les mots de code bruités interceptés par l'observateur. Nous avons la relation

$$\tilde{y}_i(j) = y_i(j) \oplus e_i(j) \quad \forall i = 1 \dots N, \quad \forall j = 1 \dots n,$$

en accord avec le modèle de canal binaire défini en introduction. Un choix de canal représentatif pour les algorithmes de reconstruction de codes en blocs est le canal binaire symétrique. Ce choix est justifié au paragraphe 2.5. L'étude théorique est donc menée en se plaçant dans le cas d'un canal binaire symétrique en ayant toujours à l'esprit que les résultats obtenus produisent une borne minimale pour les performances attendues sur l'ensemble des canaux. Dans ce cadre, les $e_i(j)$ sont des variables aléatoires indépendantes à valeurs dans F et suivent une loi uniforme de paramètre ε , le taux d'erreur binaire (TEB) du canal, *i.e.*

$$\mathcal{P}r(e_i(j) = 1) = \varepsilon \quad \text{et} \quad \mathcal{P}r(e_i(j) = 0) = 1 - \varepsilon.$$

Nous notons par ailleurs $\mathcal{H}$, le code dual de $\mathcal{C}$, c'est-à-dire le sous-espace de F^n de dimension $(n - k)$ et orthogonal à $\mathcal{C}$, *i.e.* $\mathcal{C}^\perp = \mathcal{H}$. Par définition, $\forall c \in \mathcal{C}$ et $\forall h \in \mathcal{H}$, $\langle h, c \rangle = \langle c, h \rangle = 0$ où $\langle \cdot, \cdot \rangle$ désigne le produit scalaire usuel. Les vecteurs de $\mathcal{H}$ sont appelés *relations de parité*. Il résulte de la définition de $\mathcal{H}$ que reconstruire le code $\mathcal{C}$ est équivalent à reconstruire son dual, c'est-à-dire à retrouver une base de $\mathcal{H}$ soit $(n - k)$ relations de parité indépendantes. Nous abordons le problème de la reconstruction de $\mathcal{C}$ dans ce sens.

Dans la majorité des schémas de codage de canal, les codes correcteurs d'erreurs sont généralement suivis d'un *entrelaceur*. Il existe différentes classes d'entrelaceurs, mais nous nous restreindrons aux plus usités, c'est-à-dire aux *entrelaceurs blocs*. De tels objets ont pour vocation à mélanger les bits entre eux afin notamment de lisser la puissance émise sur le canal mais surtout de répartir l'erreur entre différents mots de code quand elle apparaît par paquets. Ceci nous donne un argument de plus pour choisir un modèle de canal binaire symétrique. En effet, l'erreur induite par les paquets devient uniforme, à une approximation près, après désentrelacement.

Définition 2.1 *Un entrelaceur $\mathcal{E}$, de taille p est une application bijective linéaire de F^p dans F^p qui conserve le poids de Hamming ,*

$$w(x) = w(\mathcal{E}(x)) \quad \forall x \in F^p.$$

Un entrelaceur $\mathcal{E}$ est défini de manière univoque par la donnée d'une permutation σ de $[1, p]$ telle que

$$(\mathcal{E}(x))(i) = x(\sigma^{-1}(i)) \quad \forall x \in F^p,$$

i.e. le $i^{\text{ème}}$ bit de x est envoyé sur le $\sigma(i)^{\text{ème}}$ bit de $\mathcal{E}(x)$. La valeur p est aussi appelé profondeur de l'entrelaceur $\mathcal{E}$.

Nous faisons l'hypothèse que la profondeur de l'entrelaceur en aval du codeur de canal est un multiple de la taille d'un mot de code, *i.e.* $\exists \alpha \in \mathbb{N}^*$ tel que $p = \alpha n$. Dans le cas contraire, nous pouvons nous ramener à la configuration précédente en reconstruisant le code $\mathcal{C}^{PPCM(n,p)}$, code produit de $\mathcal{C}$, suivi d'un entrelaceur $\mathcal{E}'$ de profondeur $PPCM(n,p)$ composé de $PPCM(n,p)/p$ entrelaceurs identiques $\mathcal{E}$. Nous posons alors $\alpha = PPCM(n,p)/n$.

En résumé, si il n'y a pas d'entrelaceur, nous devons reconstruire le code en bloc linéaire $\mathcal{C}$ de matrice génératrice G , sinon la profondeur de l'entrelaceur est un multiple de la taille de $\mathcal{C}$ et nous devons reconstruire le code engendré par la matrice génératrice G' de taille $\alpha n \times \alpha k$ telle que

$$G' = P \times \begin{pmatrix} G & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & G \end{pmatrix},$$

où P est la matrice de l'entrelaceur.

Deux types d'approches, introduites par G. Planquette [154] sont possibles pour répondre au problème de la reconstruction des codes en blocs linéaires. La première consiste à faire des hypothèses sur les relations de parité et à les valider à partir de tests statistiques. La deuxième consiste à détecter une variation de rang pour une matrice bien choisie. La dernière méthode fait l'objet de deux approches différentes : la première consiste à rechercher des mots de poids faible dans un code particulier, dans l'esprit des travaux de A. Valembois [195, 196] et M. Cluzeau [50, 51] et la seconde plus directe, à adapter l'algorithme de Gauss, comme l'ont suggéré G. Burel et R. Gautier [37] ainsi que G. Sicot et S. Houcke [173, 174, 175]. Nous avons fait le choix d'adopter l'approche de G. Sicot et S. Houcke pour deux raisons : tout d'abord, celle initiée par G. Planquette et A. Valembois a été revisitée très récemment, notamment par M. Cluzeau et d'autre part, l'analyse de l'algorithme de Sicot-Houcke sous l'angle algébrique n'avait pas encore été étudiée. Néanmoins, la comparaison des deux approches reste à faire et fait l'objet de travaux en cours. De ce fait, nous ne parlons pas de l'approche de A. Valembois dans ce chapitre. Nous reformulons alors l'algorithme de Sicot-Houcke dans le formalisme du chapitre 1 et proposons une analyse fine des mécanismes sous-jacents permettant de mettre en évidence des bornes de complexité. Pour ne pas faire perdre le fil au lecteur, nous avons pris le parti d'évaluer certaines probabilités *a posteriori* dans le paragraphe « analyse des algorithmes ». Les résultats de ces travaux ont fait l'objet de la publication [19].

2.1 Étude algébrique

2.1.1 Formalisation du problème de reconstruction

Soit $\mathcal{C}$ un code en bloc linéaire de longueur n et de dimension k , de matrice génératrice G et de code dual $\mathcal{H}$. Soit $(\tilde{B}_i)_{i=1 \dots (N+1)n}$ la concaténation binaire de $(N+1)$ mots de codes $(\tilde{y}_i)_{i=1 \dots N+1}$, bruités par le canal, aussi appelé *train binaire*. L'adversaire intercepte le train binaire à partir d'un certain indice et obtient ainsi $(\tilde{B}'_i)_{i=1 \dots (N+1)n-d} = (\tilde{B}_{i+d})_{i=1 \dots (N+1)n-d}$, où $d \in \{0, n-1\}$ est appelé *facteur de désynchronisation*.

Reconstruire le code en bloc linéaire $\mathcal{C}$ consiste à prendre en entrée $(\tilde{B}_{i+d})_{i=1 \dots (N+1)n-d}$, le train binaire intercepté, et à renvoyer les paramètres n et k ainsi que $(n-k)$ vecteurs d'une base de $\mathcal{H}$, *i.e.* $(n-k)$ *relations de parité* indépendantes.

Pour ce faire, l'observateur émet une hypothèse (n_e, d_e) sur (n, d) et construit une matrice $\tilde{C}(n_e, d_e)$ de taille $N \times n$ en espérant qu'à chaque ligne corresponde un mot de code bruité. Nous expliquons maintenant comment construire la matrice $\tilde{C}(n_e, d_e)$. Notre objectif est donc de retrouver le dual $\mathcal{H}$ du code $\mathcal{C}$ avec la seule connaissance des mots de

code bruités $(\tilde{y}_i)_{i=1\dots N}$. Pour cela, nous disposons de la relation qui définit le dual de $\mathcal{C}$,

$$\langle h, y_i \rangle = \langle y_i, h \rangle = 0 \quad \forall h \in \mathcal{H}, \quad (2.1)$$

qu'il nous faut exprimer sous forme matricielle. Nous nous sommes placés dans le cas très général d'un observateur qui ne possède aucune information *a priori* relative à la transmission qu'il observe. Il n'y a donc aucune raison *a priori* pour qu'il connaisse précisément le début des mots de code dans le train binaire qu'il intercepte. Il nous faut donc prendre en compte le facteur de désynchronisation. Supposons tout d'abord que l'observateur connaisse la valeur de n , *i.e.* $n_e = n$. Il doit alors faire une hypothèse d_e sur d . Pour que chaque ligne corresponde à un mot de code, il « jette » les $(n - d_e)$ premiers bits de $(\tilde{B}'_i)$ et commence à remplir la matrice $\tilde{C}(d_e)$ à partir de $\tilde{B}'_{(n-d_e+1)}$, en changeant de ligne tous les n bits. En gardant à l'esprit que $\tilde{B}'_{(n-d)} = \tilde{y}_1(n)$, nous raisonnons par rapport à d .

- Si $d_e > d$ alors le premier bit de la matrice est

$$\tilde{B}'_{(n-d_e+1)} = \tilde{B}'_{(n-d)-(d_e-d)+1} = \tilde{y}_1(n+1-(d_e-d)),$$

- si $d_e \leq d$ alors le premier bit de la matrice est

$$\tilde{B}'_{(n-d_e+1)} = \tilde{B}'_{(n-d)+(d-d_e)+1} = \tilde{y}_2(1+(d-d_e)).$$

Exemple 2.1

Pour bien visualiser cette construction, considérons un train binaire composé de la concaténation de mots d'un code en bloc linéaire de longueur 7 connue de l'observateur,

$$(\tilde{B}_i) \quad |\tilde{y}_1(1)\tilde{y}_1(2)\dots\tilde{y}_1(7)||\tilde{y}_2(1)\dots\tilde{y}_2(7)||\tilde{y}_3(1)\dots$$

Supposons que celui-ci commence l'interception à partir du 3^{ème} bit, *i.e.* $d=2$, il obtient alors

$$(\tilde{B}'_i) \quad |\tilde{y}_1(3)\tilde{y}_1(4)\dots\tilde{y}_1(7)\tilde{y}_2(1)\tilde{y}_2(2)||\tilde{y}_2(3)\dots\tilde{y}_2(7)\tilde{y}_3(1)\tilde{y}_3(2)||\dots$$

Il tente alors une première hypothèse, $d_e = 4$. Il jette alors les $(n - d_e) = 3$ premiers bits de $(\tilde{B}'_i)$. La première ligne de $\tilde{C}(d_e)$ est donc

$$\begin{bmatrix} \tilde{y}_1(6) & \tilde{y}_1(7) & \tilde{y}_2(1) & \tilde{y}_2(2) & \tilde{y}_2(3) & \tilde{y}_2(4) & \tilde{y}_2(5) \end{bmatrix}.$$

Il tente ensuite une seconde hypothèse, $d_e = 1$. Il jette alors les $(n - d_e) = 6$ premiers bits de $(\tilde{B}'_i)$. La première ligne de $\tilde{C}(d_e)$ est donc

$$\begin{bmatrix} \tilde{y}_2(2) & \tilde{y}_2(3) & \tilde{y}_2(4) & \tilde{y}_2(5) & \tilde{y}_2(6) & \tilde{y}_2(7) & \tilde{y}_3(1) \end{bmatrix}.$$

La première ligne de $\tilde{C}(d_e)$ est un mot de code bruité si et seulement si $d_e = d$. Elle correspond dans ce cas, à $\tilde{y}_2$.

Nous pouvons alors en déduire l'expression de $\tilde{C}(d_e)$. Dans le cas où $d_e > d$,

$$\begin{aligned} \left[\tilde{C}(d_e) \right]_{ij} &= \tilde{y}_i(j + n - (d_e - d)), \quad \forall i = 1 \dots N, \quad j = 1 \dots (d_e - d), \\ \left[\tilde{C}(d_e) \right]_{ij} &= \tilde{y}_{i+1}(j - (d_e - d)), \quad \forall i = 1 \dots N, \quad j = (d_e - d) + 1 \dots n, \end{aligned}$$

sinon

$$\begin{aligned} \left[\tilde{C}(d_e) \right]_{ij} &= \tilde{y}_{i+1}(j + (d - d_e)), & \forall i = 1 \dots N, \quad j = 1 \dots n - (d - d_e), \\ \left[\tilde{C}(d_e) \right]_{ij} &= \tilde{y}_{i+2}(j - n + (d - d_e)), & \forall i = 1 \dots N, \quad j = n - (d - d_e) + 1 \dots n. \end{aligned}$$

De même, sans connaissance *a priori*, l'observateur doit aussi deviner la longueur n des mots de code. Il doit alors faire une hypothèse n_e pour l'estimer. Il « jette » les $(n_e - d_e)$ premiers bits de $(\tilde{B}'_i)$ et commence à remplir la matrice $\tilde{C}(n_e, d_e)$ à partir de $\tilde{B}'_{(n_e - d_e + 1)}$, en changeant de ligne tous les n_e bits.

Remarque 2.1 :

Les lignes de la matrice $\tilde{C}(n_e, d_e)$ sont des mots de codes bruités par le canal si et seulement si $(n_e, d_e) = (n, d)$, $\alpha \in \mathbb{N}$.

Soient les matrices $\tilde{C} = \tilde{C}(n, d)$ et $C = C(n, d)$ de taille $N \times n$ dont les N lignes sont les mots de code bruités et interceptés, respectivement les mots de code émis sur le canal binaire, *i.e.* $[\tilde{C}]_{ij} = \tilde{y}_i(j)$ et $[C]_{ij} = y_i(j)$. De plus, si nous définissons la *matrice d'erreur* E par $[E]_{ij} = e_i(j)$ alors $\tilde{C} = C \oplus E$. De même, à la matrice $\tilde{C}(n_e, d_e)$ correspondent $C(n_e, d_e)$ et $E(n_e, d_e)$ telles que $\tilde{C}(n_e, d_e) = C(n_e, d_e) \oplus E(n_e, d_e)$.

Une fois $\tilde{C}(n_e, d_e)$ construite, l'observateur la découpe en deux sous-matrices $\tilde{C}_1(n_e, d_e)$ et $\tilde{C}_2(n_e, d_e)$ de tailles respectives $n_e \times n_e$ et $(N - n_e) \times n_e$ vérifiant

$$\begin{aligned} \tilde{C}_1(n_e, d_e) &= C_1(n_e, d_e) \oplus E_1(n_e, d_e), \\ \tilde{C}_2(n_e, d_e) &= C_2(n_e, d_e) \oplus E_2(n_e, d_e), \end{aligned} \tag{2.2}$$

où $\tilde{C}_1(n_e, d_e)$ est la matrice carrée supérieure de $\tilde{C}(n_e, d_e)$. La proposition 2.1 nous permet alors d'exprimer le problème de reconstruction des codes en blocs linéaires à partir d'un critère algébrique très simple.

Proposition 2.1 *Dans les conditions précédentes, en supposant que parmi les n mots de code, $(n - k)$ forment une famille libre,*

$$\mathcal{H} = \text{Ker } (C_1(n, d)).$$

Preuve : immédiate d'après la définition et l'équation (2.1) page précédente. ■

L'hypothèse d'indépendance de $(n - k)$ mots de code y_i est raisonnable dans la mesure où les mots d'information envoyés sont en général issus d'une compression, d'un chiffrement puis de la sortie d'un brasseur. Le problème de reconstruction des codes en blocs linéaires est alors équivalent à déterminer $\text{Ker } C(n, d)$ en observant $\tilde{C}(n_e, d_e)$ avec n_e et d_e choisis par l'observateur.

Pour répondre à ce problème, nous avons besoin tout d'abord d'élargir la notion de rang pour des matrices un peu particulières, appelées *matrices bruitées*.

Définition 2.2 *Une matrice $\tilde{M}$ à coefficients dans $GF(2)$ de dimensions $p \times q$, telles que $p \gg q$ est dite bruitée si et seulement si elle peut s'écrire sous la forme*

$$\tilde{M} = M \oplus E,$$

avec $\text{rg}(\tilde{M}) \geq \text{rg}(M)$ et E matrice binaire, dont les coefficients sont tirés aléatoirement suivant une distribution uniforme de paramètre ε , le TEB du canal. M est appelée matrice non bruitée associée à $\tilde{M}$ et E matrice d'erreur associée à $\tilde{M}$.

Remarque 2.2 :

Nous rappelons que la *densité* $d(v)$ d'un vecteur binaire v de longueur l est donnée par

$$d(v) = \frac{w(v)}{l},$$

où $w(\cdot)$ est le poids de Hamming. Cette définition s'étend de façon naturelle à la densité d'une matrice M de taille $p \times q$, en posant

$$d(M) = \frac{1}{pq} \sum_{i=1}^q w(M_i),$$

où M_i est le $i^{\text{ème}}$ vecteur colonne. En d'autres termes, la densité d'une matrice est la moyenne de la densité de ses vecteurs colonnes.

Par construction, les matrices $\tilde{C}(n_e, d_e)$, $\tilde{C}_1(n_e, d_e)$ et $\tilde{C}_2(n_e, d_e)$ sont des matrices bruitées. Dans le cadre de la reconstruction des codes en blocs linéaires nous cherchons à obtenir une relation entre le rang de $\tilde{C}(n_e, d_e)$ et celui de $C(n_e, d_e)$ permettant d'estimer le second à partir du premier. Le théorème du rang nous donne une première relation : $\text{rg}(C(n_e, d_e)) = n - \dim(\text{Ker}(C(n_e, d_e)))$. Prenons un vecteur h de $\mathcal{H} = \text{Ker}(C(n, d))$, alors

$$\tilde{C}(n_e, d_e)h = C(n_e, d_e)h \oplus Eh = Eh.$$

Si l'erreur est faible, la matrice E est peu dense et possède des lignes constituées exclusivement de 0. Dans ce cas favorable, on peut donc espérer que Eh soit de poids faible. Notre stratégie est, assez naturellement, de rechercher des vecteurs h , tels que Eh soit de poids faible. En pratique, pour un TEB faible, les vecteurs du noyau de $C(n, d)$ sont envoyés dans une boule centrée en 0 et de faible rayon, $\mathcal{B}_{d_H}(0, \gamma N)$, $\gamma \in]0, 1/2[$, par $\tilde{C}(n, d)$ (d_H étant la distance de Hamming). Le paramètre γ est évalué au paragraphe 2.3.2. Le problème de reconstruction se réduit alors à un problème de recherche de mots de poids faible dans un code, ici le (N, n) -code en bloc linéaire $\mathcal{C}'$ engendré par les vecteurs colonnes de $\tilde{C}(n, d)$. En effet, soit v un mot de $\mathcal{C}'$ de poids faible, il existe une combinaison linéaire $(\alpha_i)_{i=1 \dots n}$, $\alpha_i \in GF(2)$, des vecteurs colonnes $(\tilde{C}^{(i)}(n, d))$ telle que

$$v = \sum_{i=1}^n \alpha_i \tilde{C}^{(i)}(n, d) = \tilde{C}(n, d)h,$$

où h est le vecteur de longueur n défini par $h = (\alpha_1, \dots, \alpha_n)$. Or, le (N, n) -code en bloc linéaire $\mathcal{C}'$ possède une distribution du poids des mots de code un peu particulière. En effet, lorsque le canal est sans erreur, on dénombre $2^{(n-k)}$ mots de poids 0 (ces mots de code ont exactement pour antécédents les vecteurs de $\mathcal{H}$) ainsi qu'un petit nombre de vecteurs de poids faible. Ceci s'explique par la distribution des poids des vecteurs du dual du code qui est liée à la distribution des poids des mots du code par la relation de MacWilliams [138]. La répartition des poids des vecteurs du dual est d'ailleurs à la base

des tests statistiques développés par M. Cluzeau pour discriminer les vecteurs du dual des autres. Les illustrations de sa thèse [51, ch. 2.4], mettent clairement en évidence les différentes répartitions des poids des vecteurs suivant qu'ils appartiennent au dual ou non. Lorsque l'erreur du canal augmente, mais reste faible, ces vecteurs ne sont plus de poids 0 mais de poids faible, comme l'illustre la figure 2.1 page 58. Les vecteurs de poids faible sont majoritairement issus du codage des vecteurs de $\mathcal{H}$. Pour s'en convaincre, démontrons tout d'abord le théorème suivant.

Théorème 2.1 *Soit W la variable aléatoire discrète à valeurs dans $[0, N]$ et définie sur l'ensemble des vecteurs binaires v de longueur n , par $W(v) = w(\tilde{C}(n, d)v)$. Alors W ne suit pas la même distribution de probabilité sur $\mathcal{H}$ et son complémentaire. Notons $P_p(v) = \frac{1+(1-2\varepsilon)^{w(v)}}{2}$.*

$$\begin{cases} \mathcal{Pr}(W(v) = x \mid v \in \mathcal{H}) &= \binom{N}{x} (1 - P_p(v))^x P_p^{(N-x)}(v), \\ \mathcal{Pr}(W(v) = x \mid v \notin \mathcal{H}) &= 2^{-N} \binom{N}{x}. \end{cases}$$

Preuve :

supposons $v \notin \text{Ker}(C(n, d))$. Le vecteur v n'est donc pas une relation de parité, ce qui implique $\forall i = 1 \dots N$, $\langle y_i, v \rangle = 1$ avec probabilité $\frac{1}{2}$. On en déduit $\langle \tilde{y}_i, v \rangle = \langle y_i, v \rangle \oplus \langle e_i, v \rangle = 1$ avec probabilité $\frac{1}{2}$. Le résultat des équations $\langle \tilde{y}_i, v \rangle$ est une variable aléatoire discrète binaire, définie sur l'ensemble des vecteurs binaires de longueur n , suivant une loi de probabilité uniforme de paramètre $\frac{1}{2}$. De plus, si on suppose que les (y_i) sont générés de façon indépendante, les $(\tilde{y}_i)$ le sont aussi ainsi que le résultat des $\langle \tilde{y}_i, v \rangle$. D'autre part, chacune de ces équations est une coordonnée du vecteur $\tilde{C}(n, d)v$. On peut alors en conclure que les coordonnées du vecteur $\tilde{C}(n, d)v$ sont N variables aléatoires binaires indépendantes suivant une loi uniforme de paramètre $\frac{1}{2}$. Enfin, il en résulte que $W(v)$ suit une distribution binomiale de paramètres $(N, \frac{1}{2})$ d'où

$$\mathcal{Pr}(W(v) = x \mid v \notin \mathcal{H}) = 2^{-N} \binom{N}{x}.$$

Supposons maintenant que $v \in \text{Ker}(C(n, d))$; alors $\langle \tilde{y}_i, v \rangle = \langle y_i, v \rangle \oplus \langle e_i, v \rangle = \langle e_i, v \rangle$. Évaluons alors la probabilité que $\langle e_i, v \rangle = 0$. Pour cela démontrons la proposition suivante.

Proposition 2.2 *Soit $v \in \text{Ker}(C(n, d))$, alors*

$$\mathcal{Pr}(\langle e_i, v \rangle = 0) = \frac{1 + (1 - 2\varepsilon)^{w(v)}}{2}.$$

Preuve :

$\langle e_i, v \rangle = 0$ chaque fois que la somme des $e_i(j)$ prise sur les j pour lesquels $v(j) = 1$ est paire, c'est-à-dire

$$\langle e_i, v \rangle = 0 \text{ si et seulement si } \sum_{j|v(j)=1} e_i(j) \equiv 0 \pmod{2}. \quad (2.3)$$

Or, par définition du canal, les $(e_i(j))$ sont des variables indépendantes qui suivent une loi uniforme de paramètre ε , on en déduit

$$\mathcal{P}r(< e_i, v > = 0) = \sum_{j=0}^{\lfloor w(v)/2 \rfloor} \binom{w(v)}{2j} \varepsilon^{2j} (1 - \varepsilon)^{w(v)-2j}.$$

Il suffit ensuite d'écrire

$$\begin{aligned} (1 - 2\varepsilon)^{w(v)} &= ((1 - \varepsilon) - \varepsilon)^{w(v)} = \sum_{i=0}^{w(v)} \binom{w(v)}{i} (-1)^i \varepsilon^i (1 - \varepsilon)^{w(v)-i}, \\ 1 &= ((1 - \varepsilon) + \varepsilon)^{w(v)} = \sum_{i=0}^{w(v)} \binom{w(v)}{i} \varepsilon^i (1 - \varepsilon)^{w(v)-i}, \\ \frac{1 + (1 - 2\varepsilon)^{w(v)}}{2} &= \sum_{i=0}^{\lfloor w(v)/2 \rfloor} \binom{w(v)}{2i} \varepsilon^{2i} (1 - \varepsilon)^{w(v)-2i}, \end{aligned}$$

pour terminer la preuve de la proposition 2.2 page précédente. ■

Cette proposition nous permet de clore la démonstration de la proposition 2.1 page précédente. Notons $P_p(v) = \frac{1+(1-2\varepsilon)^{w(v)}}{2}$. Comme précédemment, les $< \tilde{y}_i, v > = < e_i, v >$ sont des variables aléatoires binaires indépendantes qui suivent une loi uniforme de paramètre $P_p(v)$, il en est de même pour les coordonnées de $\tilde{C}(n, d)v$. Enfin, il en résulte que $W(v)$ suit une distribution binomiale de paramètres $(N, P_p(v))$, d'où

$$\mathcal{P}r(W(v) = x \mid v \in \mathcal{H}) = \binom{N}{x} (1 - P_p(v))^x P_p^{(N-x)}(v). \quad \blacksquare$$

En remarquant que $P_p(v) \in]1/2, 1[$ lorsque $\varepsilon \in]0, 1/2[$ nous pouvons alors démontrer la proposition suivante.

Proposition 2.3 Soient T un entier fixé et $\gamma \in]0, 1/2[$, alors

$$\lim_{N \rightarrow \infty} \mathcal{P}r(v \in \mathcal{H} \mid W(v) \leq \gamma T) = 1.$$

Preuve :

Pour démontrer cette proposition, nous allons chercher la limite de $\mathcal{P}r(v \notin \mathcal{H} \mid W(v) \leq \gamma T)$.

$$\begin{aligned} \mathcal{P}r(v \notin \mathcal{H} \mid W(v) \leq \gamma T) &= \mathcal{P}r(v \notin \mathcal{H}) \frac{\mathcal{P}r(W(v) \leq \gamma T \mid v \notin \mathcal{H})}{\mathcal{P}r(W(v) \leq \gamma T)}, \\ &= \mathcal{P}r(v \notin \mathcal{H}) \frac{1}{1 + \frac{\mathcal{P}r(W(v) \leq \gamma T \mid v \in \mathcal{H})}{\mathcal{P}r(W(v) \leq \gamma T \mid v \notin \mathcal{H})}}, \\ &= \frac{\mathcal{P}r(v \notin \mathcal{H})}{1 + Q(N)}, \end{aligned}$$

où

$$Q(N) = \frac{P_p(v)^N \sum_{x=0}^{\lfloor \gamma T \rfloor} \binom{N}{x} \left(\frac{1}{P_p(v)} - 1 \right)^x}{2^{-N} \sum_{x=0}^{\lfloor \gamma T \rfloor} \binom{N}{x}} \geq (2P_p(v))^N \left(\frac{1}{P_p(v)} - 1 \right)^{\lfloor \gamma T \rfloor},$$

d'après le théorème 2.1 page 55. En remarquant que $\alpha P_p(v) > 1$, il en résulte que $\lim_{N \rightarrow \infty} Q(N) = +\infty$ et donc

$$\lim_{N \rightarrow \infty} \Pr(v \notin \mathcal{H} \mid W(v) \leq \gamma T) = 0.$$

Ceci nous permet de conclure

$$\lim_{N \rightarrow \infty} \Pr(v \in \mathcal{H} \mid W(v) \leq \gamma T) = 1. \blacksquare$$

En faisant l'hypothèse raisonnable que N est de l'ordre de n^2 , on en déduit que si $\tilde{C}(n, d)h$ est de poids faible, alors, avec forte probabilité, $h \in \text{Ker}(C(n, d)) = \mathcal{H}$. Lorsque $n_e \neq n$, en faisant l'hypothèse que N est suffisamment grand (de l'ordre de n^2) et γ , suffisamment petit, d'après la répartition des vecteurs de poids faible que nous venons de mettre en évidence, la probabilité de trouver un vecteur de poids faible est proche de 0, en considérant $\mathcal{C}'$ comme un code aléatoire.

2.1.2 Algorithme de Gauss randomisé

Pour répondre au problème de reconstruction des codes en blocs linéaires, trois approches sont possibles :

- la recherche exhaustive,
- les algorithmes de recherche de mots de poids faible dans un code linéaire : algorithmes de Leon [129], de Stern [181], de Lee-Brickell [128] et enfin de Canteaut-Chabaud [43, 42],
- l'algorithme de Sicot-Houcke [173].

La première technique est de complexité exponentielle en $\mathcal{O}(2^n)$ et n'est pas réalisable en pratique pour les longueurs de codes usuels. Les algorithmes de recherche de mots de poids faible ont déjà été étudiés à maintes reprises et adaptés à des fins de reconstruction de codes correcteurs d'erreurs, notamment dans les thèses de A. Valembois [195] et de M. Cluzeau [51]. Les analyses des algorithmes classiques de recherche de mots de poids faible ont été réalisées sous l'angle de la recherche de mots de poids faible dans des codes en blocs linéaires aléatoires. Or, les codes engendrés par les matrices d'interception sont un peu particuliers : leur dimension et la répartition de leurs mots de poids faible évoluent en fonction de l'erreur et sont très différents des codes classiques. La figure 2.1 page suivante, illustre l'évolution de la répartition des poids en fonction de l'erreur pour une matrice d'interception initialement non bruitée de dimension 100×20 , de rang 14. Cette matrice est formée par des mots d'un code de longueur 20 et de dimension 14. Nous avons donc choisi d'adopter et d'analyser finement l'algorithme de Sicot-Houcke qui consiste en une randomisation de l'algorithme de Gauss pour les matrices bruitées. L'objectif est alors d'avoir des tirages indépendants pour nos matrices à étudier afin de découvrir plus vite des relations de poids faible ; en contre-partie nous payons la complexité d'un pivot de Gauss à chaque itération. Cette approche diffère des précédentes, en ce sens qu'elle tire parti de la résistance aux erreurs intrinsèque à l'algorithme de Gauss. En effet, le processus de Gauss n'est pas perturbé lorsque le nombre d'erreurs affectant des bits d'un mot de code impliqués dans une relation de parité est paire. D'autre part, quand celui-ci est impair, sous certaines conditions mises en évidence au paragraphe 2.3.2, les mots de poids faible apparaissent dans une base de $(\mathcal{C}')^\perp$.

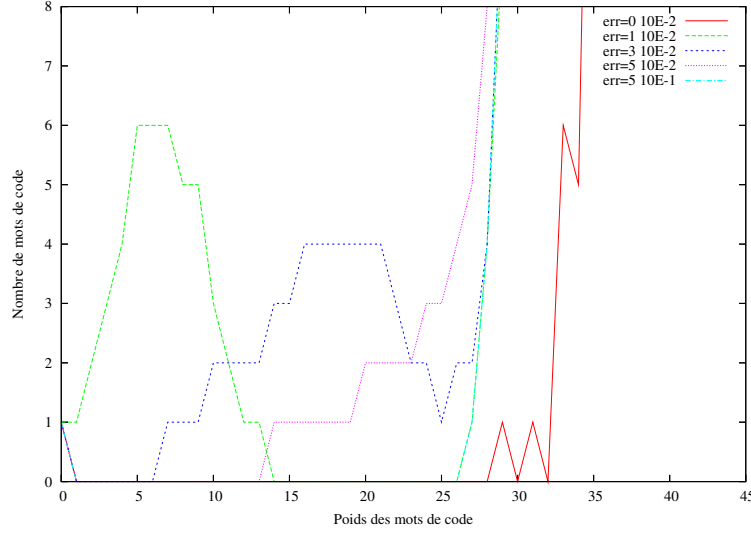


FIG. 2.1 – Répartition des mots de code dans un code de longueur 100

Le principe de l'algorithme de Gauss randomisé pour des matrices bruitées est le suivant. Celui-ci se décompose en deux étapes :

- une étape algébrique, consistant en l'application de l'algorithme de Gauss afin de trouver un ensemble de vecteurs candidats,
- une étape statistique de test d'hypothèses, afin de sélectionner parmi les vecteurs candidats ceux qui appartiennent à $\mathcal{H}$.

Nous disons que la première étape permet de *détecter* une relation de parité et la seconde de la *valider*. Comme son nom l'indique, cet algorithme est randomisé. Pour cela, nous générons l matrices $\tilde{C}^{(i)}(n_e, d_e)$, par un processus de Monte Carlo, en effectuant une permutation aléatoire des N lignes de la matrice $\tilde{C}(n_e, d_e)$. l est un paramètre de l'algorithme de Gauss randomisé. Cette étape ne change clairement pas les vecteurs du noyau de $C(n_e, d_e)$. Chacune de ces matrices peut être écrite sous la forme de l'équation (2.2) page 53,

$$\begin{aligned}\tilde{C}_1^{(i)}(n_e, d_e) &= C_1^{(i)}(n_e, d_e) \oplus E_1^{(i)}(n_e, d_e), \\ \tilde{C}_2^{(i)}(n_e, d_e) &= C_2^{(i)}(n_e, d_e) \oplus E_2^{(i)}(n_e, d_e).\end{aligned}\tag{2.4}$$

La première étape réside en l'application de l'algorithme de Gauss classique aux matrices $\tilde{C}_1^{(i)}(n_e, d_e)$. Chacune des $\tilde{C}_1^{(i)}(n_e, d_e)$ peut alors s'écrire sous la forme

$$A^{(i)} \cdot \tilde{C}_1^{(i)}(n_e, d_e) \cdot B^{(i)} = T^{(i)},$$

où $T^{(i)}$ est une matrice triangulaire inférieure, $A^{(i)}$ matrice inversible maintenant les échanges de lignes et $B^{(i)}$ maintenant les combinaisons linéaires des colonnes. L'application associée à $A^{(i)}$ conservant le poids de Hamming, nous obtenons l ensembles de vecteurs candidats $\mathcal{B}_i = \{B_j^{(i)}, j = 1 \dots n\}$, $i = 1 \dots l$. Si $\tilde{C}_1^{(i)}$ n'est pas de rang plein, $\mathcal{B}_i$ contient les vecteurs du noyau de $\tilde{C}_1^{(i)}(n_e, d_e)$.

Enfin, la seconde étape permet de sélectionner des vecteurs appartenant potentiellement au noyau de $C(n_e, d_e)$. Pour chaque vecteur v des $\mathcal{B}_i$, il nous faut discriminer entre deux hypothèses, H_1 , « $v \in \text{Ker}(C(n_e, d_e))$ » et H_2 , « $v \notin \text{Ker}(C(n_e, d_e))$ ». Pour ce

Algorithme de Gauss randomisé

Entrées : une matrice M de taille $p \times q$ avec $p > q$,

$\gamma \in [0, 1]$,

$l \in N^*$.

Sortie : (l, γ) -RKer (M) .

```

1   $K \leftarrow \{0\}$ 
2  Pour  $i$  de 1 à  $l$ 
3      Générer  $M^{(i)}$  en effectuant une permutation aléatoire des lignes de  $M$ 
4      Appliquer l'algorithme de Gauss à  $M_1^{(i)}$  et évaluer  $\mathcal{B}_i$ 
5      Pour tout  $v$  de  $\mathcal{B}_i$ 
6          Si  $d(Mv) \leq \gamma$  alors  $K \leftarrow K \cup \{v\}$ 
7          Fin Si
8      Fin Pour
9  Fin Pour
10 Renvoyer  $Vect(K)$ 

```

FIG. 2.2 – Algorithme de Gauss randomisé pour les matrices bruitées

faire, nous notons D la variable aléatoire discrète à valeurs dans $[0, 1]$ et définie sur l'ensemble des vecteurs binaires v de longueur n par $D(v) = d(\tilde{C}(n_e, d_e)v)$. La distribution un peu particulière du (N, n) -code en blocs linéaire $\mathcal{C}'$, mise en évidence au paragraphe précédent et le théorème 2.1, impliquent que D ne suit pas la même distribution de probabilité sur $\mathcal{H}$ et son complémentaire. La règle de décision associée au test d'hypothèses est la suivante. Observant v , nous décidons H_1 si $D(v) \leq \gamma$ et H_2 sinon. γ est le second paramètre de l'algorithme de Gauss randomisé. Le pseudo-code de l'algorithme est donné par la figure 2.2. Nous appelons *noyau randomisé* de paramètres l et γ , de $\tilde{C}(n_e, d_e)$, la sortie de l'algorithme de Gauss randomisé appliqué à $\tilde{C}(n_e, d_e)$. Celui-ci est noté

$$(l, \gamma)\text{-RKer } (\tilde{C}(n_e, d_e)).$$

La notion de noyau randomisé étend la notion de noyau aux matrices bruitées. En effet, soit M une matrice non bruitée de taille $p \times q$, avec $p \gg q$ et $h \in \text{Ker}(M)$, alors $h \in \mathcal{B}_i$, $\forall i = 1 \dots l$, ensembles de vecteurs candidats; h est donc détecté par l'algorithme de Gauss randomisé. De plus $d(h) = 0 \leq \gamma$; h est donc validé par l'algorithme et appartient à $Vect(K)$. On en déduit

$$\text{Ker}(M) \subset (l, \gamma)\text{-RKer}(M).$$

Soit $v \in (l, \gamma)\text{-RKer}(M)$, alors $D(v) \leq \gamma$. En utilisant la proposition 2.3 page 56, on montre aisément que

$$\lim_{p \rightarrow \infty} \Pr(v \in \text{Ker}(M) \mid D(v) \leq \gamma) = 1. \quad (2.5)$$

En prenant p suffisamment grand, de l'ordre de q^2 , $v \in \text{Ker}(M)$ avec forte probabilité. Avec cette même probabilité,

$$\text{Ker } (M) = (l, \gamma)\text{-RKer } (M). \quad (2.6)$$

Dans le même esprit, nous pouvons introduire le *rang randomisé*. Nous appelons *rang randomisé* de paramètres l et γ , de la matrice bruitée $\tilde{M}$ de taille $p \times q$, l'entier noté $(l, \gamma)\text{-rrg } (\tilde{M})$ défini par

$$(l, \gamma)\text{-rrg } (\tilde{M}) = p - \dim \left((l, \gamma)\text{-RKer } (\tilde{M}) \right).$$

La notion de rang randomisé étend la notion de rang aux matrices bruitées. Soit M matrice non bruitée de taille $p \times q$, avec $p \gg q$, d'après la relation (2.6),

$$(l, \gamma)\text{-rrg } (\tilde{M}) = p - \dim \left((l, \gamma)\text{-RKer } (\tilde{M}) \right) = p - \dim (\text{Ker } (M)) = \text{rg } (M),$$

avec forte probabilité.

Il nous faut maintenant noter quelques difficultés intuitives qu'il convient de prendre en compte lors du choix des paramètres.

Soit $\tilde{C}(n_e, d_e) = C(n_e, d_e) \oplus E(n_e, d_e)$ la matrice bruitée construite avec les bits interceptés, alors

- la probabilité que les vecteurs de $\text{Ker } (C(n_e, d_e))$ apparaissent dans $\mathcal{B}_i$ est d'autant plus faible que $E(n_e, d_e)$ est dense,
- la probabilité qu'un vecteur de $\text{Ker } (C(n_e, d_e))$ soit détecté mais non validé est d'autant plus forte que $E(n_e, d_e)$ est dense,
- il existe avec faible probabilité (cf. relation (2.5) page précédente) des vecteurs appartenant à $(l, \gamma)\text{-RKer } (\tilde{C}(n_e, d_e))$ mais pas à $\text{Ker } (C(n_e, d_e))$.

Ces difficultés sont explicitées et quantifiées en termes de probabilité au paragraphe 2.3.2 page 69. D'autre part, nous pouvons noter les cas favorables :

- la probabilité de détecter un vecteur de $\text{Ker } (C(n_e, d_e))$ est d'autant plus forte que le nombre d'itérations l est grand,
- la validation est d'autant plus fiable que N est grand.

Pour résoudre le problème de reconstruction des codes en blocs linéaires, il nous faut donc évaluer le noyau randomisé de $\tilde{C}(n, d)$. Pour ce faire, nous avons besoin de deviner les paramètres $n_e = n$ et $d_e = d$. Nous présentons maintenant les algorithmes de reconstruction pour retrouver ces paramètres à partir de l'algorithme de Gauss randomisé. Rappelons que la densité de la matrice d'erreur associée au canal binaire est exactement le TEB du canal. D'après les remarques précédentes, on s'attend à ce que la reconstruction soit « plus facile » quand on intercepte beaucoup de bits (N grand) et quand l'erreur du canal est faible (matrice d'erreur peu dense).

2.2 Algorithmes de reconstruction

2.2.1 Canal sans erreur

Étudions tout d'abord le cas le plus simple d'un canal sans erreur. Cette hypothèse correspond à la situation dans laquelle l'observateur veut retrouver les paramètres d'un

code en bloc linéaire qui sont stockés dans un équipement qu'il possède. L'équipement peut alors être vu comme une boîte noire qui peut être stimulée et qui va en sortie émettre un flux de bits codé et sans erreur.

Dans ce cas très précis, nous avons $\tilde{C}(n_e, d_e) = C(n_e, d_e)$ et la matrice interceptée est non bruitée. L'algorithme de Gauss randomisé renvoie dans ce cas le noyau d'une matrice non bruitée avec une probabilité proche de 1 (cf. relation (2.6) page précédente). Pour cela, il effectue l pivots de Gauss sur une matrice de taille n ; nous utilisons donc préférentiellement le noyau et le rang classiques de $C(n_e, d_e)$ à son noyau et rang randomisés. De plus, d'après la proposition 2.1 page 53, l'évaluation de $\text{Ker}(C_1(n, d))$ nous renvoie directement $\mathcal{H}$. Nous utilisons donc l'algorithme de Gauss classique plutôt que celui de Gauss randomisé. Afin de détecter les bons paramètres n et d , il est utile de remarquer que chercher une relation de parité est équivalent à rechercher des colonnes liées dans la matrice $C(n, d)$. En effet, si h est une relation de parité, alors

$$\langle C_i(n, d), h \rangle = \sum_{j=1}^n h(j) \cdot y_i(j) = \sum_{j|h(j)=1} y_i(j) = 0 \quad \forall i = 1 \dots N,$$

où $C_i(n, d)$ est la $i^{\text{ème}}$ ligne de $C(n, d)$. En d'autres termes, si h est une relation de parité, la somme des colonnes d'indice j pour lequel le bit $h(j)$ est à 1 vaut 0 ; ces colonnes sont liées.

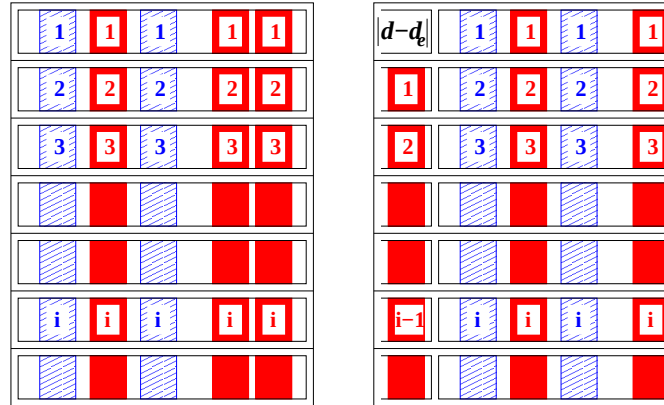


FIG. 2.3 – Représentation de $C(n, d)$ et $C(n, d_e)$ avec deux relations de parité

Dans l'hypothèse où nous avons deviné la bonne valeur de n_e et estimé le facteur de désynchronisation d , l'évaluation du noyau de $C(n, d)$ se fait sans problème à l'aide de l'algorithme de Gauss. Le rang $\text{rg}(C(n, d))$ est donc égal à $n - (n - k) = k$ d'après le théorème du rang. Supposons maintenant que $n_e = n$, $d_e \neq d$ et h relation de parité. L'introduction d'une désynchronisation $\delta = |d - d_e|$ peut avoir deux effets distincts sur h comme l'illustre la figure 2.3 :

- le dernier bit à 1 de h est d'indice supérieur à $(n - \delta)$ et dans ce cas, les colonnes correspondantes à h d'indice augmenté de δ ne sont plus liées ; $(h \gg \delta)^2 \notin \text{Ker}(C(n, d_e))$,

² $h \gg \delta$ signifie que h est décalé de δ bits vers la droite.

- le dernier bit à 1 de h est d'indice inférieur ou égal à $(n - \delta)$ ce qui implique que les colonnes correspondantes à h d'indice augmenté de δ sont liées; ($h \gg \delta$) $\in \text{Ker}(C(n, d_e))$.

On en déduit qu'avec une probabilité proche de 1,

$$\text{rg}(C(n, d)) \leq \text{rg}(C(n, d')) \quad \forall d'. \quad (2.7)$$

Nous négligeons ainsi la probabilité que le rang diminue lors du décalage, c'est-à-dire la probabilité qu'au moins deux vecteurs indépendants du noyau n'appartenant pas à $\mathcal{H}$ apparaissent « spontanément ». Cette approximation se vérifie en pratique. La relation de parité hachurée est conservée par l'introduction d'une désynchronisation d tandis que la pleine est « cassée ». En effet, la somme des bits hachurés d'une même ligne vaut 0 pour chaque ligne ce qui n'est plus le cas pour les pleins car un bit plein par ligne se retrouve sur la suivante.

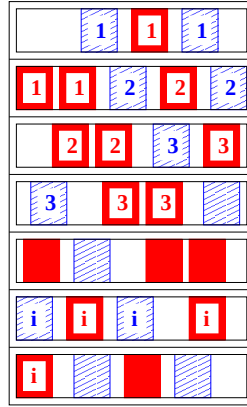


FIG. 2.4 – Représentation de $C(n_e, d_e)$ avec deux relations de parité

Dans l'hypothèse où nous avons choisi $n_e \neq \beta n$, $\beta \in \mathbb{N}^*$, aucune relation de parité n'est conservée et nous considérons que $C(n_e, d_e)$ se comporte comme une matrice binaire aléatoire, comme l'illustre la figure 2.4. Cette hypothèse forte se vérifie dans la pratique. Soit R une variable aléatoire discrète à valeurs dans $\mathbb{N}^+$, définie sur l'ensemble des matrices binaires par $R(M) = \text{rg}(M)$. Un résultat classique d'algèbre [132, p. 455], donne la probabilité qu'une matrice binaire aléatoire de dimensions $p \times q$ soit de rang plein

$$\prod_{i=0}^{q-1} (1 - 2^{i-p}). \quad (2.8)$$

Enfin, la majoration donnée en annexe A, nous permet de conclure que

$$\Pr(R(C(n_e, d_e)) < n_e) \leq 2^{-N} (2^{n_e} - 1) \leq 2^{-N+n_e}. \quad (2.9)$$

On en déduit qu'avec forte probabilité, $R(C(n_e, d_e)) = n_e$. Pour pouvoir comparer $C(n_e, d_e)$ et $C(n, d)$ deux matrices qui ont des tailles différentes, il nous faut introduire le *rang normalisé*.

Définition 2.3 Soit M une matrice de dimension $p \times q$, on appelle rang normalisé de M le réel de $[0, 1]$ défini par

$$\frac{\text{rg}(M)}{\min(p, q)}.$$

Le rang normalisé est une variable aléatoire discrète, à valeurs dans $[0, 1]$ et définie sur l'ensemble des matrices binaires. De plus,

$$\mathcal{P}r\left(\frac{\text{rg}(C(n_e, d_e))}{n_e} = 1\right) = \mathcal{P}r(R(C(n_e, d_e)) = n_e) \geq 1 - 2^{-N+n_e}.$$

La relation (2.7) page précédente nous permet d'en déduire qu'avec forte probabilité,

$$\frac{\text{rg}(C(n, d))}{n} \leq \frac{\text{rg}(C(n, d_e))}{n} \leq \frac{\text{rg}(C(n_e, d_e))}{n_e} = 1 \quad \forall d_e.$$

Cette relation nous donne un premier critère pour deviner les bons paramètres pour n_e et d_e . Ce critère est appelé *critère du rang* et s'exprime par

$$(n, d) = \underset{n_e, d_e}{\text{Argmin}} \left(\frac{\text{rg}(C(n_e, d_e))}{n_e} \right).$$

Il en résulte alors un algorithme de reconstruction des codes en blocs linéaires dans le cas d'un canal sans erreur. La première étape consiste à faire des hypothèses croissantes sur n_e puis tester toutes les hypothèses sur d pour d_e variant de 0 à $(n_e - 1)$. On remplit la matrice $C(n_e, d_e)$ en conséquence. Dans un deuxième temps, on évalue le rang normalisé de $C(n_e, d_e)$ à l'aide de l'algorithme de Gauss classique. Si celui-ci diminue alors n_e et d_e sont considérés comme plus « proches » de n et d . Enfin l'algorithme renvoie le paramètre n_e et le rendement r pour lesquels le rang normalisé de $C(n_e, d_e)$ est minimal ainsi que le noyau de $C(n_e, d_e)$ comme estimation du code dual. Le pseudo-code de l'algorithme est donné par la figure 2.5 page suivante.

2.2.2 Canal avec erreur

Les raisonnements qui ont été faits dans le paragraphe précédent peuvent se transposer aisément en remplaçant les notions traditionnelles de rang, noyau et algorithme de Gauss par leur version randomisée.

Dans l'hypothèse où nous avons deviné la bonne valeur de n_e et estimé le facteur de désynchronisation d , l'algorithme de Gauss randomisé appliqué à $\tilde{C}(n_e, d_e)$, nous renvoie avec une certaine probabilité de succès le noyau de $C(n, d)$. Le rang randomisé (l, γ) -rrg $(\tilde{C}(n, d))$ est donc égal à $n - (n - k) = k$ avec cette même probabilité de succès. Cette probabilité de succès est évaluée lors de l'analyse de l'algorithme au paragraphe 2.3.2. Supposons maintenant que $n_e = n$, $d_e \neq d$ et soit h une relation de parité. L'introduction d'une désynchronisation $\delta = |d - d_e|$ peut avoir deux effets distincts sur h :

- le dernier bit à 1 de h est d'indice supérieur à $(n - \delta)$ et dans ce cas, les colonnes correspondantes à h d'indice augmenté de δ ne sont plus liées ; $(h \gg \delta) \notin \text{Ker}(C(n, d_e))$ et $\tilde{C}(n, d_e)(h \gg \delta) = C(n, d_e)(h \gg \delta) + E(h \gg \delta)$ n'est plus nécessairement de poids faible,

**Algorithme de reconstruction des codes en blocs linéaires
pour un canal sans erreur**

Entrées : (y_i) les bits interceptés,
 $n_{max} \geq 2$ paramètre d'arrêt.

Sorties : $\mathcal{H}$ le code dual de $\mathcal{C}$ qui a généré les (y_i) ou $\emptyset$,
 r le rendement du code ou 0,
 n la dimension du code ou 0.

```

1   $\mathcal{H} \leftarrow \emptyset$ 
2   $n \leftarrow 0$ 
3   $r \leftarrow 1$ 
4  Pour  $n_e$  de 2 à  $n_{max}$ 
5    Pour  $d_e$  de 0 à  $(n_e - 1)$ 
6      construire  $C(n_e, d_e)$  à partir des  $(y_i)$ 
7      Si  $\text{rg}(C(n_e, d_e))/n_e < r$  alors
8         $r \leftarrow \text{rg}(C(n_e, d_e))/n_e$ 
9         $n \leftarrow n_e$ 
10        $\mathcal{H} \leftarrow \text{Ker}(C(n_e, d_e))$ 
11     Fin Si
12   Fin Pour
13 Fin Pour
14 Si  $n = 0$  alors renvoyer  $(\emptyset, 0, 0)$ 
15 Sinon renvoyer  $(\mathcal{H}, r, n)$ 

```

FIG. 2.5 – Algorithme de reconstruction des codes en blocs linéaires dans un canal sans erreur

- le dernier bit à 1 de h est d'indice inférieur ou égal à $(n - \delta)$ ce qui implique que la somme des colonnes de $C(n, d_e)$ correspondantes à h d'indice augmenté de δ vaut 0 ; $\tilde{C}(n, d_e)(h \gg \delta)$ est donc de poids faible avec forte probabilité (cf. théorème 2.1).

On en déduit qu'avec forte probabilité,

$$(l, \gamma)\text{-rrg}(C(n, d)) \leq (l, \gamma)\text{-rrg}(C(n, d_e)) \quad \forall d_e. \quad (2.10)$$

Nous négligeons ainsi la probabilité que le rang randomisé de $C(n_e, d_e)$ diminue lors du décalage que de nouveaux vecteurs indépendants de poids faible apparaissent dans le code engendré par les colonnes de $\tilde{C}(n_e, d_e)$. Comme pour le cas d'un canal sans erreur, cette approximation se vérifie en pratique.

Dans l'hypothèse où nous avons choisi $n_e \neq \beta n$, $\beta \in \mathbb{N}^*$, aucune relation de parité n'est conservée et nous considérons que $C(n_e, d_e)$ se comporte comme une matrice binaire aléatoire, ainsi que $\tilde{C}(n_e, d_e)$, par définition. En faisant l'hypothèse que N est suffisamment

grand (de l'ordre de n^2) et γ suffisamment petit, d'après la répartition des vecteurs de poids faible mise en évidence au paragraphe 2.1.1, la probabilité de trouver un vecteur de poids faible est proche de 0, en considérant le code engendré par les colonnes de $\tilde{C}(n_e, d_e)$ comme un code aléatoire. Ceci implique que lors de l'étape statistique de Gauss randomisé, seul le vecteur nul sera validé et l'algorithme retournera $\{0\}$ avec forte probabilité. Par définition, (l, γ) -rrg $(\tilde{C}(n_e, d_e))$ sera égal à n_e avec cette même probabilité. Pour pouvoir comparer $\tilde{C}(n_e, d_e)$ et $\tilde{C}(n, d)$ deux matrices qui ont des tailles différentes, il nous faut introduire la version randomisée du rang normalisé. Soit $\tilde{M}$ une matrice bruitée de dimension $p \times q$, on appelle *rang randomisé normalisé* de paramètres (l, γ) , de $\tilde{M}$, le réel de $[0, 1]$ défini par

$$\frac{(l, \gamma)\text{-rrg}(\tilde{M})}{\min(p, q)}.$$

La relation (2.10) page précédente nous permet d'en déduire qu'avec forte probabilité,

$$\frac{(l, \gamma)\text{-rrg}(\tilde{C}(n, d))}{n} \leq \frac{(l, \gamma)\text{-rrg}(\tilde{C}(n, d_e))}{n} \leq \frac{(l, \gamma)\text{-rrg}(\tilde{C}(n_e, d_e))}{n_e} = 1 \quad \forall d_e.$$

Comme pour le cas du canal sans erreur, nous obtenons un critère du rang dans sa version randomisée.

$$(n, d) = \underset{n_e, d_e}{\text{Argmin}} \left(\frac{(l, \gamma)\text{-rrg}(\tilde{C}(n_e, d_e))}{n_e} \right).$$

Il en résulte alors un algorithme de reconstruction des codes en blocs linéaires dans le cas d'un canal avec erreurs, qui n'est ni plus ni moins que la version randomisée de l'algorithme de reconstruction dans un canal sans erreur. La première étape consiste à faire des hypothèses croissantes sur n_e puis tester toutes les hypothèses sur d pour d_e variant de 0 à $(n_e - 1)$. On remplit la matrice $\tilde{C}(n_e, d_e)$ en conséquence. Dans un deuxième temps, on évalue le rang randomisé normalisé de $\tilde{C}(n_e, d_e)$. Si celui diminue alors n_e et d_e sont considérés comme plus « proches » de n et d . Enfin l'algorithme renvoie le paramètre n_e et le rendement r pour lesquels le rang randomisé normalisé de $\tilde{C}(n_e, d_e)$ est minimal, ainsi que le noyau randomisé de $\tilde{C}(n_e, d_e)$ comme estimation du code dual. Le pseudo-code de l'algorithme est donné en figure 2.6 page suivante.

2.3 Analyse des algorithmes

Dans ce paragraphe, nous présentons une analyse des algorithmes détaillés dans les paragraphes précédents. Nous nous intéressons dans un premier temps, au canal sans erreur, puis dans un deuxième temps nous abordons le cas d'un canal bruité. Pour chacune de ces analyses, deux cas distincts sont traités. Le premier cas considère $n_e \neq \beta n$ et correspond à la période pendant laquelle l'algorithme va « deviner » la bonne valeur de n_e . Le second considère $n_e = n$ et correspond à la période durant laquelle l'algorithme va tout d'abord se synchroniser, puis estimer le noyau de $C(n, d)$. De plus, les analyses effectuées s'appuient sur une hypothèse forte selon laquelle $C(n_e, d_e)$ et $\tilde{C}(n_e, d_e)$ (cf. paragraphe 2.2.2) se comportent comme des matrices binaires aléatoires. Cette hypothèse est néanmoins vérifiée en pratique.

Avant toute analyse, il convient de rappeler les paramètres qui sont pris en compte et la manière dont ils varient.

**Algorithme de reconstruction des codes en blocs linéaire
pour un canal avec erreur**

Entrées : $(\tilde{y}_i)$ les bits interceptés,
 n_{max} paramètre d'arrêt,
 $\gamma \in [0, 1]$ et $l \in \mathbb{N}^*$ paramètres de Gauss randomisé.

Sorties : $\mathcal{H}$ le code dual de $\mathcal{C}$ qui a généré les (y_i) ou $\emptyset$,
 r le rendement du code ou 0,
 n la dimension du code ou 0.

```

1   $\mathcal{H} \leftarrow \emptyset$ 
2   $n \leftarrow 0$ 
3   $r \leftarrow 1$ 
4  Pour  $n_e$  de 2 à  $n_{max}$ 
5    Pour  $d_e$  de 0 à  $(n_e - 1)$ 
6      construire  $\tilde{C}(n_e, d_e)$  à partir des  $(\tilde{y}_i)$ 
7      Si  $(l, \gamma)$ -rrg  $(\tilde{C}(n_e, d_e))/n_e < r$  alors
8         $r \leftarrow (l, \gamma)$ -rrg  $(\tilde{C}(n_e, d_e))/n_e$ 
9         $n \leftarrow n_e$ 
10        $\mathcal{H} \leftarrow (l, \gamma)$ -RKer  $(\tilde{C}(n_e, d_e))$ 
11     Fin Si
12   Fin Pour
13 Fin Pour
14 Si  $n = 0$  alors renvoyer  $(\emptyset, 0, 0)$ 
15 Sinon renvoyer  $(\mathcal{H}, r, n)$ 
```

FIG. 2.6 – Algorithme de reconstruction des codes en blocs linéaires dans un canal avec erreur

- Nous reconstruisons un code en bloc linéaire de longueur n et de dimension k . Ces paramètres sont fixés par le système de communication que nous tentons de remonter. La longueur n du code est le paramètre d'évaluation de la complexité.
- L'observateur intercepte un signal dans un canal binaire symétrique de paramètre ε . Nous évaluons la probabilité de succès des algorithmes en fonction de celui-ci. En pratique, ε varie de 0 à 10^{-2} .
- Lors de l'observation, l'adversaire introduit un décalage d qu'il ne maîtrise pas. Ce paramètre est fixé dès le début de l'interception et ne varie pas. L'observateur doit alors faire une hypothèse (n_e, d_e) sur la valeur de (n, d) , avec $0 \leq n_e \leq n$ et $0 \leq d_e \leq n_e - 1$.
- Pour chaque hypothèse (n_e, d_e) , il construit alors des matrices d'interception de taille $N \times n$, où N est de l'ordre de n^2 . Il doit donc intercepter $\mathcal{O}(n^3)$ bits. Pour une meilleure lisibilité, nous faisons l'hypothèse, que toutes les matrices d'interception possèdent N lignes, quelle que soit la valeur de n_e .

- Enfin, l'algorithme de Gauss randomisé prend en entrée le couple de paramètres (l, γ) . Le paramètre $l \in \mathbb{N}^*$ est le nombre de tirages de Monte Carlo effectués par l'algorithme et $\gamma \in]0, 1/2[$ est le seuil de détection de l'étape statistique de test d'hypothèses (cf. paragraphe 2.1.2 page 59).

Dans le cadre de l'analyse de l'algorithme dans le cas sans erreur, nous étudions tout d'abord la probabilité que l'algorithme renvoie un vecteur h de $\mathcal{H}$, c'est-à-dire la probabilité que l'algorithme détecte et valide h . Nous évaluons aussi la probabilité de fausse alarme, c'est-à-dire la probabilité que l'algorithme détecte et valide un vecteur $v \notin \mathcal{H}$. Le raisonnement est effectué sur une unique itération de l'algorithme, *i.e.* $l = 1$. Le calcul de ces probabilités détermine ensuite le choix de l pour des conditions opérationnelles de reconstruction.

Dans ce contexte, nous abordons la correction de l'algorithme sous l'angle probabiliste. En effet, bien que l'algorithme de reconstruction soit déterministe, sa sortie est correcte avec une certaine probabilité. Nous étudions alors la probabilité P_{succ} que la sortie de l'algorithme de reconstruction dans le cas sans erreur soit correcte.

L'algorithme de reconstruction dans le cas de canaux bruités est quant à lui probabiliste. Pour chacun des cas $n_e \neq \beta n$ et $n_e = \beta n$, nous évaluons les probabilités de fausse alarme et de détection en raisonnant sur une unique itération de l'algorithme de Gauss randomisé. Pour étudier la probabilité de fausse alarme dans le cas $n_e \neq \beta n$, l'analyse est partitionnée en deux configurations distinctes. Dans la première configuration, $\tilde{C}(n_e, d_e)$ n'est pas de rang plein et dans la seconde, $\tilde{C}(n_e, d_e)$ est de rang plein mais un vecteur candidat est validé. De même, pour étudier la probabilité de détection dans le cas $n_e = \beta n$, nous traitons deux configurations distinctes. Dans la première configuration, la relation de parité détectée appartient à $\text{Ker}(\tilde{C}(n_e, d_e))$, dans la seconde, elle n'appartient à $\text{Ker}(\tilde{C}(n_e, d_e))$ mais est candidate et validée par l'algorithme. Finalement, la probabilité de détection nous permet d'évaluer le nombre d'itérations nécessaires à la reconstruction et donc la complexité de l'algorithme.

2.3.1 Canal sans erreur

Nous montrons dans ce paragraphe que l'algorithme déterministe de reconstruction des codes en blocs linéaires pour des canaux sans bruit retourne le dual de $\mathcal{C}$ avec une probabilité de succès $\mathcal{P}_{succ}$ telle que

$$\mathcal{P}_{succ} \geq 1 - n^2 2^{-N+n}.$$

De plus, sa complexité est en

$$\mathcal{O}(n^5).$$

Cas $n_e \neq \beta n$

Considérons tout d'abord le cas où $n_e \neq \beta n$, $\beta \in \mathbb{N}^*$. Nous évaluons tout d'abord la probabilité de fausse alarme $\mathcal{P}_{fa}^{(0)}(N, n_e)$ de l'algorithme de reconstruction, c'est-à-dire la probabilité que celui-ci renvoie un vecteur $v \notin \mathcal{H}$. Celle-ci vérifie la proposition suivante.

Proposition 2.4 Soit $n_e \neq \beta n$,

$$\mathcal{P}_{fa}^{(0)}(N, n_e) \leq 2^{-N+n_e}.$$

Preuve :

L'algorithme de reconstruction renvoie un vecteur $v \notin \mathcal{H}$ lorsque

$$\frac{\text{rg}(C(n_e, d_e))}{n_e} \leq \frac{\text{rg}(C(n, d))}{n} = 1 - \frac{k}{n}.$$

La probabilité de fausse alarme est par définition

$$\mathcal{P}_{fa}^{(0)}(N, n_e) = \Pr \left(R(C(n_e, d_e)) \leq n_e \left(1 - \frac{k}{n} \right) \right).$$

On en déduit

$$\mathcal{P}_{fa}^{(0)}(N, n_e) \leq \Pr(R(C(n_e, d_e)) < n_e) \leq 2^{-N+n_e},$$

d'après la relation (2.8) page 62. ■

Pour conclure le cas $n_e \neq \beta n$, nous négligeons la probabilité de détecter et valider un vecteur de $\text{Ker}(C(n, d))$, les relations de parité n'étant pas conservées comme l'illustre la figure 2.4 page 62.

$$\mathcal{P}_{det}^{(0)}(N, n_e) \text{ est négligeable.}$$

Cas $n_e = \beta n$

Dans le cas $d_e \neq d$, nous pouvons tenir exactement le même raisonnement que pour la proposition 2.4 en considérant la sous-matrice constituée de $C(\beta n, d_e)$ privée des colonnes associées aux relations de parité valides (cf. figure 2.4 page 62) ; la borne reste donc valable. Dans le cas $d_e = d$, la probabilité de fausse alarme vaut 0 et la probabilité de détection vaut 1.

Analyse de complexité

Avec une faible probabilité $\mathcal{P}_{echec}$, l'algorithme ne retourne pas les bonnes relations de parité. Nous pouvons néanmoins borner cette probabilité.

Proposition 2.5 La probabilité d'échec $\mathcal{P}_{echec}$ de l'algorithme de reconstruction dans le cas d'un canal binaire sans erreur est bornée par

$$\mathcal{P}_{echec} \leq n^2 2^{-N+n}.$$

Preuve :

l'algorithme de reconstruction échoue lorsque en parcourant les n_e et d_e ,

$$\frac{\text{rg}(C(n_e, d_e))}{n_e} \leq \frac{\text{rg}(C(n, d))}{n} = 1 - \frac{k}{n}.$$

La probabilité d'un tel événement a été évaluée dans la proposition 2.4 page ci-contre. On en déduit la probabilité de succès de l'algorithme $\mathcal{P}_{succ}$,

$$\mathcal{P}_{succ} = \prod_{n_e=2}^n \left(1 - \mathcal{P}_{fa}^{(0)}\right)^{n_e}.$$

En utilisant la borne de la proposition 2.4 page précédente,

$$\begin{aligned} \mathcal{P}_{succ} &\geq \prod_{n_e=2}^n \left(1 - \mathcal{P}_{fa}^{(0)}\right)^n, \\ &\geq \left(1 - \mathcal{P}_{fa}^{(0)}\right)^{n^2}, \\ &\geq 1 - n^2 \mathcal{P}_{fa}^{(0)}. \end{aligned}$$

Finalement, on en déduit,

$$\mathcal{P}_{succ} \geq 1 - n^2 2^{-N+n}. \blacksquare$$

La proposition 2.5 page ci-contre, met en évidence l'importance du nombre de mots interceptés. En effet, la probabilité de succès tend exponentiellement vers 1 quand N augmente.

$$\lim_{N \rightarrow \infty} \mathcal{P}_{succ} = 1.$$

Le paramètre n est estimé *en aveugle*, il faut donc tester tous les cas de 2 à n . Pour chaque estimation de n_e de n , toutes les valeurs de désynchronisation d_e sont testées. Enfin, pour chaque test, il faut calculer $\text{rg}(C(n_e, d_e)) = \text{rg}(C_1(n_e, d_e))$ avec forte probabilité (d'après la proposition 2.1 page 53) à l'aide de l'algorithme de Gauss en $\mathcal{O}(n_e^3)$. Il en résulte que la complexité en temps de l'algorithme de reconstruction des codes en blocs linéaires est donnée par

$$\mathcal{O}\left(\sum_{n_e=2}^n n_e^3\right) = \mathcal{O}(n^5).$$

2.3.2 Canal avec erreur

Dans ce paragraphe, nous calculons les probabilités de détection $\mathcal{P}_{det}$ et de fausse alarme $\mathcal{P}_{fa}$ pour l'algorithme probabiliste de reconstruction des codes en blocs linéaires dans des canaux bruités. La probabilité de fausse alarme est majorée par

$$\mathcal{P}_{fa} \leq 2^{-N} + \frac{n^2}{2} e^{-\frac{N}{2}},$$

la probabilité de détection est minorée par

$$\mathcal{P}_{det} \geq P_p(h)^{\beta n} \sum_{i=0}^{\lfloor \gamma N \rfloor} \binom{N}{i} (1 - P_p(h))^i P_p(h)^{(N-i)},$$

et la complexité de l'algorithme est en

$$\mathcal{O}\left(\frac{n^5}{\mathcal{P}_{det}}\right).$$

Cas $n_e \neq \beta n$

Considérons tout d'abord le cas où $n_e \neq \beta n$, $\beta \in \mathbb{N}^*$. Nous évaluons maintenant la probabilité de fausse alarme de l'algorithme de reconstruction, c'est-à-dire la probabilité de détecter et valider une relation de parité qui n'appartient pas à $\text{Ker}(C(n, d))$. Nous rappelons que notre raisonnement est fondé sur une unique itération de l'algorithme de Gauss randomisé. Les probabilités calculées nous permettent ainsi de déterminer le nombre nécessaire d'itérations à la reconstruction. Pour un n_e donné, l'algorithme de Gauss randomisé renvoie un noyau randomisé non restreint à 0 dans les deux configurations suivantes :

1. $\tilde{C}(n_e, d_e)$ n'est pas de rang plein,
2. $\tilde{C}(n_e, d_e)$ est de rang plein et au moins un des vecteurs candidats a été validé.

On cherche ainsi à évaluer

$$\begin{aligned} & \Pr \left((l, \gamma)\text{-RKer}(\tilde{C}(n_e, d_e)) \neq \emptyset \right) \\ &= \\ & \Pr \left(\text{rg}(\tilde{C}(n_e, d_e)) < n_e \right) \Pr \left((l, \gamma)\text{-RKer}(\tilde{C}(n_e, d_e)) \neq \emptyset \mid \text{rg}(\tilde{C}(n_e, d_e)) < n_e \right) \\ &+ \\ & \Pr \left(\text{rg}(\tilde{C}(n_e, d_e)) = n_e \right) \Pr \left((l, \gamma)\text{-RKer}(\tilde{C}(n_e, d_e)) \neq \emptyset \mid \text{rg}(\tilde{C}(n_e, d_e)) = n_e \right), \end{aligned}$$

sous la forme $\mathcal{P}_{fa}^{(1)}(N, n_e) + \mathcal{P}_{fa}^{(2)}(N, n_e)$.

Proposition 2.6 *Soit $n_e \neq \beta n$,*

$$\mathcal{P}_{fa}^{(1)}(N, n_e) = 1 - \prod_{i=0}^{n_e-1} (1 - 2^{i-N}).$$

Preuve :

Dans la première configuration, si $\tilde{C}(n_e, d)$ n'est pas de rang plein, au moins un des vecteurs de $\mathcal{B}_1$ appartient au noyau de $\tilde{C}(n_e, d)$. Ce vecteur sera donc validé par l'étape statistique avec probabilité égale à 1. On en déduit $\mathcal{P}_{fa}^{(1)}(N, n_e) = \Pr \left(\text{rg}(\tilde{C}(n_e, d_e)) < n_e \right) = \Pr \left(R(\tilde{C}(n_e, d_e)) < n_e \right)$. La relation (2.9) page 62 nous permet de conclure. ■

Dans la seconde configuration, la probabilité pour $\tilde{C}(n_e, d_e)$ d'être de rang plein est donnée par

$$\Pr(\text{rg}(\tilde{C}(n_e, d_e)) = n_e) = \prod_{i=0}^{n_e-1} (1 - 2^{i-N}),$$

d'après la relation (2.8) page 62. $\tilde{C}(n_e, d_e)$ est donc non seulement de rang plein mais l'algorithme de Gauss randomisé valide au moins un vecteur qui n'est pas une relation de parité. Il ne nous reste plus qu'à évaluer la probabilité d'un tel événement.

La proposition 2.1 page 55 permet d'expliciter la probabilité qu'au moins un vecteur v de $\mathcal{B}_1$ soit validé sachant $v \notin \text{Ker}(\tilde{C}(n_e, d_e))$ dans la configuration 2.

$$\begin{aligned}
& \Pr \left((l, \gamma)\text{-RKer}(\tilde{C}(n_e, d_e)) \neq \emptyset \mid \text{rg}(\tilde{C}(n_e, d_e)) = n_e \right) \\
&= \Pr \left(\exists v \in (l, \gamma)\text{-RKer}(\tilde{C}(n_e, d_e)) \mid v \notin \text{Ker} \tilde{C}(n_e, d_e) \right) \\
&= \Pr \left(\exists v \in \mathcal{B}_1, D(v) \leq \gamma \mid v \notin \text{Ker} \tilde{C}(n_e, d_e) \right) \\
&= 1 - \Pr \left(\forall v \in \mathcal{B}_1, W(v) > \gamma N \mid v \notin \text{Ker} \tilde{C}(n_e, d_e) \right) \\
&= 1 - \left(\Pr \left(W(v) > \gamma N \mid v \notin \text{Ker} \tilde{C}(n_e, d_e) \right) \right)^{n_e},
\end{aligned}$$

en supposant les n_e variables aléatoires $(W(v_i))_{v_i \in \mathcal{B}_1}$ indépendantes. D'après le théorème 2.1 page 55, on en déduit la probabilité de fausse alarme dans la configuration 2

$$\mathcal{P}_{fa}^{(2)}(N, n_e) = \left(\prod_{i=0}^{n_e-1} (1 - 2^{i-N}) \right) \left(1 - \left(2^{-N} \sum_{x=\lfloor \gamma N \rfloor + 1}^N \binom{N}{x} \right)^{n_e} \right).$$

Finalement, la probabilité $\mathcal{P}_{fa}^{(3)}$ que l'algorithme de Gauss randomisé renvoie au moins un vecteur qui n'appartient pas à $\text{Ker}(C(n, d))$, dans le cas $n_e \neq \beta n$ est donnée par $\mathcal{P}_{fa}^{(1)} + \mathcal{P}_{fa}^{(2)}$,

$$\mathcal{P}_{fa}^{(3)}(N, n_e) = 1 - (2^{-n_e N}) \prod_{i=0}^{n_e-1} (1 - 2^{i-N}) \left(\sum_{x=\lfloor \gamma N \rfloor + 1}^N \binom{N}{x} \right)^{n_e}.$$

En pratique, cette probabilité est difficile à évaluer, néanmoins nous pouvons en donner une majoration assez fine.

Proposition 2.7

$$\mathcal{P}_{fa}^{(3)}(N, n_e) \leq 2^{-N} + n_e(1 - 2^{-N+n_e})e^{-2N(\frac{1}{2}-\gamma)^2}.$$

Preuve :

la minoration calculée en annexe A page 249 nous donne

$$\prod_{i=0}^{n_e-1} (1 - 2^{i-N}) \geq 1 - 2^N(2^{n_e} - 1) \geq 1 - 2^{-N+n_e}, \quad (2.11)$$

et d'autre part, la somme des coefficients binomiaux peut être elle aussi minorée en utilisant l'inégalité suivante [144],

$$\forall \lambda \in [0, \frac{1}{2}] \quad \sum_{i=0}^{\lambda p} \binom{p}{i} \leq 2^p e^{-2p(\frac{1}{2}-\lambda)^2}. \quad (2.12)$$

Nous obtenons donc,

$$\begin{aligned}
\left(2^{-N} \sum_{x=\lfloor \gamma N \rfloor + 1}^N \binom{N}{x} \right)^{n_e} &= \left(1 - 2^{-N} \sum_{x=0}^{\lfloor \gamma N \rfloor} \binom{N}{x} \right)^{n_e}, \\
&\geq \left(1 - e^{-2N(\frac{1}{2}-\gamma)^2} \right)^{n_e}, \\
&\geq 1 - n_e e^{-2N(\frac{1}{2}-\gamma)^2}.
\end{aligned} \quad (2.13)$$

Les équations (2.11) page précédente et (2.13) page précédente nous permettent d'écrire

$$\begin{aligned}\mathcal{P}_{fa}^{(3)}(N, n_e) &\leq 1 - (1 - 2^{-N+n_e})(1 - n_e e^{-2N(\frac{1}{2}-\gamma)^2}), \\ &\leq 2^{-N} + n_e(1 - 2^{-N+n_e})e^{-2N(\frac{1}{2}-\gamma)^2}. \blacksquare\end{aligned}$$

Comme nous le supposons, la probabilité de fausse alarme tend exponentiellement vers 0 quand N augmente.

$$\lim_{N \rightarrow \infty} \mathcal{P}_{fa}^{(3)}(N, n_e) = 0.$$

Pour conclure le cas $n_e \neq \beta n$, nous négligeons la probabilité de détecter un vecteur de $\text{Ker}(C(n, d))$, les relations de parité n'étant pas conservées comme l'illustre la figure 2.4 page 62.

$$\mathcal{P}_{det}^{(1)}(N, n_e) \text{ négligeable.}$$

Cas $n_e = \beta n$

Dans un premier temps nous étudions la probabilité de fausse alarme $\mathcal{P}_{fa}^{(4)}(N, \beta n)$, c'est-à-dire la probabilité de détecter au moins un vecteur qui n'est pas une relation de parité. Comme précédemment, nous raisonnons avec une unique itération de l'algorithme de Gauss randomisé.

Proposition 2.8

$$\mathcal{P}_{fa}^{(4)}(N, \beta n) \leq \frac{(\beta n)^2}{2} e^{-2N(\frac{1}{2}-\gamma)^2}.$$

Preuve :

notons $\mathcal{B}'_1$ l'ensemble des vecteurs de $\mathcal{B}_1$ n'appartenant pas à $\text{Ker}(C(\beta n, d_e))$. $\mathcal{P}_{fa}^{(4)}(N, \beta n)$ peut s'exprimer en fonction du cardinal de $\mathcal{B}'_1$.

$$\begin{aligned}\mathcal{P}_{fa}^{(4)}(N, \beta n) &= \Pr(\exists v \in \mathcal{B}'_1, D(v) \leq \gamma \mid v \notin \mathcal{H}), \\ &= \sum_{i=0}^{\beta n} \Pr(\exists v \in \mathcal{B}'_1, W(v) \leq \gamma N \mid v \notin \mathcal{H}, \#\mathcal{B}'_1 = i) \Pr(\#\mathcal{B}'_1 = i), \\ &\leq \sum_{i=0}^{\beta n} \Pr(\exists v \in \mathcal{B}'_1, W(v) \leq \gamma N \mid v \notin \mathcal{H}, \#\mathcal{B}'_1 = i), \\ &\leq \sum_{i=0}^{\beta n} \left(1 - \Pr(\forall v \in \mathcal{B}'_1, W(v) > \gamma N \mid v \notin \mathcal{H}, \#\mathcal{B}'_1 = i)\right), \\ &\leq \sum_{i=0}^{\beta n} 1 - \left(2^{-N} \sum_{x=\lfloor \gamma N \rfloor + 1}^N \binom{N}{x}\right)^i,\end{aligned}$$

d'après le théorème 2.1 page 55 et en supposant les $(W(v_i))_{v_i \in \mathcal{B}'_1}$ indépendants. En injectant l'équation (2.13) page précédente,

$$\mathcal{P}_{fa}^{(4)}(N, \beta n) \leq \sum_{i=0}^{\beta n} i e^{-2N(\frac{1}{2}-\gamma)^2} = \frac{(\beta n)((\beta n) - 1)}{2} e^{-2N(\frac{1}{2}-\gamma)^2}. \blacksquare$$

Là encore, la probabilité de fausse alarme tend exponentiellement vers 0 quand N augmente.

$$\lim_{N \rightarrow \infty} \mathcal{P}_{fa}^{(4)}(N, n_e) = 0.$$

Pour évaluer la probabilité que l'algorithme de Gauss randomisé renvoie une relation de parité, i.e.

$$\Pr \left(h \in (l, \gamma)\text{-RKer } (\tilde{C}(\beta n, d) \mid h \in \mathcal{H}) \right),$$

nous étudions deux configurations distinctes :

1. h appartient à $\text{Ker } (\tilde{C}_1(\beta n, d_e))$,
2. h appartient à $\mathcal{B}_1$ mais pas à $\text{Ker } (\tilde{C}_1(\beta n, d_e))$.

Pour ce faire, nous introduisons W_i , $i \in \{1, 2\}$, les variables aléatoires discrètes à valeurs dans $[0, \beta n]$, respectivement $[0, N - \beta n]$, définies sur les vecteurs binaires v de longueur βn par $W_i(v) = w(\tilde{C}_i(\beta n, d)v)$. Nous écrivons alors la probabilité recherchée sous la forme

$$\begin{aligned} & \Pr \left(h \in (l, \gamma)\text{-RKer } (\tilde{C}(\beta n, d) \mid h \in \mathcal{H}) \right) \\ &= \Pr \left((h \in \mathcal{B}_1) \cap (W(h) \leq \gamma N) \mid h \in \mathcal{H} \right), \\ &= \Pr \left((h \in \mathcal{B}_1) \cap (W_1(h) = 0) \cap (W(h) \leq \gamma N) \mid h \in \mathcal{H} \right) \\ &+ \Pr \left((h \in \mathcal{B}_1) \cap (W_1(h) > 0) \cap (W(h) \leq \gamma N) \mid h \in \mathcal{H} \right), \\ &= \Pr(W_1(h) = 0 \mid h \in \mathcal{H}) \Pr(W(h) \leq \gamma N \mid h \in \mathcal{H}, W_1(h) = 0) \\ &+ \Pr \left((h \in \mathcal{B}_1) \cap (W_1(h) > 0) \mid h \in \mathcal{H} \right) \\ &\quad \Pr(W(h) \leq \gamma N \mid h \in \mathcal{H}, (h \in \mathcal{B}_1) \cap (W_1(h) > 0)), \\ &= \mathcal{P}_{det}^{(2)} + \mathcal{P}_{det}^{(3)}, \end{aligned}$$

en remarquant que $\forall h \in \text{Ker } (\tilde{C}_1(\beta n, d))$, $h \in \mathcal{B}_1$.

Nous calculons tout d'abord la probabilité que h appartienne à $\text{Ker } (\tilde{C}_1(\beta n, d_e))$. Pour ce faire nous utilisons la proposition suivante.

Proposition 2.9 *Soit h une relation de parité, i.e. $h \in \text{Ker } (C(\beta n, d))$ alors*

$$h \in \text{Ker } (\tilde{C}_1(\beta n, d)) \text{ si et seulement si } \sum_{j|h(j)=1} [E_1(\beta n, d)]_{ij} \equiv 0 \pmod{2} \quad \forall j = 1 \dots \beta n.$$

Preuve : découle de l'équation (2.3) page 55. ■

On en déduit la proposition suivante.

Proposition 2.10 *Soit $h \in \text{Ker } (C(\beta n, d_e))$, de poids de Hamming $w(h)$, alors la probabilité que h appartienne à $\text{Ker } (\tilde{C}_1(\beta n, d))$ est donnée par*

$$\Pr(W_1(h) = 0 \mid h \in \mathcal{H}) = \left(\frac{1 + (1 - 2\varepsilon)^{w(h)}}{2} \right)^{\beta n}.$$

Preuve : application directe de la proposition 2.2 page 55 et de la proposition 2.10. ■

Pour calculer la probabilité que h soit à la fois dans la configuration 1 et soit détecté, il reste maintenant à évaluer la probabilité $\Pr(W(h) \leq \gamma N \mid h \in \mathcal{H}, W_1(h) = 0)$ que le

vecteur h soit validé par l'algorithme de Gauss randomisé. En s'inspirant du théorème 2.1 page 55 ,

$$\begin{aligned} \Pr(W(h) \leq \gamma N \mid h \in \mathcal{H}, W_1(h) = 0) &= \Pr(W_2(h) \leq \gamma N \mid h \in \mathcal{H}), \\ &= \sum_{i=0}^{\lfloor \gamma N \rfloor} \binom{N - \beta n}{i} (1 - P_p(h))^i P_p(h)^{(N - \beta n - i)}. \end{aligned}$$

Proposition 2.11 *La probabilité $\mathcal{P}_{det}^{(2)}(N, \beta n, h, \gamma)$ qu'une relation de parité $h \in \mathcal{H}$ soit dans la configuration 1 et validée par l'algorithme de Gauss randomisé est minorée par*

$$\mathcal{P}_{det}^{(2)}(N, \beta n, h, \gamma) \geq P_p(h)^{\beta n} \left(1 - \frac{(1 - P_p(h))}{\gamma^2} \left(\frac{P_p(h)}{N} - (1 - P_p(h)) \right) \right).$$

Preuve :

par définition de $\mathcal{P}_{det}^{(2)}$,

$$\begin{aligned} \mathcal{P}_{det}^{(2)}(N, \beta n, h, \gamma) &= P_p(h)^{\beta n} \Pr(W_2(h) \leq \gamma N \mid h \in \mathcal{H}) \\ &= P_p(h)^{\beta n} (1 - \Pr(W_2(h) > \gamma N \mid h \in \mathcal{H})). \end{aligned}$$

La variable aléatoire $W_2(h)$ suit une loi binomiale de paramètres $(N - \beta n, (1 - P_p(h)))$ sur $\mathcal{H}$. L'inégalité Markov [33, p. 80], nous permet d'écrire

$$\Pr(W_{2|\mathcal{H}}(h) > \gamma N) \leq \frac{E(W_{2|\mathcal{H}}^2(h))}{\gamma^2 N^2} = \frac{V(W_{2|\mathcal{H}}(h)) + E(W_{2|\mathcal{H}}(h))^2}{\gamma^2 N^2}.$$

Nous en déduisons

$$\begin{aligned} \mathcal{P}_{det}^{(2)}(N, \beta n, h, \gamma) &\geq P_p(h)^{\beta n} \left(1 - \frac{(N - \beta n)P_p(h)(1 - P_p(h)) + (N - \beta n)^2(1 - P_p(h))^2}{\gamma^2 N^2} \right), \\ &\geq P_p(h)^{\beta n} \left(1 - \frac{P_p(h)(1 - P_p(h)) + N(1 - P_p(h))^2}{\gamma^2 N} \right). \end{aligned}$$

■

D'autre part, nous avons une majoration évidente

$$P_p(h)^{\beta n} \geq \mathcal{P}_{det}^{(2)}(N, \beta n, h, \gamma).$$

Ceci nous permet d'en déduire un encadrement quand N tend vers l'infini,

$$P_p(h)^{\beta n} \geq \lim_{N \rightarrow +\infty} \mathcal{P}_{det}^{(2)}(N, \beta n, h, \gamma) \geq P_p(h)^{\beta n} \left(1 - \left(\frac{1 - P_p(h)}{\gamma} \right)^2 \right).$$

Augmenter le nombre de mots de code interceptés où le seuil permet d'améliorer sensiblement la probabilité de détection, néanmoins, celle-ci reste bornée. Cette borne montre clairement que les facteurs limitants sont, comme l'on pouvait s'y attendre, la taille des mots de codes, n , l'erreur du canal, ε et le poids de la relation de parité, $w(h)$. En effet, lorsque ε ou $w(h)$ augmente, $P_p(h)$ diminue.

Par ailleurs, pour étudier le comportement de la probabilité de détection quand l'erreur du canal tend vers 0, il faut noter que $P(h)$ tend vers 1.

$$\lim_{\varepsilon \rightarrow 0} \mathcal{P}_{det}^{(2)}(N, \beta n, h, \gamma) = \lim_{P(h) \rightarrow 1} \mathcal{P}_{det}^{(2)}(N, \beta n, h, \gamma) = 1.$$

En conclusion, la probabilité de détection tend d'autant plus vite vers 1 que l'erreur du canal est faible et que N est grand.

Enfin, la configuration 2 apparaît notamment quand il existe une ligne de la matrice $\tilde{C}_1(\beta n, d_e)$ pour laquelle le nombre d'erreurs pris sur les colonnes associées à une relation de parité est impair et que celles-ci restent strictement sous la diagonale tout au long de l'algorithme de Gauss (sauf pour la dernière colonne traitée par Gauss pour laquelle on autorise que l'erreur soit sur la diagonale). Pour bien visualiser le phénomène, considérons l'exemple suivant.

Exemple 2.2

$$\begin{aligned} \tilde{C}_1(n, d) &= \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\ &= C_1(n, d) + E_1(n, d). \end{aligned}$$

La somme des colonnes 1, 3 et 5 de $C_1(n, d)$ vaut 0, ce qui correspond à la relation de parité $h = (1, 0, 1, 0, 1)$. Nous remarquons aussi que 1 seule erreur apparaît à la ligne 4 et 5 pour les colonnes 1, 3 et 5, comme décrit précédemment. En effectuant un pivot de Gauss sur les colonnes de $\tilde{C}_1(n, d)$, nous obtenons $h \in \mathcal{B}_1$ mais pas à $\text{Ker}(\tilde{C}_1(n, d))$, ce qui correspond bien à la configuration 2.

Malheureusement, comme certaines lignes sont éventuellement échangées lors du processus de Gauss, il se peut que des erreurs en nombre impair au dessus de la diagonale se retrouvent dessous et réciproquement. Dans ce cas, nous ne pouvons pas estimer la probabilité associée à la configuration 2. La probabilité de détection est alors minorée par la probabilité de détection dans la configuration 1 et a été évaluée expérimentalement.

Analyse de complexité

Pour analyser la complexité en temps de l'algorithme de reconstruction, nous devons d'abord évaluer le paramètre l nécessaire à l'algorithme de Gauss randomisé. Nous avons exprimé la probabilité de détection d'une relation de parité h en fonction notamment de son poids de Hamming. Il en résulte que plus une relation de parité possède un poids élevé, plus elle est difficile à détecter. Il nous faut donc se focaliser sur celle qui est le plus difficile à détecter, c'est-à-dire celle de poids maximum. Notons $\mathcal{P}_{det}(N, n)$ la probabilité de détecter cette relation de parité. Nous avons par définition,

$$\mathcal{P}_{det} = \min_{h \in \text{Ker}(n, d)} \{\mathcal{P}_{det}^{(2)}(N, n, h)\}.$$

En choisissant, $l > \frac{1}{\mathcal{P}_{det}}$, avec forte probabilité, cette relation sera détectée, ainsi que les autres relations de parité indépendantes.

De même, si nous posons $\mathcal{P}_{fa} = 2^{-N} + \frac{n^2}{2}e^{-\frac{N}{2}}$, alors $\mathcal{P}_{fa}^{(3)}(N, n_e) \leq \mathcal{P}_{fa}$, $\forall n_e, d_e$ ainsi que $\mathcal{P}_{fa}^{(4)}(N, n_e)$ d'après les propositions 2.7 page 71 et 2.8 page 72 . Il nous faut donc choisir

$$\frac{1}{\mathcal{P}_{det}} < l \ll \frac{1}{\mathcal{P}_{fa}}.$$

Nous en déduisons ainsi le nombre minimum N_{min} de mots de code à intercepter pour pouvoir effectuer les l tirages de Monte Carlo.

$$l \leq \sqrt{\binom{N_{min}}{n}}.$$

Le nombre minimum de bits à intercepter est donc nN_{min} .

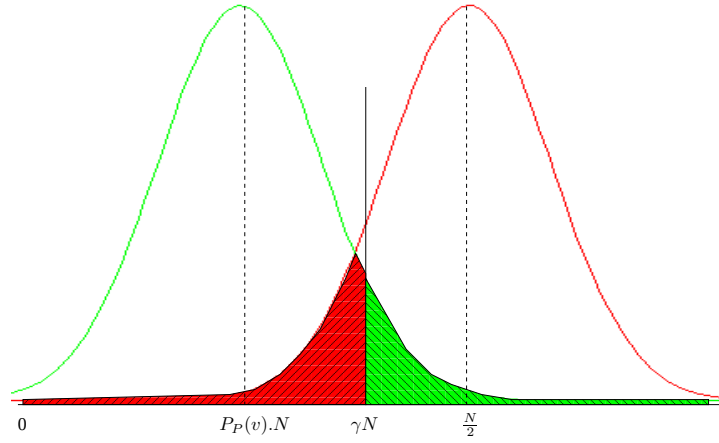


FIG. 2.7 – Lois suivies par W

D'autre part, le choix de γ est primordial pour maîtriser les fausses alarmes et les non-détections. En effet, γN est le seuil de décision séparant deux lois de probabilités décrites dans le théorème 2.1 page 55 ; il permet de discriminer ainsi les deux hypothèses, « v appartient à $\text{Ker}(C(n_e, d_e))$ ou non ». La probabilité de fausse alarme, correspond à l'aire pleine à gauche du seuil et la probabilité de non-détection, à l'aire pleine à droite du seuil sur la figure 2.7.

Une des stratégies consiste à choisir un γ_{opt} qui minimise l'erreur lors de l'étape statistique, c'est-à-dire la somme des aires pleines.

$$\gamma_{opt}(v) = \underset{\gamma}{\text{Argmin}} \left(1 + \sum_{x=0}^{\lfloor \gamma N \rfloor} \binom{N}{i} (2^{-N} - (1 - P_P(v))^i P_P(v)^{N-i}) \right).$$

En pratique, nous avons plutôt tendance à minimiser la probabilité de fausse alarme. En effet, une fausse alarme est beaucoup plus coûteuse en traitements ultérieurs qu'une non-détection. Enfin, comme dans le cas du canal binaire sans erreur, les paramètres n et d

doivent être estimés *en aveugle*. La complexité en temps de l'algorithme de reconstruction est donc

$$\mathcal{O}\left(\frac{n^5}{\mathcal{P}_{det}}\right).$$

2.4 Résultats expérimentaux

Dans ce paragraphe, nous procédons en deux temps. Nous validons tout d'abord les résultats théoriques sur des exemples simples dont les courbes permettent de bien visualiser le comportement des différentes probabilités en fonction des paramètres de taille, de poids des relations de parité et d'erreur. Puis, nous testons les algorithmes sur des paramètres de codes en blocs linéaires utilisés dans différentes normes citées en annexe B.

Dans un premier temps, nous générons aléatoirement des matrices d'interception de taille $N \times n$ et de dimension $(n - 1)$. L'unique relation de parité est de poids w . Nous simulons ensuite l'action du canal de transmission en introduisant aléatoirement des erreurs suivant une loi uniforme de paramètre ε . Pour chacune des valeurs de ces paramètres, 1000 tirages sont effectués, les courbes tracées sont ensuite moyennées sur l'ensemble de ces tirages. De plus, pour l'algorithme de Gauss randomisé, nous fixons γ égal à 1 et évaluons *a posteriori* les courbes en fonction du paramètre de seuil. Le nombre d'itérations pour ce même algorithme a été fixé à $l = 1000$.

2.4.1 Impact du poids d'une relation de parité

Pour mesurer l'impact du poids d'une relation de parité, nous fixons $\varepsilon = 10^{-2}$, $n = 10$, $N = 100$ et nous faisons varier w de 2 à 10. Les performances sont représentées sous forme de courbe ROC (*Receiver Operating Characteristic*). Traditionnellement, les courbes ROC sont représentatives des performances des détecteurs statistiques et permettent ainsi de les comparer entre eux. Elles représentent la *probabilité de vrais positifs*, *i.e.* la probabilité de détecter et valider une relation de parité, en fonction de la probabilité de fausse alarme et mettent ainsi en évidence le compromis à faire lors du choix du seuil. En règle générale, on se fixe une probabilité maximum de fausse alarme, puis on choisit le seuil en conséquence.

Dans le cadre de notre étude, les courbes ROC sont quasi-constantes et correspondent à la probabilité maximale de détection. La figure 2.8 page suivante représente les courbes ROC pour les différentes valeurs du poids de la relation de parité. Elle illustre ainsi l'impact exponentiel du poids de la relation de parité. La forme un peu particulière de ces courbes s'explique par le fait que les deux distributions de probabilité sont très éloignées. Comme on recherche des vecteurs de poids faible, un seuil petit permet de détecter rapidement toutes les relations de parité sans fausse alarme. Dès que le seuil augmente, toutes les relations de parité sont détectées mais on voit alors apparaître des fausses alarmes. Il est alors assez naturel de s'intéresser à l'évolution des probabilités de détection et de fausse alarme en fonction du seuil.

L'expérimentation révèle tout d'abord une probabilité de détection de décroissance exponentielle avec l'augmentation du poids de la relation de parité, comme l'illustre la figure 2.9 page suivante. De plus, l'écart avec les courbes théoriques de minoration met en

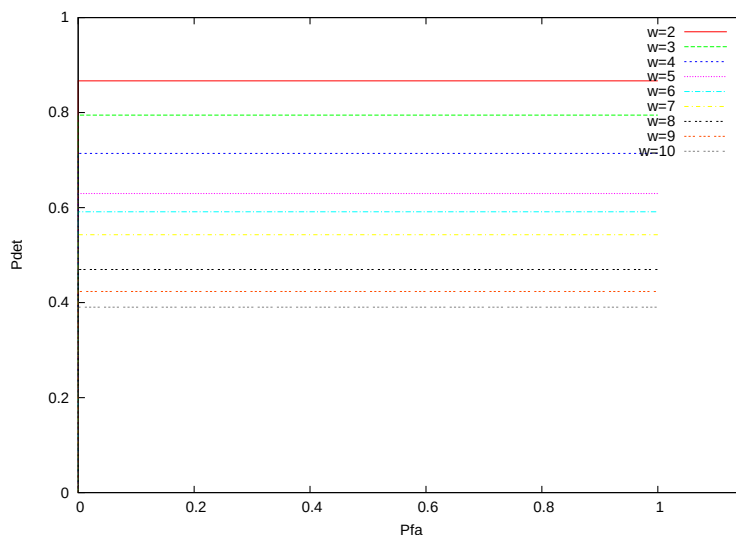
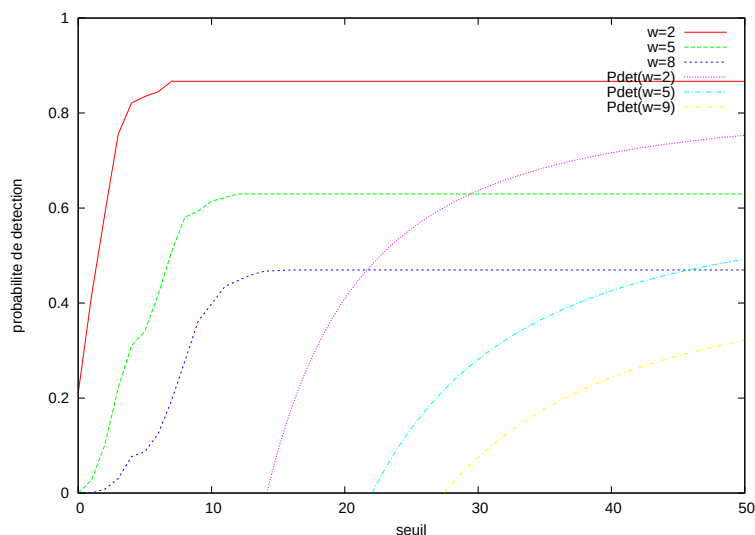


FIG. 2.8 – Courbes ROC en fonction du poids de la relation de parité

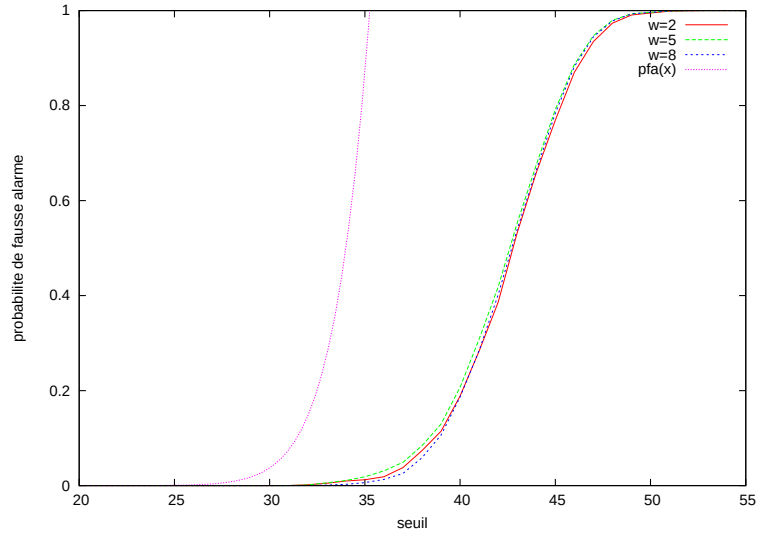
FIG. 2.9 – Probabilité de détection en fonction du seuil, paramètre w

évidence que la configuration 2 du calcul de la probabilité de détection est non négligeable.

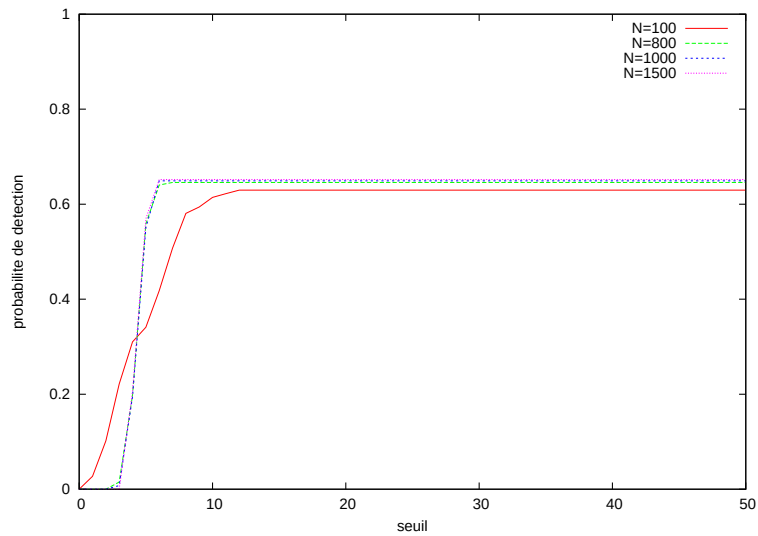
Enfin, la figure 2.10 page suivante confirme que le poids des relations de parité n'a pas d'impact sur la probabilité de fausse alarme de l'algorithme. De plus, la majoration théorique annoncée est vérifiée.

2.4.2 Impact du nombre de mots de code interceptés

Pour mesurer l'impact du nombre de mots de code interceptés, nous fixons $\varepsilon = 10^{-2}$, $n = 10$, $w = 5$ et nous faisons varier N de 100 à 1500. Les courbes ROC étant là encore quasi-constantes, nous nous intéressons principalement aux courbes des probabilités de détection et de fausse alarme en fonction du seuil.

FIG. 2.10 – Probabilité de fausse alarme en fonction du seuil, paramètre w

La figure 2.11 met en évidence que l'augmentation du nombre d'observations favorise légèrement la détection, même si celle-ci est bornée. En fait, augmenter le nombre d'observations fait diminuer notablement le nombre de fausses alarmes (fig. 2.12 page suivante). Dans des conditions opérationnelles, N est plutôt « grand » par rapport à n . Dans notre exemple, $n = 10$ et $N = 1000$ mots de code correspondent à seulement 9,6 Ko.

FIG. 2.11 – Probabilité de détection en fonction du seuil, paramètre N

2.4.3 Impact de l'erreur

Pour mesurer l'impact de l'erreur introduite par le canal, nous fixons $n = 10$, $w = 5$, $N = 100$ et nous faisons varier ε de 10^{-5} à $5 \cdot 10^{-2}$. Les courbes ROC étant là encore quasi-constantes, nous nous intéressons principalement aux courbes des probabilités de

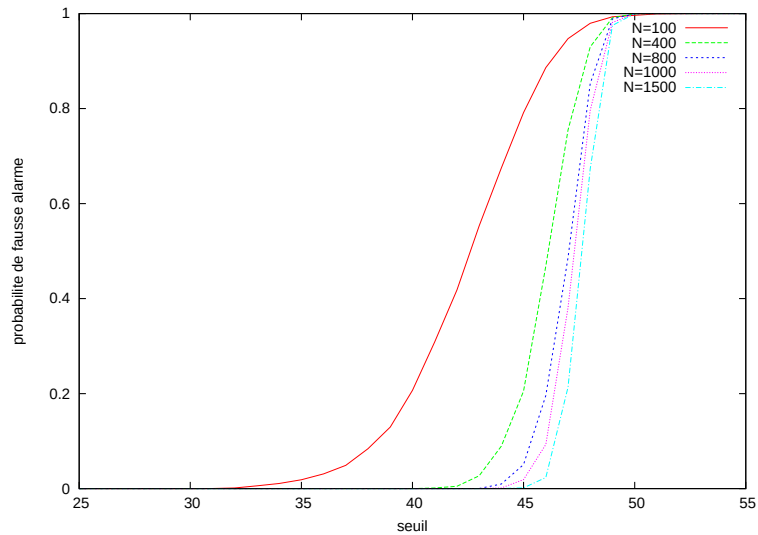


FIG. 2.12 – Probabilité de fausse alarme en fonction du seuil, paramètre N

détection et de fausse alarme en fonction du seuil.

La figure 2.13 illustre l'importance du facteur d'erreur sur la probabilité de détection de la relation de parité. La moindre correction d'erreur pendant le processus de reconstruction permet ainsi un gain notable de la probabilité de détection. La figure 2.14 page suivante confirme que la probabilité de fausse alarme est indépendante du bruit introduit par le canal.

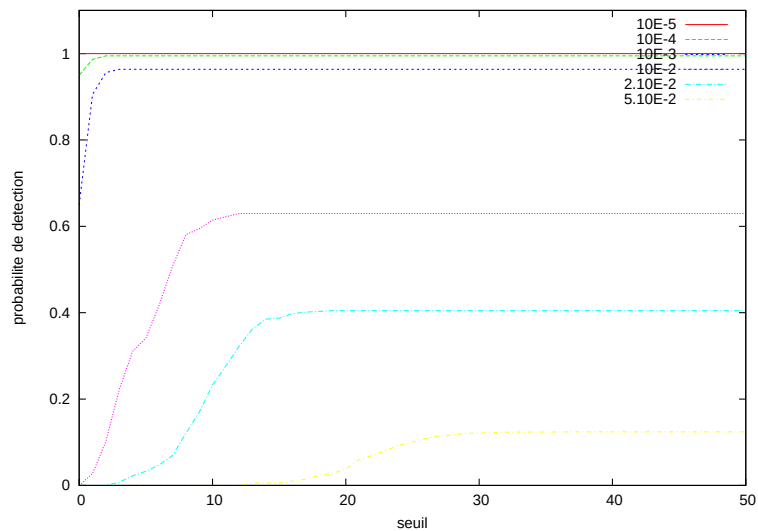
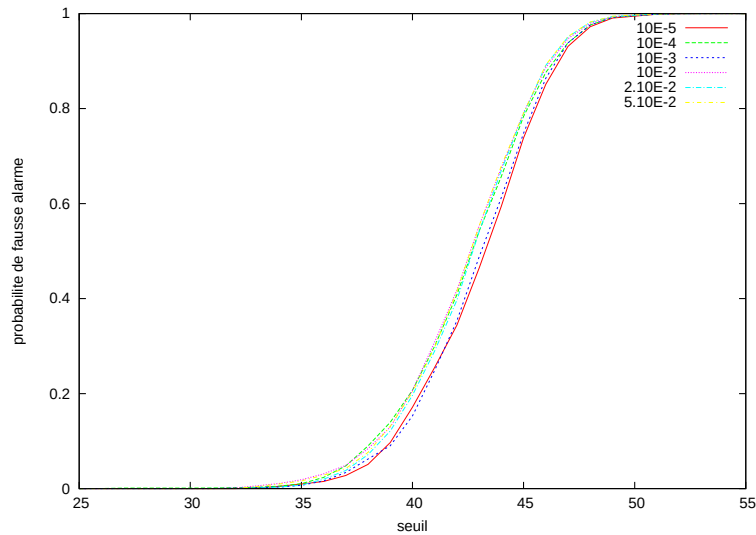


FIG. 2.13 – Probabilité de détection en fonction du seuil, paramètre ε

2.4.4 Des codes bien réels

Dans ce paragraphe, nous présentons les résultats expérimentaux de la reconstruction des codes en blocs linéaires extraits des normes présentées dans l'annexe B. De plus, tout

FIG. 2.14 – Probabilité de fausse alarme en fonction du seuil, paramètre ε

comme M. Cluzeau [51], nous nous sommes intéressés à la reconstruction de codes Low Density Parity Check (LDPC) et d'un code linéaire aléatoire. Ces codes LDPC possèdent la bonne propriété d'avoir des relations de parité de poids faible, ce qui favorise le processus de reconstruction comme l'illustrent les résultats 2.2 page 84 et 2.16 page suivante. L'objectif de ces simulations est de mettre en évidence de « bonnes » valeurs pour les paramètres γ (seuil de détection) et l (nombre d'itérations) de l'algorithme de Sicot-Houcke. Pour ce faire, nous avons tout d'abord fait varier le TEB de 10^{-5} à $2 \cdot 10^{-1}$. Enfin, nous nous sommes placés dans le cas où n et d ont été correctement estimés. Le nombre d'itérations et le temps nécessaires pour reconstruire le code sont alors mesurés. Lorsque tout le code est reconstruit, le nombre d'itérations nous donne une idée sur la probabilité de détection de la relation de parité de poids le plus fort et donc la plus difficile à détecter. Les algorithmes ont été implémentés en Magma et les simulations ont été effectuées sur des processeurs AMD Thurion 2000+ MP, cadencés à 1,6 GHz ; le nombre de mots interceptés a été fixé à 2000, ce qui correspond à une quantité variant de 7,6 Ko à 245 Ko suivant les codes.

Comme nous l'avons montré dans la partie théorique, un des facteurs limitants est la taille du code. Les codes en blocs linéaires les plus longs n'ont pas pu être reconstruits en un temps raisonnable. De plus, comme l'illustrent les résultats, le nombre d'itérations augmente avec la longueur du code de façon exponentielle. De la même manière, le nombre d'itérations augmente de façon exponentielle avec l'erreur du canal. En pratique, pour un même TEB et des longueurs de code comparables, les LDPC apparaissent comme plus faciles à reconstruire que les codes cycliques. Les codes aléatoires présentent quant à eux une difficulté intermédiaire. Cela s'explique aisément par le fait que les LDPC possèdent des relations de parité de poids faible. Il est à noter néanmoins que la reconstruction des codes cycliques n'est pas optimale dans le sens où la structure même de ce type de code n'a pas été exploitée ; on peut espérer faire mieux en prenant en compte les propriétés algébriques de tels codes. Pour les LDPC, les algorithmes décrochent aux alentours d'une erreur de $5 \cdot 10^{-3}$ tandis que pour les codes cycliques la limite semble plutôt être proche de 10^{-3} et pour les codes aléatoires proche de 10^{-2} . D'autre part, il faut noter qu'aucune

fausse alarme n'a été détectée sur l'ensemble des simulations. Nous nous sommes limités à une journée sans détecter de nouvelle relation de parité pour conclure à la non détection du code (ND).

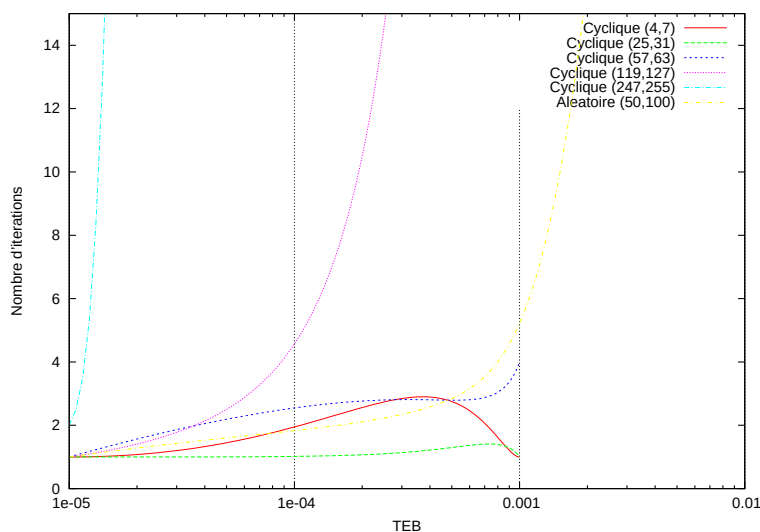


FIG. 2.15 – Reconstruction des codes cycliques

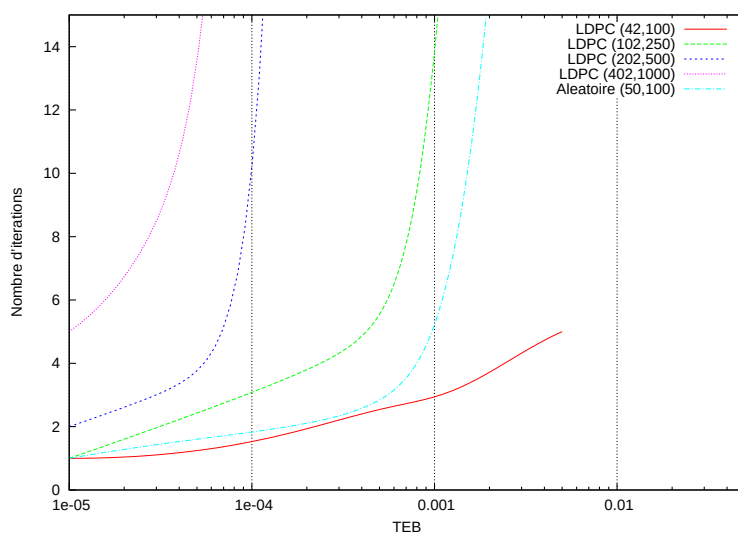


FIG. 2.16 – Reconstruction de codes LDPC

2.5 Conclusion et perspectives

Les algorithmes de reconstruction présentés précédemment nous permettent de retrouver les relations de parité, c'est-à-dire le code dual $\mathcal{H}$ du code $\mathcal{C}$ à reconstruire et enfin $\mathcal{C}$. Malheureusement, sans aucune autre hypothèse, trouver le codeur utilisé et donc décoder est équivalent au problème de décodage d'un code aléatoire dont on sait que c'est un problème NP-complet [29].

Codes	(k, n)	TEB 10 ⁻⁴		TEB 3.10 ⁻⁴		TEB 5.10 ⁻⁴		TEB 8.10 ⁻⁴		TEB 10 ⁻³	
		temps	nb ité.	temps	nb ité.	temps	nb ité.	temps	nb ité.	temps	nb ité.
GSM g _{c0}	(4,7)										
100%		0.02 s	1	0.08 s	3	0.03 s	3	0.04	3	0.03 s	1
CDMA 2000 g _{c0}	(25,31)										
100%		0.1 s	1	0.11 s	1	0.11 s	1	0.21 s	2	0.11 s	1
GSM g _{c0}	(57,63)										
90%		0.21 s	1	0.42 s	2	0.41 s	2	-	-	-	-
100%		0.63 s	3	0.63 s	3	0.61 s	3	0.43 s	2	0.86 s	4
UMTS g _{c0}	(119,127)										
CDMA 2000 g _{c2}											
25%		-	-	0.86 s	2	0.83 s	2	3.11 s	8	5.88 s	14
50%		-	-	1.28 s	3	1.68 s	4	12.38 s	32	24.88 s	59
90%		0.42 s	1	1.7 s	4	2.99 s	7	20.44 s	53	50.74 s	120
100%		0.83 s	2	2.12 s	5	3.85 s	9	20.82 s	54	2 mn	284
GSM g _{c2}	(247,255)										
25%		1.67 s	2	10.88 s	13	51 mn 24 s	3999	ND	ND	ND	ND
50%		2.48 s	3	1 mn 21 s	98	4 h 15 mn	19 772	ND	ND	ND	ND
90%		4.97 s	6	3 mn 53 s	282	11 h 3 mn	51 540	ND	ND	ND	ND
100%		11.55 s	14	4 mn 21 s	315	11 h 31 mn	53 700	ND	ND	ND	ND

TAB. 2.1 – Reconstruction des codes en blocs des normes GSM, UMTS et CDMA 2000, $\gamma = 0.15$

<i>Codes</i>	(k, n)	TEB 5.10^{-4}		TEB 8.10^{-4}		TEB 10^{-3}		TEB 2.10^{-3}		TEB 5.10^{-3}	
		<i>temps</i>	<i>nb ité.</i>	<i>temps</i>	<i>nb ité.</i>	<i>temps</i>	<i>nb ité.</i>	<i>temps</i>	<i>nb ité.</i>	<i>temps</i>	<i>nb ité.</i>
<i>LDPC 100</i>	<i>(42,100)</i>										
50%		-	-	-	-	-	-	-	-	0.65 s	2
90%		0.32 s	1	0.64 s	2	-	-	0.64 s	2	0.98 s	3
100%		0.95 s	3	0.95 s	3	0.62 s	2	1.27 s	4	1.63 s	5
<i>LDPC 250</i>	<i>(102,250)</i>										
50%		-	-	-	-	-	-	1.61 s	2	2 mn	75
90%		1.64 s	2	1.62 s	2	1.61 s	2	2.4 s	3	1 mn 41 s	125
100%		4.08 s	5	4.05 s	5	4.07 s	5	10.45 s	13	1 mn 43 s	141
<i>LDPC 500</i>	<i>(202,500)</i>										
50%		-	-	1.9 s	1	3.62 s	2	9.8 s	6	10 j 2 h	607 944
90%		5.89 s	3	5.56 s	3	7.24 s	4	16.28	10	ND	ND
100%		15.6 s	8	24.17 s	13	25.3 s	14	19.57 s	12	ND	ND
<i>LDPC 1000</i>	<i>(402,1000)</i>										
50%		10.18 s	2	11.97 s	3	20.53 s	6	11 j 31 mn	316 633	ND	ND
90%		25.84 s	5	28.22 s	7	34.35 s	10	ND	ND	ND	ND
100%		4 mn 10 s	48	3 mn 59 s	56	3 mn 26 s	61	ND	ND	ND	ND

<i>Codes</i>	(k, n)	TEB 10^{-3}		TEB 2.10^{-3}		TEB 5.10^{-3}		TEB 10^{-2}		TEB 2.10^{-2}	
		<i>temps</i>	<i>nb ité.</i>	<i>temps</i>	<i>nb ité.</i>	<i>temps</i>	<i>nb ité.</i>	<i>temps</i>	<i>nb ité.</i>	<i>temps</i>	<i>nb ité.</i>
<i>code aléatoire</i>	<i>(50,100)</i>										
25%		-	-	-	-	-	-	10.13 s	31	ND	ND
50%		-	-	0.31 s	1	0.65 s	2	25.29 s	77	ND	ND
90%		0.64 s	2	0.62 s	2	1.32 s	4	52.05 s	159	ND	ND
100%		0.95 s	3	1.25 s	4	1.63 s	5	54.63 s	167	ND	ND

TAB. 2.2 – Reconstruction de codes LDPC ($\gamma = 0.2$) et d'un code aléatoire ($\gamma = 0.15$)

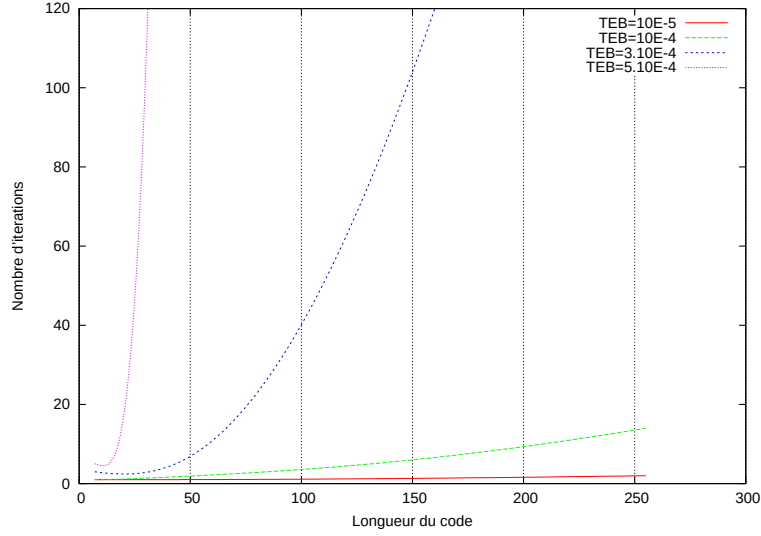


FIG. 2.17 – Nombre d'itérations en fonction de la longueur de codes cycliques

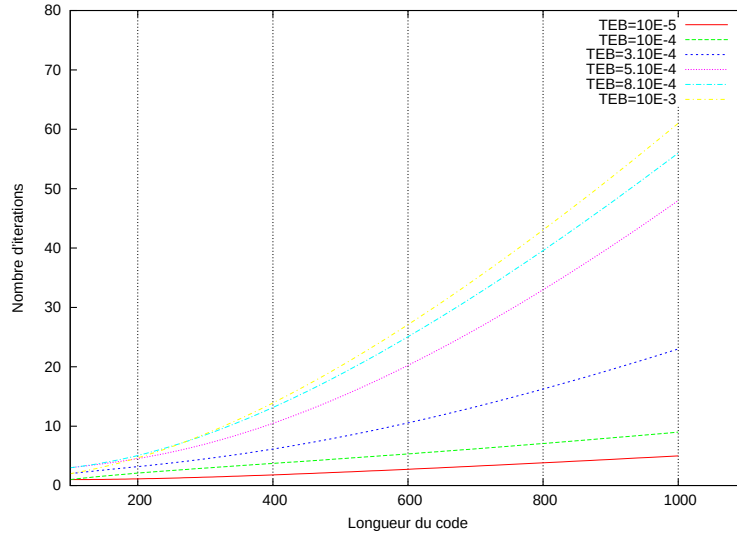


FIG. 2.18 – Nombre d'itérations en fonction de la longueur de codes LDPC

La complexité de tels algorithmes, pour des canaux binaires avec de l'erreur, est exponentielle en le poids de Hamming de la relation de parité de plus fort poids, mais aussi en la taille du code et l'erreur du canal. La taille des entrelaceurs est donc un facteur limitant capital. De plus, l'exposant de l'erreur est le produit de cette taille par le poids maximum pris sur l'ensemble des relations de parité. La seule façon de gagner un peu sur ces facteurs est de réussir à faire baisser artificiellement l'erreur du canal. Un des axes qui semble prometteur est celui proposé par M. Cluzeau [51] ; il consiste à essayer de corriger l'erreur en cours de reconstruction en utilisant des techniques de décodage itératif.

Les résultats obtenus vont dans le sens des hypothèses intuitives énoncées au paragraphe 2.1.2 et confortent aussi le choix du canal binaire symétrique. Notamment, plus la matrice d'erreur est dense, plus il est difficile de détecter une relation de parité. Augmenter le nombre d'itérations de l'algorithme de Gauss randomisé ou le nombre de bits interceptés

augmente la probabilité de détection. L'algorithme de Gauss randomisé permet même de retrouver des relations de parité qui n'apparaissent pas dans le noyau de $\tilde{C}_1$. Aux vues de l'impact de la répartition de l'erreur dans la matrice d'interception $\tilde{C}$, le canal binaire symétrique est le plus contraignant pour l'attaquant. En effet, l'erreur étant uniformément répartie sur toutes les lignes de la matrice $\tilde{C}$, aucune n'est à privilégier. En revanche, pour des erreurs en paquets, les lignes avec très peu d'erreur permettent de retrouver quasiment toutes les relations de parité en très peu d'itérations.

Plusieurs améliorations sont possibles et à prendre en compte lors de l'implémentation des algorithmes. La première permet d'éliminer des fausses alarmes en comptant la fréquence d'apparition des relations validées par l'algorithme de Gauss randomisé et en la comparant à celle attendue théoriquement quand on connaît l'erreur du canal. Dans le cas contraire, les fréquences d'apparition nous donnent une estimation de l'erreur introduite par le canal. Dans le cas de canaux sans erreur, une randomisation de l'algorithme, même avec très peu d'itérations permet de faire chuter drastiquement la probabilité de fausse alarme. D'autre part, on peut prévoir une adaptation automatique de l'algorithme en augmentant le nombre d'itérations l de l'algorithme de Gauss randomisé lorsque l'on a estimé n puis d , afin de concentrer nos efforts.

Enfin, l'étude des relations de parité doit nous permettre d'obtenir un peu d'information sur la nature de l'entrelaceur, notamment des couples d'ensembles d'antécédents et d'images par la permutation associée. Les algorithmes de reconstruction de codes en blocs linéaires ont été présentés dans le cadre de codes binaires. Les techniques mises en œuvre ne sont pas restreintes à $GF(2)$ et peuvent se généraliser naturellement à $GF(q)$. Dans ce cas, il reste néanmoins à étudier la modélisation et le comportement de l'erreur. En effet, dans $GF(2)$, celle-ci se compense sous certaines hypothèses que nous avons développées (parité et positionnement), il faut encore trouver l'équivalent dans $GF(q)$. Néanmoins, les algorithmes de reconstruction présentés restent valables dans $GF(q)$ mais avec une probabilité de succès restant à évaluer.

Reconstruction des codes convolutifs

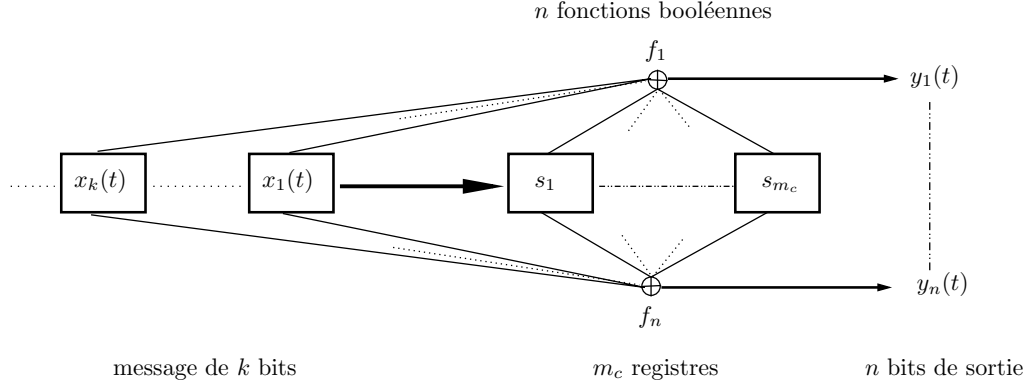
« Découvrir c’est bien souvent dévoiler
quelque chose qui a toujours été là, mais que
l’habitude cachait à nos regards. »

Arthur Koestler (*Le cri d’Archimède*)

Sommaire

3.1	Étude algébrique	89
3.1.1	Retrouver les mineurs d’ordre k	90
3.1.2	Déterminer une matrice génératrice	93
3.2	Algorithmes de reconstruction	94
3.2.1	Canal sans erreur	95
3.2.2	Canal avec erreur	104
3.3	Analyse des algorithmes	105
3.3.1	Canal sans erreur	105
3.3.2	Canal avec erreur	108
3.4	Résultats expérimentaux	109
3.5	Conclusion et perspectives	112

LES codes convolutifs ont été largement étudiés dans le chapitre 1 ainsi que leur représentation algébrique. Dans ce chapitre, nous présentons des algorithmes de reconstruction d’un code convolutif dans le même esprit que ceux présentés dans le chapitre 2. En effet, nous utilisons l’algorithme de Gauss randomisé (cf. paragraphe 2.1.2 page 57) et tirons avantageusement parti de la structure algébrique forte des codes convolutifs pour obtenir plus d’équations afin de retrouver les relations de parité. D’autre part, la représentation algébrique des codes convolutifs introduite dans le chapitre 1, nous permet tout d’abord d’exprimer de façon simple des relations entre les bits des mots de code, puis de bénéficier ensuite de la puissance des outils algébriques. Dans un premier temps, nous exprimons le problème de reconstruction sous forme algébrique. Soit à reconstruire un (n, k, m) -code convolutif $\mathcal{C}$. À chaque « top » d’horloge t , k bits d’information $x_1(t) \dots x_k(t)$ entrent dans les m_c registres du codeur et n bits codés $y_1(t) \dots y_n(t)$ sont renvoyés par les n fonctions linéaires $f_1 \dots f_n$ comme l’illustre la figure 3.1.

FIG. 3.1 – (n, k, m_c) -codeur convolutif

En utilisant le formalisme introduit au chapitre 1, nous pouvons écrire une équation liant les bits de sortie aux bits en entrée et aux fonctions booléennes,

$$Y(D) = G(D)X(D), \quad (3.1)$$

de la même forme que l'équation (1.6) page 29 (la seule différence réside dans l'utilisation de la notation matricielle française). Nous nous plaçons dans le canal binaire symétrique. Nous notons $\tilde{Y}(D)$, la sortie du codeur bruitée par le canal,

$$\tilde{Y}(D) = Y(D) \oplus E(D),$$

où les $E_i(D) = \sum_{j \geq 0} e_i(j)D^j$, pour $i = 1 \dots n$, sont des séries formelles à coefficients dans $F = GF(2)$. Dans le cas d'un canal binaire symétrique les $e_i(j)$ sont des variables aléatoires indépendantes à valeurs dans F et suivent une loi uniforme de paramètre ε , le taux d'erreur binaire (TEB) du canal, *i.e.*

$$\mathcal{Pr}(e_i(j) = 1) = \varepsilon \quad \text{et} \quad \mathcal{Pr}(e_i(j) = 0) = 1 - \varepsilon.$$

Nous faisons l'hypothèse raisonnable que la matrice du codeur utilisé n'est pas catastrophique ; un code correcteur d'erreurs ayant plutôt vocation à corriger les erreurs qu'à les propager. D'autre part, nous supposons que le codeur est non récursif et non poinçonné ; c'est-à-dire que sa matrice génératrice $G(D)$ est polynomiale. De plus, nous considérons ici un schéma de codage seul, sans entrelaceur. Nous expliquons comment traiter ces cas particuliers au paragraphe 3.5.

Le problème de la reconstruction des codes convolutifs a été posé et partiellement résolu par B. Rice [162] en 1995. Celui-ci s'est essentiellement intéressé à la reconstruction des codes convolutifs de rendement $\frac{1}{n}$ pour un canal non bruité. Un peu plus tard, É. Filiol a mis en évidence une technique permettant de reconstruire des codes convolutifs quelconques pour un canal bruité. Nous avons affiné la méthode d'É. Filiol et amélioré notablement la complexité et ainsi que le nombre de bits interceptés nécessaires au processus de reconstruction. Dans le cas d'un canal sans erreur, paragraphe 3.2.1, nous améliorons tout d'abord les résultats de B. Rice pour les codes convolutifs de rendement $\frac{1}{n}$ en utilisant l'algorithme de Berlekamp-Massey [28, 140] puis, nous améliorons l'algorithme d'É. Filiol en mettant en évidence une structure particulière dans ses équations. Dans le cas d'un

canal avec erreur, paragraphe 3.2.2, nous utilisons l'algorithme de Gauss randomisé (cf. paragraphe 2.1.2 page 57) afin de relaxer les contraintes imposées par la méthode d'É. Filiol et s'autoriser sous certaines hypothèses de l'erreur dans la matrice interceptée. De ce fait, nous expliquons pourquoi les résultats expérimentaux d'É. Filiol sont meilleurs que ceux qu'il annonce en théorie. De plus, la simplification des équations dans le cas sans erreur se généralise dans le cas d'un canal bruité. Nous effectuons ensuite l'analyse des algorithmes ainsi détaillés au paragraphe 3.3 et démontrons la conjecture d'É. Filiol [71, p. 170]. Nous présentons ensuite au paragraphe 3.4 les résultats expérimentaux que nous avons obtenus. Enfin, au paragraphe 3.5, nous expliquons comment traiter le cas des codes convolutifs récurrents, poinçonnés et des schémas de codage suivis d'un entrelaceur et proposons de nouveaux angles d'attaque pour améliorer encore plus la complexité du processus de reconstruction des codes convolutifs. Le résultat de ces travaux ont été publiés dans [9, 19].

3.1 Étude algébrique

Dans ce paragraphe, nous cherchons des relations liant les bits de sortie d'un codeur convolutif et les polynômes constituant sa matrice génératrice $G(D)$. Pour ce faire, nous procédons en deux étapes successives. Nous construisons tout d'abord au paragraphe 3.1.1, à partir de la définition d'un codeur convolutif, $(n - k)$ systèmes d'équations qui lient les bits de sortie, les polynômes de $G(D)$ et les mots d'information. Pour chaque système, nous éliminons les mots d'information et mettons ensuite en évidence une relation matricielle entre les bits de sortie ($Y_i(D)$) et un certain nombre de mineurs $\Delta_j(D)$ d'ordre k de sous-matrices de $G(D)$. L'équation (3.5) page 92 montre que ces mineurs appartiennent au noyau d'une matrice définie à l'aide des ($Y_i(D)$). De plus, les systèmes d'équations sont construits de façon à posséder un mineur en commun afin de retrouver les bonnes solutions dans les noyaux. Par définition, ces mineurs sont des sommes de produits de k polynômes de $G(D)$; la seconde étape consiste donc à trouver des relations entre ces mineurs d'ordre k et les polynômes de la matrice génératrice. Ces relations sont mises en évidence au paragraphe 3.1.2. Cette étape utilise exactement les mêmes techniques que la première mais avec des entrées différentes; des séries constituées par les bits de sortie ($Y_i(D)$), pour la première et des polynômes ($\Delta_j(D)$), pour la seconde. L'équation (3.7) page 94 montre que les polynômes de $G(D)$ appartiennent au noyau d'une matrice définie à l'aide des ($\Delta_j(D)$). Chaque système permet au final de retrouver $k + 1$ lignes de $G(D)$ dont les k premières lignes. La mise en commun de ces k lignes permet de retrouver les bonnes solutions dans les noyaux.

Seule la première étape fait intervenir les bits interceptés. Comme pour la reconstruction des codes en blocs linéaires, nous évaluons les noyaux de matrices construites à partir des bits interceptés. Dans le cas des codes convolutifs, les ($\Delta_j(D)$) jouent le rôle des relations de parité. La première étape est donc équivalente à reconstruction de $(n - k)$ codes en blocs linéaires dont la matrice d'interception est définie par la relation (3.4) page 92 et dont les relations de parité sont les ($\Delta_j(D)$). Dans le cas bruité, cette première étape est résolue en mettant en œuvre l'algorithme de reconstruction des codes en blocs linéaires dans le cas avec erreurs. La deuxième étape est, au contraire, une résolution de systèmes où l'erreur n'intervient plus et où tous les paramètres de taille sont fixés.

3.1.1 Retrouver les mineurs d'ordre k

L'objectif de ce paragraphe est de mettre en évidence une relation liant les $(Y_i(D))$ et les polynômes de la matrice génératrice du codeur. Pour cela, il faut revenir à la définition des codeurs convolutifs. L'équation (3.1) page 88 peut s'écrire sous forme développée

$$\begin{cases} X_1(D)G_{1,1}(D) + \dots + X_k(D)G_{1,k}(D) = Y_1(D), & (E_1) \\ \vdots & \vdots \\ X_1(D)G_{n,1}(D) + \dots + X_k(D)G_{n,k}(D) = Y_n(D). & (E_n) \end{cases} \quad (3.2)$$

Nous pouvons alors construire $(n - k)$ systèmes $(S_i)_{i=1 \dots (n-k)}$ à $(k + 1)$ équations. Le système (S_i) est composé des équations (E_j) pour $j = 1 \dots k$ et de l'équation (E_{k+i}) . À chaque système (S_i) , correspond un $(k + 1, k)$ -sous-codeur. Nous étudions (S_1) ; l'étude des autres systèmes étant strictement identique. (S_1) s'écrit alors

$$\begin{cases} X_1(D)G_{1,1}(D) + \dots + X_k(D)G_{1,k}(D) = Y_1(D), & (E_1) \\ \vdots & \vdots \\ X_1(D)G_{k+1,1}(D) + \dots + X_k(D)G_{k+1,k}(D) = Y_{k+1}(D). & (E_{k+1}) \end{cases}$$

Définissons $M_1(D)$ la matrice carrée de taille $(k + 1)$ à coefficients dans $F((D))$ par

$$M_1(D) = \begin{pmatrix} G_{1,1}(D) & \dots & G_{1,k}(D) & Y_1(D) \\ \vdots & & \vdots & \vdots \\ G_{k+1,1}(D) & \dots & G_{k+1,k}(D) & Y_{k+1}(D) \end{pmatrix},$$

et prouvons ensuite la proposition suivante.

Proposition 3.1 *Soit $\Delta_i^{(1)}(D)$ le mineur d'ordre k de $M_1(D)$ associé à $Y_i(D)$ (c'est-à-dire le déterminant de $M_1(D)$ privée de la $i^{\text{ème}}$ ligne et dernière colonne), alors*

$$\sum_{i=1}^{k+1} Y_i(D) \Delta_i^{(1)}(D) = 0.$$

Preuve :

Notons $M_1^{(i)}(D)$ la $i^{\text{ème}}$ colonne de $M_1(D)$. D'après la définition de (S_1) , nous avons

$$\sum_{i=1}^k X_i(D) M_1^{(i)}(D) + M_1^{(k+1)}(D) = 0.$$

Nous en déduisons que $\det M_1(D) = 0$. Nous développons ce déterminant suivant la $k^{\text{ième}}$ colonne de $M_1(D)$ et obtenons la formule de la proposition 3.1. ■

Remarque 3.1 :

Les inconnus du système sont les $\Delta_i^{(1)}(D)$ et ses coefficients sont les $Y_i(D)$.

Corollaire 3.1 *L'ensemble $\mathcal{D}$ des vecteurs $(P_1(D), \dots, P_{k+1}(D))$ qui vérifient la relation de la proposition 3.1 possède une structure de $F[D]$ -module.*

Preuve :

Cette proposition se montre aisément en revenant aux définitions et en utilisant l'équation de la proposition 3.1 page ci-contre. ■

Remarque 3.2 :

Cette structure de module été initialement mise en évidence par É. Filiol [71, chap. 8.2.3, p. 163] pour le cas de $(n, 1)$ -code convolutifs. Nous étendons son résultat au cas général d'un (n, k) -code convolutif.

D'autre part, les matrices génératrices polynomiales (MGP) engendrant le même code convolutif $\mathcal{C}$ sont $F[D]$ -équivalentes. Soit $G_0(D)$ une matrice canonique de $\mathcal{C}$ et $G(D)$ une MGP engendrant le même code. Soit $\delta_i^{(1)}$ le mineur d'ordre k de $M_1(D)$, construite avec $G_0(D)$, associé à $Y_i(D)$. Il existe donc $P(D) \in F[D]$ tel que $G(D) = P(D)G_0(D)$. Ceci implique

$$\sum_{i=1}^{k+1} Y_i(D)(P^k(D)\delta_i^{(1)}(D)) = 0.$$

On en déduit alors que les mineurs d'ordre k de toutes les MGP d'un même code convolutif appartiennent au même sous-module de longueur 1 du module des solutions de l'équation de la proposition 3.1 page précédente. De plus, il est aisé de montrer que ce sous-module de longueur 1 est engendré par le vecteur des $(k+1)$ mineurs d'ordre k , $\delta_i^{(1)}(D)$ de $G_0(D)$. Ce générateur est facilement calculable à partir de n'importe quel élément du sous-module.

Nous notons $\mathcal{G}$ l'application de $(F[D])^{k+1}$ dans $(F[D])^{K+1}$ qui, à un vecteur $P(D)$, associe le générateur du $F(D)$ -module de longueur 1 à qui il appartient.

$$\begin{aligned} \mathcal{G} : (F[D])^{k+1} &\longrightarrow (F[D])^{K+1} \\ P(D) &\longrightarrow \frac{1}{\text{PGCD}(P_1(D), \dots, P_{k+1}(D))} P(D). \end{aligned}$$

Notons, $d_0^{(1)}$ le degré interne du sous $(k+1, k)$ -codeur de $G_0(D)$ associé à (S_1) ($d_0^{(1)} = \max_{i=1 \dots k+1} \{\deg \delta_i^{(1)}(D)\}$) et $d \geq d_0^{(1)}$. Notons N le nombre de mots de code de longueur n interceptés. Sans perte de généralité, N est supposé constant. Nous faisons l'hypothèse raisonnable que $N > (k+1)d$. En développant l'équation de la proposition 3.1, nous avons

$$\begin{aligned} \sum_{i=0}^{k+1} \left(\left(\sum_{j=0}^N y_i(j) D^j \right) \left(\sum_{l=0}^d \Delta_i(l) D^l \right) \right) &= 0, \\ \sum_{i=0}^{k+1} \left(\sum_{j=0}^N \left(\sum_{l=0}^j y_i(j) \Delta_i(j-l) \right) D^j \right) &= 0, \\ \sum_{j=0}^N \left(\sum_{i=0}^{k+1} \sum_{l=0}^j y_i(j) \Delta_i(j-l) \right) D^j &= 0. \end{aligned}$$

Ceci implique que pour $\forall j \geq d$, le coefficient de D^j vaut 0. En effectuant une translation d'indices et en considérant que $\forall l > d, \Delta_i(l) = 0$,

$$\forall j \geq 0, \quad \sum_{i=1}^{k+1} \sum_{l=0}^d y_i(l+j) \Delta_i(d-l) = 0. \quad (3.3)$$

Si nous définissons le vecteur $\Delta^{(1)}$ par $(\Delta_1^{(1)}(0), \dots, \Delta_1^{(1)}(d), \dots, \Delta_{k+1}^{(1)}(0), \dots, \Delta_{k+1}^{(1)}(d))^T$ et $C^{(1)}(k, d)$ la matrice de taille $(N - d + 1) \times (k + 1)(d + 1)$ par

$$C^{(1)}(k, d) = \begin{pmatrix} y_1(d) & \dots & y_1(0) & \dots & y_{k+1}(d) & \dots & y_{k+1}(0) \\ y_1(d+1) & \dots & y_1(1) & \dots & y_{k+1}(d+1) & \dots & y_{k+1}(1) \\ \vdots & & \vdots & & \vdots & & \vdots \\ y_1(N) & \dots & y_1(N-d) & \dots & y_{k+1}(N) & \dots & y_{k+1}(N-d) \end{pmatrix}, \quad (3.4)$$

l'équation (3.3) page précédente, peut s'écrire sous forme matricielle,

$$C^{(1)}(k, d)\Delta^{(1)} = 0. \quad (3.5)$$

Remarque 3.3 :

Nous attirons le lecteur sur le fait que nous utilisons un isomorphisme implicite de $F^{(k+1)(d+1)}$ dans $(F^{(d)}[D])^{(k+1)}$, où $F^{(d)}[D]$ désigne l'ensemble des polynômes de degré inférieur ou égal à d et à coefficients dans F . Celui-ci associe au vecteur

$$\Delta^{(1)} = (\Delta_1^{(1)}(0), \dots, \Delta_1^{(1)}(d), \dots, \Delta_{k+1}^{(1)}(0), \dots, \Delta_{k+1}^{(1)}(d))^T$$

de $F^{(k+1)(d+1)}$, le vecteur

$$\Delta^{(1)}(D) = \left(\Delta_1^{(1)}(D) = \sum_{j=0}^d \Delta_1^{(1)}(j)D^j, \dots, \Delta_{k+1}^{(1)}(D) = \sum_{j=0}^d \Delta_{k+1}^{(1)}(j)D^j \right)^T$$

de $(F^{(d)}[D])^{(k+1)}$. Dans le reste du chapitre, nous utilisons l'une ou l'autre des notations en fonction du contexte.

À ce stade, l'équation (3.5) est de la même forme que celle obtenue pour la reconstruction des codes en blocs linéaires. La matrice d'interception est un peu différente, mais le problème à résoudre est identique; il consiste à évaluer le noyau de la matrice d'interception. Dans le cas des codes convolutifs, $\Delta^{(1)}$ fait office de relation de parité. D'autre part, pour lever toute ambiguïté, il convient de noter que le paramètre d prend ici une signification différente. Pour les codes en blocs linéaires, il représente le facteur de désynchronisation, pour les codes convolutifs, il désigne le degré d'un sous-codeur. Nous adoptons donc la même méthodologie que pour les codes en blocs.

Sans connaissance *a priori*, l'observateur doit deviner la longueur n des mots de code. Il doit alors faire une hypothèse n_e pour l'estimer. De même, il doit faire une hypothèse k_e et d_e sur k et d . D'autre part, nous ne considérons pas de désynchronisation pour la reconstruction des codes convolutifs. En effet, la relation de la proposition 3.1 page 90, reste vérifiée lorsque l'on multiplie les $Y_i(D)$ par $D^{-\alpha}$; ce qui correspond à un facteur de désynchronisation de α . L'observateur va alors construire la matrice $\tilde{C}^{(1)}(n_e, k_e, d_e)$ à partir des bits interceptés sur le canal. Une fois remplie, l'observateur va découper $\tilde{C}^{(1)}(n_e, k_e, d_e)$ en deux sous-matrices $\tilde{C}_2^{(1)}(n_e, k_e, d_e)$ et $\tilde{C}_1^{(1)}(n_e, k_e, d_e)$ de tailles respectives $((k_e + 1)(d_e + 1)) \times ((k_e + 1)(d_e + 1))$ et $(N - (k_e + 1)(d_e + 1)) \times ((k_e + 1)(d_e + 1))$

vérifiant

$$\begin{aligned}\tilde{C}_1^{(1)}(n_e, k_e, d_e) &= C_1^{(1)}(n_e, k_e, d_e) \oplus E_1^{(1)}(n_e, k_e, d_e), \\ \tilde{C}_2^{(1)}(n_e, k_e, d_e) &= C_2^{(1)}(n_e, k_e, d_e) \oplus E_2^{(1)}(n_e, k_e, d_e),\end{aligned}$$

où la matrice $\tilde{C}_1^{(1)}(n_e, k_e, d_e)$ étant la matrice carrée supérieure de $\tilde{C}^{(1)}(n_e, k_e, d_e)$. Nous pouvons alors exprimer le problème de reconstruction des codes convolutifs à partir d'un critère algébrique simple.

Proposition 3.2 Soient le vecteur $\delta^{(1)}(D) = (\delta_1^{(1)}(D), \dots, \delta_{k+1}^{(1)}(D))$, $d \geq d_0^{(1)}$ et $l = (d - d_0^{(1)})$. Sous l'hypothèse que $(k+1)(d+1) - (l+1)$ lignes de $(C_1^{(1)}(n, k, d))$ soient libres,

$$\text{Ker } (C_1^{(1)}(n, k, d)) = \{\delta^{(1)}(D)P(D) \mid P(D) \in F^{(l)}[D]\},$$

où $F^{(l)}[D]$ est l'ensemble des polynômes de degré inférieur ou égal à l et à coefficients dans F . $\text{Ker } (C_1^{(1)}(n, k, d))$ est un $F^{(l)}[D]$ -module de longueur 1 engendré par $\delta^{(1)}(D)$.

Preuve :

Application directe du corollaire 3.1 page 90. L'hypothèse que $(k+1)(d+1) - (l+1)$ lignes de $(C_1^{(1)}(n, k, d))$ soient libres est raisonnable dans la mesure où les mots d'information envoyés sont en général issus d'une compression, d'un chiffrement puis de la sortie d'un brasseur. ■

Le problème de reconstruction des codes convolutifs est donc équivalent à retrouver les $\text{Ker } (C_1^{(j)}(n, k, d))$ en observant $\tilde{C}^{(j)}(n_e, k_e, d_e)$, pour $j = 1 \dots (n - k)$. Il est à noter que les vecteurs $\Delta^{(j)}(D)$, solutions des systèmes (S_j) ont leur dernière coordonnée identique, soit

$$\forall j = 1 \dots (n - k), \quad \Delta_{k+1}^{(j)}(D) = \det \begin{pmatrix} G_{1,1}(D) & \dots & G_{1,k}(D) \\ \vdots & & \vdots \\ G_{k,1}(D) & \dots & G_{k,k}(D) \end{pmatrix}.$$

Cette coordonnée commune à toutes les solutions des $\mathcal{S}_j$ permet de distinguer dans l'évaluation des différents noyaux, les solutions valides des autres.

3.1.2 Déterminer une matrice génératrice

Nous avons trouvé au paragraphe précédent une relation entre les $(Y_i(d))$ et les $\Delta^{(j)}$. En appliquant exactement la même approche, nous mettons maintenant en évidence une relation entre les $\Delta^{(j)}$ et les polynômes de $G(D)$. Pour remonter aux polynômes de la matrice génératrice, nous définissons les k matrices $L_1^{(j)}(D)$ suivantes

$$\forall j = 1 \dots k, \quad L_1^{(j)}(D) = \begin{pmatrix} G_{1,1}(D) & \dots & G_{1,k}(D) & G_{j,1}(D) \\ \vdots & & \vdots & \vdots \\ G_{k+1,1}(D) & \dots & G_{k+1,k}(D) & G_{j,k+1}(D) \end{pmatrix}.$$

La dernière colonne de $L_1^{(j)}(D)$ étant identique à sa $j^{\text{ième}}$, $\det L_1^{(j)}(D) = 0$. En le développant suivant la dernière colonne nous obtenons

$$\forall j = 1 \dots k, \quad \sum_{i=0}^{k+1} \Delta_i^{(1)}(D) G_{i,j}(D) = 0. \quad (3.6)$$

Remarque 3.4 :

Les inconnus du système sont les $G_{i,j}(D)$ et les coefficients du système sont les $\Delta_i^{(1)}(D)$. Soit $l = \deg G(D)$ la mémoire du codeur (cf. définition 1.8 page 32), le vecteur $\Gamma_j^{(1)}$ défini par $(G_{1,j}(0), \dots, G_{1,j}(l), \dots, G_{k+1,j}(0), \dots, G_{k+1,j}(l))$ et la matrice carrée $F^{(1)}(k, l)$ de taille $(k+1)(l+1)$ définie par

$$\begin{pmatrix} \Delta_1^{(1)}(l) & \dots & \Delta_1^{(1)}(0) & \dots & \Delta_{k+1}^{(1)}(l) & \dots & \Delta_{k+1}^{(1)}(0) \\ \vdots & & \vdots & & \vdots & & \vdots \\ \Delta_1^{(1)}((k+2)(l+1)-2) & \dots & \Delta_1^{(1)}(0) & \dots & \Delta_{k+1}^{(1)}((k+2)(l+1)-2) & \dots & \Delta_{k+1}^{(1)}(0) \end{pmatrix},$$

avec $\forall i = 1 \dots (k+1)$, et $\forall j > d$, $\Delta_i^{(1)}(j) = 0$. La relation (3.6) page précédente s'exprime sous forme matricielle par

$$\forall j = 1 \dots k, F^{(1)}(k, l) \Gamma_j^{(1)} = 0. \quad (3.7)$$

Il en résulte que $\{\Gamma_j^{(1)} \mid j = 1 \dots k\} \subset \text{Ker } F^{(1)}(k, l)$. De plus, pour un j fixé, les vecteurs $\Gamma_j^{(i)}(D)$ ont leurs k premières coordonnées identiques, c'est-à-dire

$$(G_{1,j}(D), \dots, G_{k,j}(D))^T,$$

ce qui permet de retrouver entièrement la matrice génératrice $G(D)$. En effet, chaque S_p pour $p = 1 \dots (n-k)$, permet d'obtenir les coordonnées partielles de chaque colonne de la matrice génératrice. La résolution de S_p nous donne

$$(G_{1,j}(D), \dots, G_{k,j}(D), G_{k+p,j}(D))^T \quad \text{pour } j = 1 \dots k.$$

Les k colonnes de $G(D)$ sont donc entièrement retrouvées en résolvant tous les systèmes S_p . De plus, les k coordonnées en commun permet de sélectionner les candidats valides lors de l'évaluation des noyaux des $F^{(p)}(k, l)$.

3.2 Algorithmes de reconstruction

Il découle de l'étude précédente, des algorithmes naturels de reconstruction des codes convolutifs. Nous abordons tous d'abord le cas du canal sans erreur et proposons dans ce contexte une approche différente des algorithmes existants en utilisant l'algorithme de Berlekamp-Massey pour le cas particulier des $(n, 1)$ -codes convolutifs. Dans le cas général, nous procédons en trois étapes. La première consiste à détecter les bons paramètres n et k du code en évaluant le noyau de $C^{(1)}(k, d)$ définie par la relation (3.4) page 92 à l'aide des bits interceptés. Dans un deuxième temps, nous résolvons les $(n-k)$ sous-systèmes (S_j) en maintenant la dernière coordonnée identique pour chaque solution. À l'issue de cette étape, nous obtenons les $\Delta^{(j)}$. Enfin, lors de la troisième et dernière étape, nous retrouvons les polynômes de $G(D)$ à partir des $\Delta^{(j)}$ en résolvant l'équation (3.6) page précédente. Les candidats valides étant détectés grâce aux k premières coordonnées en commun.

Nous traitons ensuite le cas d'un canal bruité. Seule la première étape fait intervenir les bits interceptés. De plus, nous avons montré que la première étape est équivalente à la reconstruction d'un code en bloc linéaire. Nous effectuons dans le cas d'un canal bruité la première étape en remplaçant l'évaluation du noyau de $\tilde{C}^{(1)}(k, d)$ par celle de son noyau randomisé; les deux autres étant identiques.

3.2.1 Canal sans erreur

Nous nous intéressons tout d'abord au cas le simple, le cas d'un canal sans erreur. L'étude d'un tel canal se justifie par l'étude en boîte noire d'un équipement détenu par l'observateur. Nous présentons deux types d'algorithmes de reconstruction des codes convolutifs. Le premier est restreint aux cas particulier des $(n, 1)$ -codes convolutifs, le second fonctionnant dans le cas général.

Cas des $(n, 1)$ -codes convolutifs

Pour des $(n, 1)$ -codes convolutifs, le système (3.2) page 90 s'écrit

$$\begin{cases} X(D)G_{1,1}(D) = Y_1(D), & (E_1) \\ \vdots & \vdots \\ X(D)G_{n,1}(D) = Y_n(D). & (E_n) \end{cases}$$

Pour tout $j = 1 \dots (n - 1)$, les (S_j) peuvent s'écrire

$$(S_j) \quad Y_j(D)G'_{1,1}(D) = Y_1(D)G'_{j,1}(D),$$

où

$$G'_{j,1}(D) = \frac{1}{\text{PGCD}(G_{1,1}(D), G_{j,1}(D))}.$$

Définissons tout d'abord $Z_j(D)$ pour $j = 1 \dots n - 1$ par

$$Z_j(D) = \begin{cases} \frac{Y_1(D)}{Y_j(D)} & \text{si } v(Y_1(D)) \geq v(Y_j(D)), \\ \frac{Y_j(D)}{Y_1(D)} & \text{sinon.} \end{cases}$$

où $v(Y_j(D))$ représente la valuation de $Y_j(D)$. Résolvons ensuite (S_1) ; la résolutions des (S_j) s'effectue de manière similaire. Supposons maintenant que $v(Y_1(D)) \geq v(Y_2(D))$ (un raisonnement symétrique pourra être mené dans le cas contraire). (S_1) peut se mettre sous la forme

$$G'_{1,1}(D) = G'_{2,1}(D)Z_1(D).$$

$G'_{2,1}(D)$ étant de degré minimal, l'algorithme de Berlekamp-Massey [28, 140] appliqué à $Z_1(D)$ permet de le retrouver. Connaissant $G'_{2,1}(D)$, il ne reste plus qu'à évaluer $G'_{1,1}(D)$. Puis de proche en proche, en résolvant les (S_j) , les $G'_{j,1}(D)$, pour $j = 3 \dots n$, sont déterminés en calculant

$$G'_{j,1}(D) = G'_{1,1}(D) \frac{Y_j(D)}{Y_1(D)}.$$

Notons $d_0 = \max_j \{\deg G'_{j,1}(D)\}$. Sans connaissance *a priori* de n et d_0 , nous devons donc faire des hypothèses n_e, d_{max} sur n et un majorant de d_0 . Nous appliquons ensuite l'algorithme de Berlekamp-Massey (BM) pour tous les (S_j) et vérifions pour chacun d'eux que le polynôme $G'_{1,1}(D)$ est identique afin de valider ou d'invalidier ces hypothèses. En pratique, nous fixons une borne n_{max} pour n_e . Les $Z_j(D)$ seront donc tronqués à l'ordre $2d_{max}$. Il en résulte un algorithme naturel de reconstruction des $(n, 1)$ -codes convolutifs dans le cas sans erreur, résumé dans la figure 3.2.

**Algorithme de reconstruction des $(1, n)$ -codes convolutifs
dans le cas d'un canal sans erreur**

Entrées : (y_i) les bits interceptés,
 n_{max} le paramètre d'arrêt,
 d_{max} borne sur le degré de la matrice génératrice.

Sortie : une matrice génératrice de $\mathcal{C}$ qui a généré les (y_i) ou $\emptyset$.

```

1  Pour  $n$  de 2 à  $n_{max}$ 
2     $G(D) = (G_1(D), \dots, G_n(D)) \leftarrow 0$ 
3    Calculer  $Z_2(D)$  à l'ordre  $2d_{max}$ 
4     $(G_1(D), G_2(D)) \leftarrow \text{BM}(Z_2(D))$ 
5    Pour tout  $i$  de 3 à  $n$ 
6      Calculer  $Z_i(D)$  à l'ordre  $2d_{max}$ 
7       $(P_1(D), P_2(D)) \leftarrow \text{BM}(Z_i(D))$ 
8      Si  $P_1(D) \neq G_1(D)$  retour à l'étape 1
9      Sinon  $G_i(D) \leftarrow P_2(D)$ 
10   Fin Si
11   Fin Pour
12   Renvoyer  $G(D)$ 
13 Fin Pour
14 Renvoyer  $\emptyset$ 

```

FIG. 3.2 – Algorithme de reconstruction des $(n, 1)$ -code convolutifs dans un canal sans erreur

Cas général

Dans le cas général d'un (n, k, m) -code convolutif, nous utilisons les relations mises en évidence au chapitre 3.1 et procédons en trois étapes. La première étape consiste à détecter les bonnes valeurs des paramètres n , k et m , la deuxième à retrouver les mineurs d'une matrice génératrice canonique du code en résolvant tous les sous-systèmes et enfin la troisième à retrouver une matrice génératrice à partir des mineurs. Trois paramètres sont à déterminer en aveugle, n , k et d , où d est le degré d'une matrice génératrice du (n, k) -code convolutif. Nous faisons les hypothèses n_e , k_e et d_e sur chacun de ces paramètres. Nous testons des valeurs croissantes de n_e de 1 à n_{max} ; puis pour chaque n_e , les valeurs de k de 1 à $(n_e - 1)$. Enfin, pour chaque couple (n_e, k_e) nous essayons tous les degrés de 1 à d_{max} , du plus petit au plus grand, pour retrouver préférentiellement une matrice génératrice du plus bas degré possible, donc de degré externe le plus petit possible. L'objectif étant de retrouver une matrice canonique du code.

Pour valider ou invalider ces hypothèses, nous observons des comportements différents des matrices $C^{(j)}(n_e, k_e, d_e)$. En effet, d'après la relation (3.5) page 92, les $\Delta^{(i)}$ pour $i = 1 \dots (n - k)$ peuvent être vues comme des relations de parité c'est-à-dire des rela-

tions entre les colonnes des matrices $C^{(j)}(n, k, d)$. Si $n_e \neq n$ ou $k_e \neq k$, les $C^{(j)}(n_e, k_e, d_e)$ peuvent alors être considérées comme des matrices binaires aléatoires, comme l'illustre la figure 2.4. Avec forte probabilité, elles sont de rang maximum. Soit d_0 la mémoire du codeur d'une matrice génératrice canonique. Si $d_e < d_0$ alors il existe au moins un des systèmes (S_j) pour lequel la matrice $C^{(j)}(n_e, k_e, d_e)$ se comportera comme une matrice aléatoire. En revanche, lorsque $d_0 \leq d_e$, nous sommes dans les conditions de la proposition 3.2 page 93.

Il en découle un algorithme naturel de reconstruction des (n, k) -codes convolutifs. Pour chaque triplet (n_e, k_e, d_e) , nous résolvons (S_1) et cherchons des solutions particulières : celles qui constituent une base du $F[D]$ -module des solutions de la relation de la proposition 3.5 page 92. Pour ce faire, il faut évaluer le noyau de $C^{(1)}(n_e, k_e, d_e)$ et, pour chacun de ses éléments, calculer le générateur du module de longueur 1 auquel il appartient. Pour calculer ce générateur, nous appliquons la fonction $\mathcal{G}$ définie au paragraphe 3.1.1 page 91.

Dans un deuxième temps, nous évaluons de la même manière les noyaux des matrices $C^{(j)}(n_e, k_e, d_e)$ et cherchons une base des solutions de la relation de la proposition 3.5. Enfin, pour construire une solution à notre système initial, nous regroupons ensemble les $(n - k)$ bases de solutions des sous-systèmes (S_j) . Il nous faut ensuite trouver un élément dans chaque module de solutions des (S_j) de $(k + 1)^{\text{ième}}$ composante identique. Nous élargissons pour cela la notion de *consistance* introduite par É. Filiol [71, chap. 8.2.4, p.167] en généralisant la coïncidence sur les polynômes commun à leur appartenance à un même module. Cette notion permet de construire un générateur du module de longueur 1 des mineurs d'ordre k des matrices génératrices engendrant le même code convolutif, à partir des générateurs des modules de $(F[D])^{(k+1)}$ de solutions des systèmes (S_j) .

Exemple 3.1

Pour bien comprendre ce qu'il se passe, considérons la résolution d'un $(3, 1)$ -codeur pour laquelle les bons paramètres ont été devinés. Supposons que (S_1) renvoie

$$(\Delta_1(D), \Delta_2(D)) \text{ avec } \text{PGCD}(\Delta_1(D), \Delta_2(D)) = 1.$$

Deux cas peuvent se présenter : soit (S_2) renvoie

$$(\Delta_1(D), \Delta_3(D)) \text{ avec } \text{PGCD}(\Delta_1(D), \Delta_3(D)) = 1,$$

et dans ce cas

$$(\Delta_1(D), \Delta_2(D), \Delta_3(D)) \text{ avec } \text{PGCD}(\Delta_1(D), \Delta_2(D), \Delta_3(D)) = 1$$

est solution. Soit (S_2) renvoie

$$(Q(D)\Delta_1(D), \Delta_3(D)) \text{ avec } \text{PGCD}(Q(D)\Delta_1(D), \Delta_3(D)) = 1,$$

et dans ce cas

$$(Q(D)\Delta_1(D), Q(D)\Delta_2(D), \Delta_3(D)) \text{ avec } \text{PGCD}(Q(D)\Delta_1(D), Q(D)\Delta_2(D), \Delta_3(D)) = 1$$

est solution.

Nous pouvons faire exactement le même raisonnement si (S_1) renvoie

$$(Q(D)\Delta_1(D), \Delta_2(D)) \text{ avec } \text{PGCD}(Q(D)\Delta_1(D), \Delta_2(D)) = 1.$$

Définition 3.1 Soient $U(D) \in (F[D])^p$ et $V(D) \in (F[D])^q$ deux vecteurs à coefficients dans $F[D]$. Soit $\mathcal{E}$ un ensemble d'éléments de $[1, \min(p, q)]$. $U(D)$ et $V(D)$ sont consistants en les éléments de $\mathcal{E}$ si et seulement si il existe $P(D) \in F[D]$ tel que

$$\forall i \in \mathcal{E}, \quad U_i(D) = P(D)V_i(D),$$

ou

$$\forall i \in \mathcal{E}, \quad V_i(D) = P(D)U_i(D).$$

$P(D)$ est alors appelé facteur de consistance.

La notion de consistance permet de gérer les deux cas de l'exemple précédent et de construire un générateur de l'ensemble de solutions à partir des générateurs de solutions des systèmes (S_j) .

Nous cherchons tout d'abord les couples solutions de (S_1) et (S_2) consistants en leur $(k+1)^{\text{ième}}$ coordonnée, puis parmi ceux-là, ceux qui le sont avec les solutions de (S_3) et ainsi de suite. Parmi les couples d'éléments consistants trouvés, nous multiplions celui dont la $(k+1)^{\text{ième}}$ composante est de plus bas degré par le facteur de consistance. Les éléments du nouveau couple possèdent alors leur $(k+1)^{\text{ième}}$ coordonnée identique. L'algorithme de la figure 3.3 page suivante, garantit la consistance des solutions de notre systèmes d'équations et construit de proche en proche une liste de $(n-k)$ vecteurs $\Delta^{(j)}(D)$ de $(F[D])^{k+1}$ tels que $\forall i, j \in [0, n]$, $\Delta_{k+1}^{(i)}(D) = \Delta_{k+1}^{(j)}(D)$ et $\Delta^{(i)}(D)$ solution de (S_i) . Il est clair que l'ensemble listes de vecteurs vérifiant cette propriété est un $F[D]$ -module. L'algorithme de la figure 3.4 nous retourne une base de ce module (nous montrons au paragraphe 3.3.1 que l'algorithme renvoie bien une base).

Enfin, la troisième étape de l'algorithme de reconstruction consiste à retrouver des matrices génératrices pouvant générer les (y_i) , à partir des vecteurs $\Delta^{(j)}(D)$ calculés lors de l'étape précédente. La relation (3.6) page 93, est de la même forme que celle de la proposition 3.5, en remplaçant les $Y_i(D)$ par les $\Delta_i^{(j)}(D)$. De même, les matrices $F^{(j)}(k, l)$ sont de forme identique aux $C^{(j)}(n_e, k_e, d_e)$. Les colonnes de $G(D)$, d'une matrice génératrice de $\mathcal{C}$ ne sont pas $F[D]$ -équivalentes, il faut donc cette fois-ci considérer tous les éléments du noyau et pas seulement les générateurs des modules de longueur 1 auxquels ils appartiennent. Chaque (S_j) , pour $j = 1 \dots (n-k)$, fournit un vecteur de $(F[D])^{k+1}$, $\Delta^{(j)}(D) = (\Delta_1^{(j)}(D), \dots, \Delta_{k+1}^{(j)}(D))$. En résolvant les systèmes 3.7 construits à partir de chacun de ces vecteurs $\Delta^{(j)}(D)$, nous obtenons des colonnes partielles candidates, $(G_{1,l}(D), \dots, G_{k,l}(D), G_{k+j,l}(D))^T$, $l \in [1, k]$. De plus, les colonnes partielles candidates issues de $\Delta^{(1)}(D)$ et celles issues de $\Delta^{(j)}(D)$ ont leur k premières coordonnées identiques. Nous choisissons les couples de colonnes partielles candidates issues de $\Delta^{(1)}(D)$ et de $\Delta^{(j)}(D)$, consistants sur leur k premières composantes. Pour chaque couple, nous multiplions par le facteur de consistance la colonne partielle dont les k premières composantes sont de plus bas degré et construisons ainsi, de proche en proche, une colonne candidate complète. L'algorithme de la figure 3.4 page 100, avec $\mathcal{F} = I_d$, nous renvoie un ensemble de listes de $(n-k)$ vecteurs de $(F[d])^{k+1}$ correspondants aux colonnes partielles. La donnée d'une liste permet de reconstituer une colonne complète en prenant le premier

Procédure de vérification de la consistance des solutions

Entrées : un ensemble $\mathcal{L}$ de listes de $(n - k)$ vecteurs de $(F[D])^{k+1}$,
 un ensemble P de vecteurs de $(F[D])^{k+1}$,
 un ensemble $\mathcal{E}$ d'éléments de $[1, k + 1]$,
 $j \in \mathbb{N}^*$ tel que $0 < j \leq (n - k)$.

Sortie : un ensemble de listes de $(n - k)$ vecteurs de $(F[D])^{k+1}$.

```

1   $Out \leftarrow \emptyset$ 
2  Pour  $i$  de 1 à  $\text{Card}(P)$ 
3    Pour  $l$  de 1 à  $\text{Card}(\mathcal{L})$ 
4      Si  $P^{(i)}(D)$  et  $L^{(l)}(D)$  sont consistants en  $\mathcal{E}$  de facteur  $Q(D)$ 
5        Marquer  $L^{(l)}(D)$ 
6        Si  $\deg L_{\mathcal{E}(1)}^{(l)}(D) \geq \deg P_{\mathcal{E}(1)}^{(i)}(D)$ 
7           $L_j^{(l)}(D) \leftarrow Q(D)P^{(i)}(D)$ 
8        Sinon
9          Pour  $m$  de 1 à  $j - 1$ 
10            $L_m^{(l)}(D) \leftarrow Q(D)L_m^{(l)}(D)$ 
11         Fin Pour
12          $L_j^{(l)}(D) \leftarrow P^{(i)}(D)$ 
13       Fin Si
14     Fin Si
15   Fin Pour
16 Fin Pour
17  $Out \leftarrow \{L^{(i)}(D) \mid L^{(i)}(D) \text{ marqué} \}$ 
18 Renvoyer  $Out$ 

```

FIG. 3.3 – Procédure de vérification de la consistance des solutions

vecteur de la liste pour les $(k + 1)$ premières coordonnées de la colonne et la dernière composante de chaque vecteur restant de la liste pour les coordonnées suivantes. Enfin, pour chaque ensemble de k colonnes candidates complètes, nous construisons une matrice $n \times k$ et vérifions que celle-ci est basique ; si c'est le cas elle peut avoir généré les (y_i) . Une optimisation évidente est de faire rentrer la deuxième et troisième étape dans la recherche des bons paramètres. En effet, cela implique dans ce cas, que pour que les paramètres n et k soient validés, il faut que la sortie des étapes deux et trois soient non vides. Il en résulte alors une chute drastique du nombre de fausses alarmes sur le choix de n et k . L'algorithme complet de reconstruction des (n, k) -codes convolutifs est résumé en figure 3.5 page 101.

Recherche de solutions particulières
du système $\left(\sum_{i=1}^k Y_i(D)P_i(D) + Y_{k+j}(D)P_{k+j}(D) = 0\right)_{j=1\dots n-k}$
- Cas sans erreur -

Entrées : $Y(D) \in (F[D])^n$,
un ensemble $\mathcal{E}$ d'éléments de $[1, k+1]$,
 $\mathcal{F}$, fonction de $(F[D])^{k+1}$ de $(F[D])^{k+1}$,
 k, d_{min} et d_{max} .

Sortie : module de solutions du système.

```

1  Pour  $d$  de  $d_{min}$  à  $d_{max}$ 
2     $B \leftarrow \emptyset$  ensemble de listes de  $(n-k)$  vecteurs de  $(F[D])^{(k+1)}$ 
3     $P \leftarrow \{ P^{(i)}(D) | P^{(i)}(D) \in \mathcal{F}(\text{Ker}(C^{(1)}(n, k, d))) \}$ 
4     $B \leftarrow \{ [P_1^{(i)}(D), \dots, P_{k+1}^{(i)}(D)], 0, \dots, 0 \mid P^{(i)}(D) \in P \}$ 
5    Si  $B = \emptyset$  retour à l'étape 1
6    Fin Si
7    Pour  $j$  de 2 à  $n-k$ 
8       $P \leftarrow \{ P^{(i)}(D) | P^{(i)}(D) \in \mathcal{F}(\text{Ker}(C^{(j)}(n, k, d))) \}$ 
9       $B \leftarrow \text{Consistance}(B, P, \mathcal{E}, j)$ 
10     Si  $B = \emptyset$  retour à l'étape 1
11     Fin Si
12   Fin Pour
13   Renvoyer  $B$ 
14 Fin Pour
15 Renvoyer  $\emptyset$ 

```

FIG. 3.4 – Algorithme de résolution du système $(S_j)_{j=1\dots(n-k)}$ dans le cas sans erreur

Exemple 3.2

Soit à reconstruire le code convolutif engendré par la matrice génératrice

$$G(D) = \begin{pmatrix} D^6 + D^4 + D^3 + D + 1 \\ D^6 + D^5 + D^2 + 1 \\ D^6 + D^5 + D^4 + D^3 + D^2 + 1 \end{pmatrix}.$$

Bien que l'algorithme BM soit plus efficace pour reconstruire des $(n, 1)$ -codes convolutifs, nous avons choisi un $(3, 1, 6)$ -codeur convolutif pour les besoins pédagogiques. Nous avons intercepté le message sans erreur de taille 400 bits, soit 50 octets, engendré par le codeur

**Algorithme de reconstruction des (n, k) -codes convolutifs
- Cas sans erreur -**

Entrées : (y_i) ensemble des bits interceptés,
 n_{max}, d_{max} .

Sortie : ensemble de matrices génératrices canoniques de degré inférieur à d_{max}
qui peuvent avoir généré les (y_i) .

```

1  Pour  $n$  de 2 à  $n_{max}$ 
2    Pour  $k$  de 1 à  $n - 1$ 
3       $S \leftarrow \emptyset$ 
4       $\Delta \leftarrow \text{Résoudre}(Y(D), k, 1, kd_{max}, \{k + 1\}, \mathcal{G})$ 
5      Si  $\Delta = \emptyset$  retour à l'étape 2
6      Fin Si
7      Pour  $i$  de 1 à  $\text{Card}(\Delta)$ 
8         $P \leftarrow \text{Résoudre}(\Delta^{(i)}(D), k, \lfloor \frac{\deg \Delta^{(i)}(D)}{k} \rfloor, \deg \Delta^{(i)}(D), \{1, \dots, k\}, \mathcal{I}_d)$ 
9        Si  $P = \emptyset$  retour à l'étape 7
10       Fin Si
11       Pour tout ensemble  $E^{(j)}$  de  $k$  colonnes complètes de  $P$ 
12         Construire  $G(D)$  une matrice avec les  $k$  colonnes complètes
13         dont les mineurs sont les  $\Delta^{(i)}(D)$ 
14          $S \leftarrow S \cup \{G(D)\}$ 
15       Fin Pour
16     Fin Pour
17     Si  $S \neq \emptyset$ 
18       Renvoyer  $S$ 
19     Fin si
20   Fin Pour
21 Renvoyer  $\emptyset$ 

```

FIG. 3.5 – Algorithme de reconstruction des (n, k) -codes convolutifs dans le cas sans erreur

$$Y = \begin{bmatrix} 1111001001100101010000101111011100110100001000001110010010000000110101 \\ 0010101101111000010010101011110010101011101100111000100101101000010011 \\ 1011101000001101001011000110101100001101011010101001001110000101110100 \\ 11010010100101010001110000110001001011010000100000111101010110101011 \\ 101011010100110100100100010110000010011110011110000101101110000111000 \\ 100110000100000100001100100110110101001110100110111100101010000000001 \\ 10001010100111001010010011100101110001111011011110110101001010111000 \\ 10011110011101101000111100001001011010110000111111011101011110111010 \\ 0001111101011001111010000001101110000111000101010010110111100111011101 \\ 00011011000111011111000101011010110101001101101000110010110111001101 \\ 0111011110011101101001100100011001101110000111000111000001010101100001 \\ 111100101111000100011100001011011011110001010110010001001110000010010 \\ 000111010100111101101110010110010110010011101011111001011011001 \\ 000000101001110101100010111011000001100100000100000101001010000110101 \\ 1010010101001110111011011110001101010110000000100001011011100001001 \\ 0100011101011111000100011111101101101010000000011000101011100100010011 \\ 101111101100011100001010010001011100110101111111100111001111110101010 \\ 101001101001010110111110100111 \end{bmatrix},$$

où les bits se lisent du plus récent au plus ancien en partant du haut à gauche vers le bas à droite. En faisant l'hypothèse que $n = 3$, nous découpons le train de bits en 3 pistes en prenant à chaque fois 1 bit sur trois.

$$Y_1 = \begin{bmatrix} 1111010011100010100000011001011011101001000001101001000111010010010111 \\ 111001010100000101110011110000000000111000101001000000011001011110000 \\ 101010111001010010011010010111010111000111011101100100011010100111101 \\ 011011100111011110000111111000001001010100100010010011110010001000 \\ 010111110101010011111110011010011010000000101000100111110101001000100 \\ 001111001111100010110001111010100110011101011010000111111 \end{bmatrix},$$

$$Y_2 = \begin{bmatrix} 1001100110111000100001100111000110011101001100110000010100001100010010 \\ 1000101001001100011011010011111010001010011001000000010000110001001000 \\ 010011001011011111100101111110110000101111101001010110001010011110110 \\ 0110111011001011010110101101111001110010101010110110010111110111001011 \\ 0110100111111110101000000101110100011010011000111011011110011000011000 \\ 101010101111000001011001111010000011111101110101111001101 \end{bmatrix},$$

$$Y_3 = \begin{bmatrix} 1000010011010000111001001110001010101110111000111011110010111011100100 \\ 00110010101110000101101011010001101110101110101101011111011111010001 \\ 0011000011101111110011001010001000111011010110011111100110010100010111 \\ 0001010100110000011110010101101100110101000110111101010000101000110000 \\ 0100111000000001011011000001001110000000100011100101100101100000111011 \\ 010110011000100100100110100100110101011110111010100111001 \end{bmatrix}.$$

Nous reconstruisons tout d'abord le premier sous-codeur dont les mineurs sont $(D^6 + D^5 + D^2 + 1, D^6 + D^4 + D^3 + D + 1)$. Supposons maintenant que nous avons fait la bonne hypothèse sur k mais pas sur d . Pour illustrer l'impact de la recherche d'une base des modules solutions, nous prenons $d_e = d + 2$ afin de ne pas avoir un noyau restreint à un unique vecteur. À l'aide des pistes Y_1 et Y_2 , nous remplissons $C^{(1)}(k, d + 2)$ d'après la relation (3.4) page 92.

$$C^{(1)}(k, d + 2) = \begin{pmatrix} 100101111 & 110011001 \\ 110010111 & 011001100 \\ 111001011 & 101100110 \\ 011100101 & 110110011 \\ 001110010 & 111011001 \\ 000111001 & 011101100 \\ 100011100 & 001110110 \\ 010001110 & 000111011 \\ 101000111 & 100011101 \\ 010100011 & 010001110 \\ 001010001 & 001000111 \\ 000101000 & 000100011 \\ 000010100 & 000010001 \\ 000001010 & 100001000 \\ 000000101 & 110000100 \\ 100000010 & 011000010 \\ 110000001 & 001100001 \\ 011000000 & 100110000 \\ \vdots & \vdots \end{pmatrix}.$$

En évaluant le noyaux de $C^{(1)}(k, d + 2)$ nous obtenons les solutions suivantes, exprimées sous leur forme binaire et polynomiale en utilisant l'isomorphisme explicité dans la remarque du paragraphe 3.1.1 page 92.

$$\begin{aligned}
(000000000000000000) &\rightarrow (0, 0), \\
(100011111111011001) &\rightarrow (D^8 + D^7 + D^6 + D^5 + D^4 + 1, D^8 + D^5 + D^4 + D^2 + D + 1), \\
(110111001100000011) &\rightarrow (D^8 + D^5 + D^4 + D^3 + D + 1, D^8 + D^7 + 1), \\
(010100110011011010) &\rightarrow (D^7 + D^6 + D^3 + D, D^7 + D^5 + D^4 + D^2 + D), \\
(011110101010110111) &\rightarrow (D^8 + D^6 + D^4 + D^3 + D^2 + D, D^8 + D^7 + D^6 + D^4 + D^3 + D), \\
(111101010101101110) &\rightarrow (D^7 + D^5 + D^3 + D^2 + D + 1, D^7 + D^6 + D^5 + D^3 + D^2 + 1), \\
(101001100110110100) &\rightarrow (D^6 + D^5 + D^2 + 1, D^6 + D^4 + D^3 + D + 1), \\
(001010011001101101) &\rightarrow (D^8 + D^7 + D^4 + D^2, D^8 + D^6 + D^5 + D^3 + D^2).
\end{aligned}$$

Pour chaque vecteur $P(D)$ non nul et solution, nous ne conservons comme candidat que le vecteur $P'(D) = \frac{P(D)}{\text{PGCD}(P_1(D), P_2(D))}$. Il ne reste plus que le vecteur

$$\Delta^{(1)}(D) = (D^6 + D^5 + D^2 + 1, D^6 + D^4 + D^3 + D + 1).$$

De la même manière nous remplissons $C^{(2)}(k, d+2)$ à l'aide de Y_1 et de Y_3 .

$$C^{(2)}(k, d+2) = \begin{pmatrix} 100101111 & 100100001 \\ 110010111 & 110010000 \\ 111001011 & 011001000 \\ 011100101 & 101100100 \\ 001110010 & 010110010 \\ 000111001 & 001011001 \\ 100011100 & 000101100 \\ 010001110 & 000010110 \\ 101000111 & 100001011 \\ 010100011 & 110000101 \\ 001010001 & 111000010 \\ 000101000 & 011100001 \\ 000010100 & 001110000 \\ 000001010 & 100111000 \\ 000000101 & 010011100 \\ 100000010 & 001001110 \\ 110000001 & 100100111 \\ 011000000 & 110010011 \\ \vdots & \vdots \end{pmatrix}.$$

En évaluant le noyaux de $C^{(1)}(k, d+2)$ nous obtenons les solutions suivantes, exprimées sous leur forme binaire et polynomiale.

$$\begin{aligned}
(000000000000000000) &\rightarrow (0, 0), \\
(100100011111011001) &\rightarrow (D^8 + D^7 + D^3 + 1, D^8 + D^5 + D^4 + D^2 + D + 1), \\
(110011101100000011) &\rightarrow (D^8 + D^6 + D^5 + D^4 + D + 1, D^8 + D^7 + 1), \\
(010111110011011010) &\rightarrow (D^7 + D^6 + D^5 + D^4 + D^3 + D, D^7 + D^5 + D^4 + D^2 + D), \\
(011100001010110111) &\rightarrow (D^8 + D^3 + D^2 + D, D^8 + D^7 + D^6 + D^4 + D^3 + D), \\
(111000010101101110) &\rightarrow (D^7 + D^2 + D + 1, D^7 + D^6 + D^5 + D^3 + D^2 + 1), \\
(101111100110110100) &\rightarrow (D^6 + D^5 + D^4 + D^3 + D^2 + 1, D^6 + D^4 + D^3 + D + 1), \\
(001011111001101101) &\rightarrow (D^8 + D^7 + D^6 + D^5 + D^4 + D^2, D^8 + D^6 + D^5 + D^3 + D^2).
\end{aligned}$$

De la même manière, nous évaluons pour chaque candidat $P'(D)$ non nul le vecteur $P'(D) = \frac{P(D)}{\text{PGCD}(P_1(D), P_2(D))}$. Il ne reste plus que le vecteur

$$\Delta^{(2)}(D) = (D^6 + D^5 + D^4 + D^3 + D^2 + 1, D^6 + D^4 + D^3 + D + 1).$$

Enfin, nous vérifions que ce vecteur est bien consistant en sa dernière coordonnée avec l'unique solution du sous-codeur 1. Nous avons bien retrouvé les mineurs correspondants aux sous-codeurs (1) et (2) ; nous retrouvons maintenant les polynômes de la matrice génératrice en résolvant les systèmes (3.7) page 94. Pour cela, nous devons exprimer les mineurs $\Delta_1^{(1)}(D)$ et $\Delta_2^{(2)}(D)$ sous leur forme binaire respective,

$$\begin{aligned}\Delta_1^{(1)}(D) &= D^6 + D^5 + D^2 + 1 & \rightarrow & \Delta_1^{(1)} = (1100101), \\ \Delta_2^{(1)}(D) &= D^6 + D^4 + D^3 + D + 1 & \rightarrow & \Delta_2^{(1)} = (1011011),\end{aligned}$$

$$\begin{aligned}\Delta_1^{(2)}(D) &= D^6 + D^5 + D^4 + D^4 + D^2 + 1 & \rightarrow & \Delta_1^{(2)} = (1111101), \\ \Delta_2^{(2)}(D) &= D^6 + D^4 + D^3 + D + 1 & \rightarrow & \Delta_2^{(2)} = (1011011).\end{aligned}$$

Les systèmes s'écrivent alors

$$F^{(1)}(k, m) = \begin{pmatrix} 1100101 & 1011011 \\ 0110010 & 0101101 \\ 0011001 & 0010110 \\ 0001100 & 0001011 \\ 0000110 & 0000101 \\ 0000011 & 0000010 \\ 0000001 & 0000001 \end{pmatrix}, F^{(2)}(k, m) = \begin{pmatrix} 1111101 & 1011011 \\ 0111110 & 0101101 \\ 0011111 & 0010110 \\ 0001111 & 0001011 \\ 0000111 & 0000101 \\ 0000011 & 0000010 \\ 0000001 & 0000001 \end{pmatrix}.$$

La résolution de ces systèmes renvoie deux noyaux de dimension 14. On vérifie aisément que les vecteurs (11011011010011) et (11011011011111) appartiennent aux noyaux respectifs de $F^{(1)}(k, m)$ et $F^{(2)}(k, m)$. Ces solutions correspondent aux solutions polynomiales $(D^6 + D^4 + D^3 + D + 1, D^6 + D^5 + D^2 + 1)$ et $(D^6 + D^4 + D^3 + D + 1, D^6 + D^5 + D^4 + D^3 + D^2 + 1)$. De plus, après restriction des solutions non nulles aux générateurs des modules de dimension 1, il ne reste plus que ces deux vecteurs qui sont consistants sur la première coordonnée. On en déduit que l'unique colonne de notre matrice génératrice est bien

$$\begin{pmatrix} D^6 + D^4 + D^3 + D + 1 \\ D^6 + D^5 + D^2 + 1 \\ D^6 + D^5 + D^4 + D^3 + D^2 + 1 \end{pmatrix}.$$

Remarque 3.5 :

Pour les $(n, 1)$ -codes convolutifs, la résolution de ces systèmes n'est pas nécessaire. En effet, dans ce cas très précis, les mineurs étant exactement les polynômes de la matrice génératrice, la première étape seule permet de conclure. Nous avons conservé la deuxième étape à des fins d'illustration.

3.2.2 Canal avec erreur

Tout comme pour la reconstruction des codes en blocs linéaires, les raisonnements précédents peuvent se transposer aisément en utilisant les notions de rang randomisé et de noyau randomisé introduits au paragraphe 2.1.2 page 57. Seule la première étape des algorithmes de reconstruction des (n, k) -codes convolutifs dans le cas sans erreur

prend en entrée des données bruitées ; la seconde étape ne fonctionne qu'avec des données exactes renvoyées par la première étape. Nous avons vu que cette première étape est équivalente à rechercher des relations de parité (relations entre les colonnes) dans les matrices $C^{(j)}(n_e, k_e, d_e)$. Or, l'algorithme de Gauss randomisé (cf. figure 2.2 page 59) permet de retrouver ces relations directement sur la matrice bruitée $\tilde{C}^{(j)}(n_e, k_e, d_e)$. Nous adaptons donc la première étape de l'algorithme de reconstruction des (n, k) -codes convolutifs dans le cas sans erreur, au contexte d'un canal bruité.

Comme précédemment, pour valider ou invalider les hypothèses faites sur n_e, k_e et d_e , nous observons le comportement différent des matrices $\tilde{C}^{(j)}(n_e, k_e, d_e)$. En effet, si $n_e \neq n$ ou $k_e \neq k$, les $C^{(j)}(n_e, k_e, d_e)$ et donc les $\tilde{C}^{(j)}(n_e, k_e, d_e)$ peuvent être considérées comme des matrices binaires aléatoires. Avec une forte probabilité, elles sont de rang maximum. Soit d_0 la mémoire du codeur de matrice génératrice canonique. Si $d_e < d_0$ alors il existe au moins un des systèmes (S_j) pour lequel la matrice $C^{(j)}(n_e, k_e, d_e)$ et donc $\tilde{C}^{(j)}(n_e, k_e, d_e)$ se comportera comme une matrice aléatoire. En revanche, lorsque $d_0 \leq d_e$, nous sommes dans les conditions de la proposition 3.2 page 93. Nous en déduisons donc qu'avec une probabilité dépendante de l'erreur, $(l, \gamma) - \text{rrg}(\tilde{C}(n, k, d_e)) < (k + 1)(d_e + 1)$.

L'algorithme de reconstruction des (n, k) -codes convolutifs pour un canal bruité se déduit alors de celui dédié au cas sans erreur. Il suffit de remplacer l'évaluation du noyau des $C^{(j)}(n_e, k_e, d_e)$ par l'évaluation du noyau randomisé de $\tilde{C}^{(j)}(n_e, k_e, d_e)$.

3.3 Analyse des algorithmes

L'étude asymptotique de la complexité de la reconstruction des codes convolutifs est un peu différent de celle des codes en blocs linéaires comme expliqué au paragraphe 1.1. Pour les codes en blocs, on considère k et n asymptotiquement grands, tandis que pour les codes convolutifs, k et n sont petits et bornés et m est asymptotiquement grand.

3.3.1 Canal sans erreur

Cas des $(n, 1)$ -codes convolutifs

Montrons tout d'abord que l'algorithme de reconstruction des $(n, 1)$ -codes convolutifs renvoie une matrice canonique. En effet, l'algorithme de Berlekamp-Massey renvoie des polynômes générateurs minimaux. On en déduit que la matrice génératrice $G(D)$ renvoyée est de degré externe minimale, elle est donc canonique. Soit à reconstruire le (n, k, m) -code convolutif $\mathcal{C}$, nous avons donc

$$\deg \mathcal{C} = m = \text{extdeg } G(D) = \text{intdeg } G(D),$$

or $\text{intdeg } G(D)$ est le degré maximum pris sur ses mineurs d'ordre k . Le degré maximum des polynômes minimaux à retrouver est donc m . En effet, dans le cas particulier des $(n, 1, m)$ -codes convolutifs, les polynômes de la matrice génératrice sont aussi les mineurs d'ordre 1.

Proposition 3.3 *Le nombre minimum $L_{\min}$ de bits nécessaires à la reconstruction des $(n, 1, m)$ -codes convolutifs dans le cas sans erreur est*

$$L_{\min} = 2nm.$$

Preuve :

L'algorithme effectue $(n - 1)$ Berlekamp-Massey pour retrouver des polynômes de degré au plus m . Pour chacun, il a besoin d'évaluer $Z_i(D)$, pour $i = 1 \dots n - 1$, à l'ordre $2m$. Cette évaluation nécessite d'avoir au minimum $2m$ bits pour chaque $Y_i(D)$. Soit au total, $2mn$ bits. ■

La méthode initiale proposée par É. Filiol [71] nécessite au minimum $n(3m + 1)$ bits ; le gain obtenu est donc d'un facteur $3/2$.

Proposition 3.4 *La complexité de l'algorithme de reconstruction des $(n, 1, m)$ -codes convolutifs dans le cas sans erreur est*

$$\mathcal{O}(nm^3).$$

Preuve :

La complexité d'une exécution de l'algorithme de Berlekamp-Massey (BM) est $\mathcal{O}(N^2)$ où $N = 2m$. Le calcul des $Z_i(D)$ étant linéaire en leur taille $(2nm)$, il sera négligé devant le coût d'une exécution de BM. De plus, l'algorithme recherche en aveugle la valeur des paramètres n et m . De plus, pour chaque valeur de $n_e < n$, il effectue deux exécutions de BM, la consistance de $G_{1,1}(D)$ n'étant pas vérifiée. Pour $n_e = n$, il doit en effectuer $(n - 1)$. La complexité de l'algorithme de reconstruction est donc

$$\mathcal{O} \left(\sum_{n_e=2}^{n-1} 2 \sum_{d_e=1}^m (2d_e)^2 + (n-1) \sum_{d_e=1}^m (2d_e)^2 \right) = \mathcal{O} \left(12(n-1) \sum_{d_e=1}^m (d_e)^2 \right) = \mathcal{O}(nm^3). \quad \blacksquare$$

Pour le cas d'un canal sans erreur É. Filiol utilise l'algorithme de Gauss sur une matrice carrée de taille $2(d_e + 1)$. Le calcul de la complexité est identique en remplaçant $(2d_e)^2$ par $(2(d_e + 1))^3$, nous obtenons $\mathcal{O}(nm^4)$, avec un gain en complexité d'un facteur $(4/3)m$.

Cas général

Théorème 3.1 *L'algorithme de reconstruction des (n, k) -codes convolutifs retourne un ensemble de matrices génératrices canoniques.*

Montrons dans un premier temps que les mineurs d'ordre k calculés à l'étape 1 de l'algorithme de reconstruction sont ceux d'une matrice basique. Après la résolution de (S_1) , nous avons

$$\text{PGCD}(\Delta_1^{(1)}(D), \dots, \Delta_{k+1}^{(1)}(D)) = 1,$$

par construction. Supposons maintenant que parmi les $l < (n - k)$ premiers vecteurs de la liste, il existe j tel que $\text{PGCD}_{i=1 \dots k+1}(\Delta_i^{(j)}(D)) = 1$. Soit $\Delta^{(l+1)}(D)$ le $(l + 1)$ ème vecteur de la liste, consistant avec les l premiers en sa composante $(k + 1)$, de facteur de consistance $Q(D)$. Si $\deg(\Delta_{k+1}^{(1)}(D)) \geq \deg(\Delta_{k+1}^{(l+1)}(D))$, alors $\Delta^{(l+1)}(D)$ est multiplié par $Q(D)$ et $\Delta^{(j)}(D)$ est inchangé. Sinon ce sont les l premiers vecteurs de la liste qui sont multipliés par $Q(D)$, dont $\Delta^{(j)}(D)$ et $\Delta^{(l+1)}(D)$ est ajouté à la liste. Or, par construction, $\text{PGCD}_{i=1 \dots k+1}(\Delta_i^{(l+1)}(D)) = 1$. Dans chaque liste, il existe donc un vecteur dont les coordonnées, sont premières entre elles. La deuxième étape renvoie donc des matrices génératrices $G(D)$ basiques. Supposons maintenant qu'elles ne sont pas réduites, *i.e.* il

existe une matrice canonique $G_0(D)$ et $P(D) \in F[D]$ tels que $G(D) = P(D)G_0(D)$. Si $\Delta_i(D)$ représente les mineurs d'ordre k de $G(D)$ et $\delta_i(D)$ ceux de $G_0(D)$ alors

$$\Delta_j(D) = P^k(D)\delta_i(D),$$

ce qui est contradictoire avec le fait que les mineurs d'ordre k n'ont pas de facteur commun. L'algorithme renvoie bien des matrices basiques et réduites, c'est-à-dire canoniques. ■

Ceci démontre la conjecture d'É. Filiol [71, p. 170]. On en déduit que le degré maximum des polynômes à retrouver lors de la première étape est inférieur ou égal à m .

Proposition 3.5 *Le nombre minimum L_{min} de bits nécessaires à la reconstruction des (n, k, m) -codes convolutifs dans le cas sans erreur est*

$$L_{min} = n((k+2)(m+1) - 1).$$

Preuve :

L'algorithme requiert $(k+1)(m+1) + (m+1) - 1$ bits pour chaque $Y_i(D)$ pour construire les matrices $C^{(j)}(n, k, m)$ soit un total de $n((k+2)(m+1) - 1)$ bits. ■

Bien que l'algorithme de É. Filiol soit identique dans l'esprit, certaines équations ont pu être grandement simplifiées et permettre ainsi des calculs encore plus fins sur la complexité et le nombre de bits nécessaires. Il propose de retrouver des polynômes de degré $((2^k - 1) \deg G(D) + 1)$, qui s'avèrent être multiples des mineurs d'ordre k . En pratique, l'algorithme qu'il conçoit trouve d'abord un générateur du module des solutions. Dans un deuxième temps, il résout formellement le système (3.6) page 93 par identification terme à terme. L'étude algébrique nous a permis de mettre en évidence la structure et la nature des solutions du système de la proposition 3.1 page 90 et surtout d'automatiser complètement le processus de reconstruction. Le calcul du nombre minimum de bits nécessaires pour la technique d'É. Filiol donne $(k+1)((2^k - 1) \deg G(D) + 1)$ avec $\deg G(D)$ de l'ordre de m/k . Le facteur gain obtenu est donc de l'ordre de $(2^k/k)$.

Proposition 3.6 *La complexité de l'algorithme de reconstruction des (n, k, m) -codes convolutifs dans le cas sans erreur est*

$$\mathcal{O}(n^5 m^4).$$

Preuve :

La recherche par pivot de Gauss des noyaux des matrices $C^{(j)}(n_e, k_e, d_e)$ étant l'étape la plus coûteuse, l'évaluation de la consistance des solutions et la recherche du générateur des modules de longueur 1 sont considérées comme négligeables. Dans les cas $n_e \neq n$, en moyenne un seul pivot de Gauss est effectué. On en déduit que retrouver le paramètre n en aveugle est de complexité

$$\mathcal{O}\left(\sum_{n_e=2}^n \sum_{k_e=1}^{n_e-1} \sum_{d_e=1}^m ((k+1)(d_e+1))^3\right) = \mathcal{O}(n^5 m^4).$$

Quand $n_e = n$, l'algorithme effectue la résolution des $(n-k)$ sous-codeurs (S_j) à partir de $k_e = k$ et $d_e \geq d_0$. Nous pouvons donc borner la complexité de cette étape par

$$\mathcal{O}\left(\sum_{k_e=1}^{n-1} \sum_{d_e=1}^m (n-k) ((k_e+1)(d_e+1))^3\right) = \mathcal{O}(n^5 m^4). \quad \blacksquare$$

Le degré estimé des polynômes reconstruits par l'algorithme d'É. Filiol est de l'ordre de $2^k m/k$. La complexité est donc améliorée par un facteur de l'ordre de

$$\left(\frac{2^k}{k}\right)^4.$$

3.3.2 Canal avec erreur

Dans le cadre d'un canal binaire bruité, l'évaluation des noyaux des matrices $\tilde{C}(n_e, k_e, d_e)$ se fait en utilisant l'algorithme de Gauss randomisé. Les $(n - k)$ vecteurs $\Delta^{(j)}$ peuvent être considérés comme des relations de parité d'un code en blocs linéaire au regard de la relation (3.5) page 92. L'algorithme de reconstruction des codes en blocs linéaires pour un canal bruité (cf. paragraphe 2.2.2), peut alors s'appliquer. La probabilité de détection de cet algorithme, calculée au paragraphe 2.3.2, est une fonction de la relation de parité de poids de Hamming maximum ; dans notre cas

$$\Delta = \underset{i=1 \dots n-k}{\operatorname{Argmax}} w(\Delta^{(i)}).$$

Dans le cadre du calcul de la probabilité de détection, É. Filiol s'est placé dans les hypothèses les plus défavorables. En pratique, ses résultats expérimentaux se sont avérés être bien meilleurs que ceux attendus par l'étude théorique. Ce décalage s'explique par le fait que l'algorithme de Gauss, renvoie les bonnes relations de parité non seulement lorsque la matrice ne contient pas d'erreur mais aussi lorsque les erreurs se compensent (nombre pair d'erreurs sur les bits correspondants aux relations de parité) ou encore lorsque celles-ci se situent toujours en dessous de la diagonale pendant tout le processus de Gauss.

D'autre part, la probabilité de fausse alarme peut être considérée comme négligeable. En effet, pour qu'un mauvais code soit détecté, il faut tout d'abord qu'au moins une des $(n - k)$ résolutions de systèmes (S_j) détecte une relation de parité qui n'est pas valide. De plus, cette relation doit être consistante pour être acceptée. Enfin, il faut aussi que les relations de parités détectées à l'étape 1 permettent à l'étape 2 de retrouver une matrice canonique d'un codeur valide.

Le calcul de $\mathcal{P}_{det}$ en fonction de Δ permet de choisir un nombre d'itérations l pour l'algorithme de Gauss randomisé. Celui-ci doit vérifier $1/\mathcal{P}_{det} \leq l$. Ceci permet d'en déduire le nombre minimum de bits N_{min} pour chaque $Y_i(D)$ nécessaire à la reconstruction. La matrice d'interception est une matrice de Hankel par blocs de taille $(m + 1)$, ce qui implique que chaque bit intercepté intervient dans $(m + 1)$ équations. Nous devons observer suffisamment de bits pour pouvoir effectuer les l tirages de Monte Carlo de l'algorithme de Gauss randomisé, c'est-à-dire

$$\binom{(m+1)N_{min}}{(m+1)(k+1)} \gg l.$$

Enfin, la complexité de l'algorithme de reconstruction des (n, k, m) -codes convolutifs dans un canal bruité est donnée par

$$\mathcal{O}\left(\frac{n^5 m^4}{\mathcal{P}_{det}}\right).$$

3.4 Résultats expérimentaux

Dans ce paragraphe, nous présentons les résultats expérimentaux de la reconstruction des codes convolutifs optimaux, étudiés au paragraphe 1.2.4, ainsi que ceux extraits des normes de l'annexe B. Comme pour les simulations de la reconstruction des codes en blocs linéaires, l'objectif est de mettre en évidence de « bonnes » valeurs pour les paramètres γ (seuil de détection) et l (nombre d'itérations) de l'algorithme de Sicot-Houcke. Nous nous sommes alors focalisés uniquement sur la première étape, la seconde étant déterministe. Pour ce faire, nous avons tout d'abord fait varier le TEB de 5.10^{-4} à 2.10^{-1} . Pour chaque valeur de l'erreur, une valeur du seuil γ_{max} pour laquelle la densité d de chaque vecteur de mineurs relatifs aux sous-codeurs est telle que $d \leq \gamma_{max}$ a été déterminée. De plus, γ_{max} a été choisi au minimum égal à 0.15. Enfin, nous nous sommes placés dans le cas où n et m ont été correctement estimés. Le nombre d'itérations et le temps nécessaires pour reconstruire le code sont alors mesurés. Lorsque tout le code est reconstruit, le nombre d'itérations nous donne une idée sur la probabilité de détection de la relation de parité de poids le plus fort et donc la plus difficile à détecter. Pour chaque valeur de l'erreur et chaque codeur, nous avons effectué une moyenne sur 100 reconstruction et obtenu ainsi le seuil maximum moyen, le temps moyen de la première étape ainsi que le nombre moyen d'itérations nécessaires. Les algorithmes ont été implémentés en Magma et les simulations ont été effectuées sur des processeurs AMD Thurion 2000+ MP, cadencés à 1,6 Ghz ; le nombre de bits interceptés a été fixé à 400, ce qui correspond à 50 octets. D'autre part, nous nous sommes limités à une heure avant de conclure à la non détection. Nous avons ainsi évalué la probabilité $\mathcal{P}_{ND}$ de ne pas détecter un codeur valide en moins d'une heure.

La résolution du système (3.7) page 94 pour le sous-codeur (1) permet de déterminer les colonnes partielles $(G_{1,j}(D), \dots, G_{k,j}(D), G_{k+1,j}(D))^T$ pour $j = 1 \dots k$. D'autre part, la résolution du même système pour le sous-codeur (p) nous permet d'obtenir le vecteur $(G_{1,j}(D), \dots, G_{k,j}(D), G_{k+p,j}(D))^T$. Nous pouvons alors adapter l'étape de résolution du système (3.7) page 94 en garantissant la consistance grâce à la relation

$$G_{k+p,j}(D) = \frac{\sum_{i=1}^k \Delta_i^{(p)}(D) G_{i,j}(D)}{\Delta_{k+1}^{(p)}(D)},$$

issue de l'équation (3.6) page 93. C'est ce choix que nous avons privilégié pour notre implémentation.

Les résultats, regroupés dans le tableau 2.1 page 83, illustrent l'efficacité de la méthode pour des codeurs convolutifs utilisés en pratique, notamment en comparaison avec ceux obtenus pour les codes en blocs linéaires. Tout d'abord, pour une même quantité de bits interceptés l , le nombre d'équations pour un code convolutif, environ l , est beaucoup plus important que pour un code en bloc linéaire, environ l/n . Ceci permet de faire chuter drastiquement la probabilité de fausse alarme. Il est à noter qu'aucune fausse alarme n'a été détectée sur l'ensemble des simulations. De plus pour chaque sous-codeur, une seule relation de parité de longueur $(k+1)(m+1)$ est à retrouver. En outre, cette longueur est en pratique est plutôt petite, pour des raisons d'implémentation. En effet, le nombre nécessaire de registres mémoire est de l'ordre de k fois le degré d'un codeur canonique.

<i>Codeur</i>	(n, k, m)	TEB 5.10^{-4}				TEB 10^{-3}				TEB 2.10^{-3}			
		<i>temps</i>	<i>nb ité.</i>	γ	$\mathcal{P}_{ND}$	<i>temps</i>	<i>nb ité.</i>	γ	$\mathcal{P}_{ND}$	<i>temps</i>	<i>nb ité.</i>	γ	$\mathcal{P}_{ND}$
(561 753)	(2,1,8)	0.019 s	2	0.15	0	0.021 s	2	0.15	0	0.024 s	2	0.15	0
(133 145 175)	(3,1,6)	0.027 s	1	0.15	0	0.03 s	1	0.15	0	0.035 s	1	0.15	0
(557 663 711)	(3,1,8)	0.032 s	1	0.15	0	0.035 s	1	0.15	0	0.045	1	0.15	0
(765 671 513 473)	(4,1,8)	0.08 s	1	0.15	0	0.089 s	1	0.15	0	0.12 s	2	0.15	0
$\begin{pmatrix} 51 & 5 & 121 \\ 15 & 40 & 111 \end{pmatrix}$	(3,2,11)	0.04 s	2	0.15	0	0.054 s	3	0.15	0	0.165 s	7	0.15	0
$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$	(4,3,0)	0.007 s	2	0.15	0	0.007 s	2	0.15	0	0.005 s	2	0.15	0
$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 \\ 0 & 3 & 1 & 0 \end{pmatrix}$	(4,3,2)	0.014 s	2	0.15	0	0.016 s	2	0.15 s	0	0.019 s	2	0.15 s	0
$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 3 & 2 & 1 \\ 0 & 2 & 5 & 5 \end{pmatrix}$	(4,3,3)	0.015 s	1	0.15	0	0.019 s	2	0.15	0	0.021 s	2	0.15	0
$\begin{pmatrix} 3 & 2 & 2 & 3 \\ 4 & 3 & 0 & 7 \\ 0 & 2 & 5 & 5 \end{pmatrix}$	(4,3,5)	0.021 s	1	0.15	0	0.026 s	2	0.15	0	0.032 s	2	0.15	0
$\begin{pmatrix} 3 & 4 & 0 & 7 \\ 6 & 1 & 4 & 3 \\ 2 & 6 & 7 & 1 \end{pmatrix}$	(4,3,6)	0.026 s	1	0.15	0	0.033 s	2	0.15	0	0.059 s	3	0.15	0
$\begin{pmatrix} 7 & 6 & 2 & 1 \\ 14 & 5 & 0 & 15 \\ 10 & 4 & 17 & 1 \end{pmatrix}$	(4,3,8)	0.039 s	2	0.15	0	0.101 s	5	0.15	0	0.237 s	10	0.15	0
$\begin{pmatrix} 1 & 14 & 16 & 3 \\ 10 & 13 & 2 & 7 \\ 16 & 0 & 3 & 13 \end{pmatrix}$	(4,3,9)	0.057 s	2	0.15	0	0.067 s	3	0.15	0	0.418 s	16	0.15	0

TAB. 3.1 – Reconstruction des codes convolutifs, TEB de 5.10^{-4} à 2.10^{-3}

<i>Codeur</i>	(n, k, m)	TEB 5.10^{-3}				TEB 10^{-2}				TEB 2.10^{-2}			
		<i>temps</i>	<i>nb ité.</i>	γ	$\mathcal{P}_{ND}$	<i>temps</i>	<i>nb ité.</i>	γ	$\mathcal{P}_{ND}$	<i>temps</i>	<i>nb ité.</i>	γ	$\mathcal{P}_{ND}$
(561 753)	(2,1,8)	0.052 s	5	0.172	0	0.175 s	14	0.202	0	1.495 s	119	0.3	0
(133 145 175)	(3,1,6)	0.051 s	2	0.15	0	0.132 s	3	0.227	0	0.484 s	9	0.3	0
(557 663 711)	(3,1,8)	0.073 s	1	0.15	0	0.192 s	2	0.215	0	1.668 s	4	0.315	0
(765 671 513 473)	(4,1,8)	0.247 s	2	0.15	0	0.764 s	7	0.215	0	7.888 s	36	0.33	0
$\begin{pmatrix} 51 & 5 & 121 \\ 15 & 40 & 111 \end{pmatrix}$	(3,2,11)	1.957s	82	0.163	0	38.982 s	1 786	0.24	0	17 mn 17 s	43 857	0.36	0.41
$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$	(4,3,0)	0.006 s	2	0.15	0	0.007 s	2	0.15	0	0.007 s	2	0.15	0
$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 \\ 0 & 3 & 1 & 0 \end{pmatrix}$	(4,3,2)	0.019 s	3	0.15	0	0.044 s	5	0.15	0	0.092 s	11	0.194	0
$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 3 & 2 & 1 \\ 0 & 2 & 5 & 5 \end{pmatrix}$	(4,3,3)	0.039 s	3	0.15	0	0.118 s	11	0.172	0	0.724 s	64	0.28	0
$\begin{pmatrix} 3 & 2 & 2 & 3 \\ 4 & 3 & 0 & 7 \\ 0 & 2 & 5 & 5 \end{pmatrix}$	(4,3,5)	0.121 s	8	0.15	0	0.611 s	37	0.22	0	18.284 s	1 127	0.294	0
$\begin{pmatrix} 3 & 4 & 0 & 7 \\ 6 & 1 & 4 & 3 \\ 2 & 6 & 7 & 1 \end{pmatrix}$	(4,3,6)	0.331 s	18	0.157	0	3.853 s	226	0.25	0	3 mn 16 s	10 444	0.366	0.02
$\begin{pmatrix} 7 & 6 & 2 & 1 \\ 14 & 5 & 0 & 15 \\ 10 & 4 & 17 & 1 \end{pmatrix}$	(4,3,8)	17.908 s	754	0.212	0	7 mn 12 s	17 997	0.322	0.04	ND	ND	ND	ND
$\begin{pmatrix} 1 & 14 & 16 & 3 \\ 10 & 13 & 2 & 7 \\ 16 & 0 & 3 & 13 \end{pmatrix}$	(4,3,9)	14.075 s	542	0.193	0	6 mn 16 s	14 305	0.29	0.07	ND	ND	ND	ND

TAB. 3.2 – Reconstruction des codes convolutifs 5.10^{-3} à 2.10^{-2}

3.5 Conclusion et perspectives

Comme pour la reconstruction des codes en blocs linéaires, les algorithmes présentés ne permettent pas de décoder. Les algorithmes de reconstruction des codes convolutifs renvoient des matrices canoniques potentielles. Sans aucune autre hypothèse, il subsistent deux indéterminées intrinsèques à la nature des codeurs convolutifs. Tout d'abord, parmi tous les codeurs $F(D)$ -équivalents, une infinité peuvent engendrer la même sortie. Par exemple, si $G(D)$ est le codeur effectif qui a généré la sortie $Y(D)$ appartenant à un code $\mathcal{C}$, à partir du message $X(D)$, alors quel que soit $P(D) \in F[D]$, il existe un message $X_1(D) = 1/P(D) \cdot X(D)$ et une matrice génératrice polynomiale $G_1(D) = P(D) \cdot G(D)$ tels que

$$Y(D) = G_1(D)X_1(D),$$

et il existe un message $X_2(D) = P(D) \cdot X(D)$ et une matrice génératrice de $\mathcal{C}$, $G_2(D) = G(D)/P(D)$ tels que

$$Y(D) = G_2(D)X_2(D).$$

De même, quel que soit P la matrice d'un entrelaceur bloc linéaire, il existe un message $X_3(D) = P^{-1}X(D)$ et une matrice génératrice de $\mathcal{C}$, $G_3(D) = G(D)P$ tels que

$$Y(D) = G_3(D)X_3(D).$$

Il existe donc une infinité de couples $(G'(D), X'(D))$ tels que $Y(D) = G'(D)X'(D)$. Sans aucune hypothèse supplémentaire, le problème de décodage est indécidable.

Pour les codes convolutifs poinçonnés, É. Filiol a montré, [71, ch. 9], que la reconstruction ne diffère en rien d'un code convolutif non poinçonné. Dans ce cas, on retrouve une matrice canonique du code convolutif après poinçonnage et non celle du code parent. De même, si le codeur utilisé est récursif et engendre le code $\mathcal{C}$, les algorithmes de reconstruction renverront une matrice génératrice canonique engendrant $\mathcal{C}$. De plus, si le codeur est systématique, les bits d'information permettent de lever les indéterminées mises en évidence. Enfin, si le schéma de codage se compose d'un code convolutif suivi d'un entrelaceur bloc linéaire de profondeur pn , $p \in \mathbb{N}^*$, il faudra reconstruire le code de dimension pn engendré par la matrice de dimension $pn \times pk$ dont la diagonale est la matrice génératrice $G(D)$. Malheureusement, l'entrelaceur « casse » la structure de module des solutions de l'étape 1. La seule différence avec la reconstruction classique des codes convolutifs, réside dans la forme des matrices $\tilde{C}(n_e, k_e, d_e)$ qui perdent leur structure particulière de matrice de Hankel par blocs. Il faut donc dans ce cas, utiliser les algorithmes de reconstruction pour les codes en blocs linéaires.

La technique de reconstruction proposée est propre aux codes convolutifs, notamment par la forme particulière de la matrice d'interception (cf. relation (3.4) page 92). D'autre part, nous avons vu au chapitre 1 que les codes linéaires peuvent être considérés comme des codes convolutifs particuliers. Du point de vue des algorithmes de reconstruction, nous pouvons montrer le contraire. En effet, le système (3.4) page 92 est équivalent au système de taille $(\lfloor \frac{N-1}{d} \rfloor - 1) \times (k+1)(d+1)$ défini par

$$\begin{pmatrix} y_1(d) & \dots & y_1(0) & \dots & y_{k+1}(d) & \dots & y_{k+1}(0) \\ y_1(2d+1) & \dots & y_1(d+1) & \dots & y_{k+1}(2d+1) & \dots & y_{k+1}(d+1) \\ \vdots & & \vdots & & \vdots & & \vdots \\ y_1(N) & \dots & y_1(N-d) & \dots & y_{k+1}(N) & \dots & y_{k+1}(N-d) \end{pmatrix}.$$

La matrice d'interception du sous-codeur n'est plus de Hankel par blocs. Résoudre ce système est alors équivalent à reconstruire un code en blocs linéaire de longueur $(k+1)(d+1)$, dont les relations de parité sont les mineurs du sous-codeur. De même, en concaténant tous les sous-systèmes, nous obtenons un système de taille $(\lfloor \frac{N-1}{m} \rfloor - 1) \times n(m+1)$ défini par

$$\begin{pmatrix} y_1(m) & \dots & y_1(0) & \dots & y_{k+1}(m) & \dots & y_n(0) \\ y_1(2m+1) & \dots & y_1(m+1) & \dots & y_n(2m+1) & \dots & y_n(m+1) \\ \vdots & & \vdots & & \vdots & & \vdots \\ y_1(N) & \dots & y_1(N-m) & \dots & y_n(N) & \dots & y_n(N-m) \end{pmatrix}.$$

Résoudre ce système est équivalent à reconstruire un code en blocs linéaire de longueur $n(m+1)$ dont les relations de parité sont les mineurs de tous les $(k, k+1, m)$ -sous-codeurs. Il apparaît enfin que la reconstruction des codes convolutifs peut être considérée comme une technique de reconstruction de codes en blocs linéaires dont les relations de parité sont les mineurs des $(k, k+1, m)$ -sous-codeurs convolutifs. La plus-value d'utiliser préférentiellement la technique particulière est double. Tout d'abord, $2(m+1)$ bits consécutifs d'une même piste permettent d'écrire $(m+2)$ équations contre 2 si l'on utilise l'algorithme de reconstruction des codes en blocs linéaires. Nous rappelons que les fausses alarmes décroissent rapidement avec l'augmentation du nombre d'équations. De plus, la taille des sous-systèmes est $n/(k+1)$ fois plus petite que celle du système équivalent. Or, la probabilité de détection est exponentiellement décroissante avec l'augmentation de la longueur du code à reconstruire. Là encore, le facteur de gain est notable. De plus, la reconstruction des codes convolutifs ne nécessite pas de synchronisation, ce qui permet le gain d'un facteur de complexité ; $\mathcal{O}(m^4)$ contre $\mathcal{O}(m^5)$ pour la reconstruction des codes en blocs linéaires.

Les matrices d'interception de codes convolutifs possèdent une structure de matrice de Hankel par blocs, c'est-à-dire qu'elles sont composées de la concaténation de matrices de Toeplitz inversées. Cette structure forte nous laisse présager d'une amélioration de la complexité de la résolution des systèmes linéaires (qui est l'étape la plus coûteuse) en adaptant les algorithmes de résolution de systèmes de Toeplitz aux systèmes de Toeplitz par blocs. En effet, les algorithmes de Shur [172] et de Levinson [131] permettent d'atteindre des complexités en $\mathcal{O}(n^2)$. L'utilisation de *FFT* (Transformée de Fourier Rapide) [93] permet même d'atteindre $\mathcal{O}(n \log n)$.

Enfin, l'algorithme de Berlekamp-Massey (BM) a permis d'améliorer notablement la complexité de l'algorithme de reconstruction des $(n, 1)$ -codes convolutifs et de diminuer le nombre minimum de bits nécessaires. Moyennant la réécriture du système initial 3.2, nous prévoyons un gain similaire si l'on arrive à appliquer des algorithmes de type Berlekamp-Massey sous sa forme matricielle, dans l'esprit de [55, 54, 145, 192, 205]. Une adaptation de l'algorithme BM dans le cas d'une suite faiblement bruitée [163, 179, 125] semble envisageable avec pour objectif une adaptation en un algorithme BM matriciel avec une suite bruitée. D'autre part, toute amélioration de l'algorithme de reconstruction des codes en blocs linéaires se traduit par une amélioration de l'étape 1 de l'algorithme de reconstruction des codes convolutifs. Une façon un peu artificielle de gagner un peu sur la complexité de l'algorithme, est de ne pas chercher en aveugle le degré du code mais de fixer une seule hypothèse d_e sur d_0 , suffisamment grande pour espérer avoir $d_e > d_0$. En cas d'échec, plutôt que d'incrémenter d_e de 1 on peut prendre un pas un peu plus grand.

Reconstruction des turbo-codes

« Dans une discussion, je suis toujours côté de l'adversaire. »

André Gide

Sommaire

4.1	Tries dynamiques	117
4.2	Algorithmes de reconstruction	120
4.2.1	Canal sans erreur	120
4.2.2	Canal avec erreur	123
4.3	Analyse des algorithmes	126
4.3.1	Canal sans erreur	126
4.3.2	Canal avec erreur	127
4.4	Résultats expérimentaux	128
4.5	Conclusion et perspectives	130

EN 1993, C. Berrou, A. Glavieux et P. Thitimajshima de l'ENST de Bretagne font sensation avec [31] en reprenant l'idée de concaténation de codes introduite par Forney [86] et en montrant qu'il est possible, pour un taux d'erreur de 10^{-5} et un rendement de 0.5, d'atteindre la limite de Shannon à 0.5 dB près. Les auteurs concatènent deux codes convolutifs en parallèle en séparant leurs entrées par un entrelaceur. Chaque code le composant codant donc une version permutée différente de la même séquence d'information comme l'illustre la figure 4.1.

Majoritairement les codes convolutifs utilisés sont systématiques et récursifs mais en règle générale, les bits systématiques issus du codeur C_2 ne sont pas transmis (sauf dans le cas des canaux de type Rayleigh à évanouissement). En pratique, nous observons très souvent $C_1 = C_2$. Dans le cas général, le rendement R du turbo-codeur s'exprime simplement en fonction de R_1 et R_2 , les rendements respectifs de C_1 et C_2 ,

$$R = \frac{R_1 R_2}{R_1 + R_2 - R_1 R_2} = \frac{R_1}{2 - R_1} \text{ si } C_1 = C_2,$$

Le décodage appliqué ensuite est itératif, chaque décodeur fournissant au suivant une information souple, ou *valeur de confiance*, lui permettant d'affiner ses propres décisions.

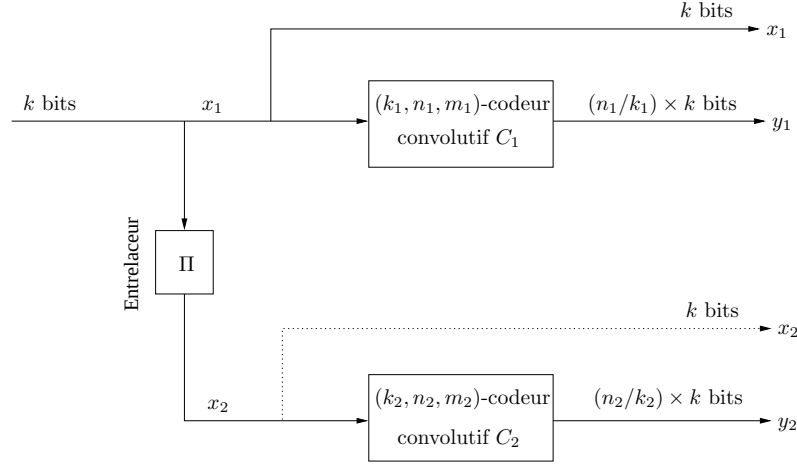


FIG. 4.1 – Schéma d'un turbo-codeur parallèle

L'algorithme de décodage est en fait un algorithme de Viterbi amélioré et adapté. C'est à ce caractère itératif qu'est dû le nom de « turbo » qui rappelle le fonctionnement du moteur turbo. Le lecteur intéressé par les aspects théoriques relatifs aux turbo-codes trouvera plus de détails dans [27, 26, 30, 31, 32, 92, 199].

Dans le cadre de la reconstruction de turbo-codes, nous nous sommes placés dans la continuité du chapitre 3. En effet, les algorithmes que nous avons présentés dans ce chapitre nous permettent de retrouver les codes convolutif $\mathcal{C}_1$ et $\mathcal{C}_2$. L'observateur intercepte $\tilde{x}_1$, $\tilde{y}_1$ et $\tilde{y}_2$, correspondant respectivement à x_1 , y_1 et y_2 bruités par le canal. Aux vues des améliorations présentées au paragraphe 3.5, la structure même des turbo-codeurs parallèles nous place dans une situation favorable. Le codeur convolutif $\mathcal{C}_1$ étant systématique, nous sommes capables de retrouver de façon univoque ses paramètres et donc de pouvoir décoder et ainsi corriger quelques erreurs de $\tilde{x}_1$ et obtenir $\hat{x}_1$. De même, pour reconstruire le code convolutif $\mathcal{C}_2$, nous avons accès à la sortie de $\mathcal{C}_2$ mais aussi à une version permutée et bruitée de son entrée. Dans le cadre de notre étude, nous nous plaçons dans le cas le plus fréquemment rencontré (cf. annexe B), c'est-à-dire dans le cas $\mathcal{C}_1 = \mathcal{C}_2$. Le cas $\mathcal{C}_1 \neq \mathcal{C}_2$ est un peu plus complexe et nécessite une étape préliminaire. Les algorithmes de reconstruction nous renvoient une matrice $G'_2(D)$ génératrice de $\mathcal{C}_2$ telle que

$$G_2(D) = G'_2(D)P(D),$$

où $P(D)$ est une matrice inversible ; $G_2(D)$ et $G'_2(D)$ étant $F(D)$ -équivalentes. Après décodage et correction de quelques erreurs, nous obtenons un message $\tilde{x}'_2$ qui correspond à l'image de x_2 par $P(D)$, bruitée. En utilisant le formalisme algébrique introduit au chapitre 1, nous devons résoudre le système suivant

$$X_1(D) + (\Pi^{-1}(D)P(D)) X'_2(D) = 0,$$

où $\Pi(D)$ est la matrice de Π , à coefficients dans $\mathbb{F}_2$. Pour ce faire, nous avons à notre disposition des instances bruitées de $X_1(D)$ et $X'_2(D)$, ce qui revient à retrouver des relations de parité d'un code linéaire particulier en observant des mots de code bruités. Il nous suffit alors d'utiliser l'algorithme de Sicot et Houcke, présenté au chapitre 2.

Nous proposons donc dans ce dernier chapitre consacré à la reconstruction des codes correcteurs d'erreurs, d'étudier un algorithme de reconstruction d'un entrelaceur bloc dans un canal quelconque, connaissant un certain nombre de couples d'entrée et de sortie. L'objectif est de retrouver l'entrelaceur Π afin d'obtenir non seulement le turbo-code mais aussi un décodeur équivalent. Pour cela, nous présentons succinctement au paragraphe 4.1 la notion de *trie dynamique* qui est au cœur des algorithmes de reconstruction d'entrelaceurs. Puis, au paragraphe 4.2, nous présentons deux algorithmes de reconstruction, adaptés respectivement aux canaux sans et avec erreur. Nous effectuons ensuite l'analyse de ces algorithmes au paragraphe 4.3 et détaillons au paragraphe 4.4 les résultats expérimentaux que nous avons obtenus. Enfin, au paragraphe 4.5, nous discutons des limites de telles techniques et des améliorations possibles. Le résultat de ces travaux ont été publiés dans [9, 14].

4.1 Tries dynamiques

Comme précédemment, nous nous plaçons dans le cas d'un canal binaire symétrique. L'étude est menée en prenant comme hypothèse que x_1 est issue d'une source infinie sans mémoire avec $\mathcal{P}_r(x_1(j) = 1) = \mathcal{P}_r(x_1(j) = 0) = 0.5, \forall j$. Cette hypothèse est raisonnable dans la mesure où x_1 correspond au message initial compressé, éventuellement chiffré et passé dans un brasseur. Le lecteur intéressé trouvera dans [8] l'étude généralisée pour une source sur un alphabet quelconque et avec une probabilité d'occurrence des symboles différente de 0.5. Le chapitre 3 nous fournit les algorithmes nécessaires pour retrouver la matrice génératrice de C_1 et celle de C_2 . La reconstruction des turbo-codes se réduit alors à un algorithme permettant de retrouver un entrelaceur bloc linéaire à partir de ses entrées et sorties éventuellement bruitées. Nous rappelons qu'un entrelaceur bloc linéaire Π de profondeur n conserve le poids de Hamming et est défini de façon univoque par la donnée d'une permutation σ de $[1, n]$ (cf. définition 2.1 page 50). Soit à retrouver l'entrelaceur défini par σ à partir d'une suite de couples d'entrée et de sortie de n bits chacune, $x = (x^i)_{i>0}, y = (y^i)_{i>0}$, tels que

$$\forall i > 0, j = 1 \dots n, \quad y^i(j) = x^i(\sigma^{-1}(j)). \quad (4.1)$$

Pour une meilleure lisibilité, x représente les valeurs prises par n bits consécutifs de x_1 au cours du temps, à l'entrée de Π et y représente les valeurs prises par n bits consécutifs de x_2 au cours du temps, à la sortie de Π .

Soient n_1, n_2 deux indices de $[1, n]$ dont on cherche à distinguer les images par σ en observant les $(y^i)_{i>0}$ et $(x^i)_{i>0}$. Supposons tout d'abord que

$$\forall i > 0, \quad x^i(n_1) = x^i(n_2),$$

alors d'après l'équation (4.1), $\exists \{m_1, m_2\} \in \{\sigma(n_1), \sigma(n_2)\}$ tels que

$$\forall i > 0, \quad y^i(m_1) = y^i(m_2).$$

L'observation des $(x^i)_{i>0}$ et $(y^i)_{i>0}$ ne nous donne aucune information sur la valeur de m_1 et m_2 . En revanche, si $\exists i$ tel que

$$x^i(n_1) \neq x^i(n_2),$$

alors $y^i(m_1) \neq y^i(m_2)$ et $x^i(n_1) = y^i(m_l)$, $l \in \{1, 2\}$ implique $\sigma(n_1) = m_l$ et $\sigma(n_2) = m_{\bar{l}}$. On en déduit que pour retrouver σ à partir des $(x^i)_{i>0}$ et $(y^i)_{i>0}$, il faut et il suffit que pour chaque couple (n_a, n_b) , $a, b = 1 \dots n$ et $a \neq b$, il existe au moins un indice i_{ab} tel que

$$x^{i_{ab}}(n_a) \neq x^{i_{ab}}(n_b).$$

Le nombre minimal de couples à observer pour qu'une telle condition soit vérifiée est $\max\{i_{ab}\}$. Dans le cas binaire, pour chaque i , un maximum de $\frac{n}{2}$ bits de x^i peuvent prendre des valeurs différentes. Dans le meilleur des cas, il faudra observer $\log_2 n$ couples d'entrée et de sortie. Nous en déduisons la proposition suivante.

Théorème 4.1 *Un algorithme optimal de reconstruction d'entrelaceurs en blocs linéaires de profondeur n requiert $\mathcal{O}(\log_2 n)$ couples d'entrée et sortie de n bits chacune.*

Nous avons considéré jusqu'à maintenant les $(x^i)_{i>0}$ et $(y^i)_{i>0}$ comme deux suites infinies de mots de n bits issues d'une source binaire sans mémoire. Les variables aléatoires X_j et Y_j pour $j = 1 \dots n$, à valeur dans $\{0, 1\}$ dont les valeurs successives sont respectivement les $x^i(j)$ et les $y^i(j)$, sont donc indépendantes. Les suites $(x^i)_{i>0} = (x^i(1), \dots, x^i(n))_{i>0}$ et $(y^i)_{i>0} = (y^i(1), \dots, y^i(n))_{i>0}$ peuvent alors être vues comme n suites infinies $x(j) = (x^i(j))_{i>0}$ et $y(j) = (y^i(j))_{i>0}$, $j = 1 \dots n$, produites par n sources binaires sans mémoire et indépendantes deux à deux. Ces suites sont appelées *mots de longueur infinie*. En pratique, l'observateur n'a bien évidemment qu'une vision limitée et donc finie de ces suites. Au $i^{\text{ème}}$ top d'horloge, celui-ci observe x^i et y^i et complète chacun des n mots en lui adjoignant sa $i^{\text{ème}}$ lettre (symbole binaire). D'après ce qui précède, pour retrouver σ à partir des $(x^i)_{i>0}$ et $(y^i)_{i>0}$, il faut et il suffit que pour chaque couple de mots $(x(n_a), x(n_b))$, $a, b = 1 \dots n$ et $a \neq b$, il existe au moins une lettre à partir de laquelle les mots $x(n_a)$ et $x(n_b)$ diffèrent.

Le problème de reconstruction des entrelaceurs blocs linéaires est alors équivalent à la classification de mots évolutifs. Pour répondre à ce problème nous introduisons la notion de *trie*, qui est une structure dynamique adaptée à la classification d'un ensemble de mots évolutifs. Pour une présentation plus détaillée de ce problème, le lecteur pourra se référer à [53, 52, 170]. Soit un alphabet $\mathcal{M}$ inclus dans $\mathbb{N}$, fini ou dénombrable et de cardinal $r \in \mathbb{N}^* \cup \{+\infty\}$. L'ensemble des mots infinis $\mathcal{M}^{\mathbb{N}}$ est défini par :

$$\mathcal{M}^{\mathbb{N}} = \{m = m_1 m_2 \dots \mid \forall j \geq 1, m_j \in \mathcal{M}\}$$

Définition 4.1 *Soient w et v de $\mathcal{M}^{\mathbb{N}}$ deux mots infinis, $u \in \mathcal{M}^l$ un mot fini tels que*

$$w = u.v,$$

alors u est appelé préfixe de w et v suffixe de w .

On introduit par ailleurs, les applications *tête* et *queue* notées respectivement $\underline{T}$ et $\underline{Q}$ définies par

$$\begin{aligned} \underline{T} : \quad \mathcal{M}^{\mathbb{N}} &\rightarrow \mathcal{M} \\ m = m_1 m_2 \dots &\longrightarrow \underline{T}(m) = m_1 \\ \\ \underline{Q} : \quad \mathcal{M}^{\mathbb{N}} &\rightarrow \mathcal{M}^{\mathbb{N}} \\ m = m_1 m_2 \dots &\longrightarrow \underline{Q}(m) = m_2 m_3 \dots \end{aligned}$$

De la même façon, on introduit la notation $\underline{Q}_{[\alpha]}$ avec $\alpha \in \mathcal{M}$ pour désigner la restriction de l'application $\underline{Q}$ aux mots commençant par le symbole α .

$$\begin{aligned} \underline{Q}_{[\alpha]} : \quad \alpha.\mathcal{M}^{\mathbb{N}} &\longrightarrow \mathcal{M}^{\mathbb{N}} \\ m = \alpha m_2 \dots &\longrightarrow \underline{Q}_{[\alpha]}(m) = m_2 m_3 \dots \end{aligned}$$

La structure se construit récursivement en regroupant dans un même sous-arbre les mots qui possèdent le même préfixe. Elle sépare donc les mots selon leur préfixe et devient constante lorsque tous les mots sont distingués.

Définition 4.2 Soit X un ensemble de mots infinis. On construit récursivement le trie associé à X , noté $\text{trie}(X)$ de la façon suivante :

- si $|X| = 0$ alors $\text{trie}(X) = \emptyset$,
- si $|X| = 1$ $X = \{x\}$ alors $\text{trie}(X)$ est le nœud externe étiqueté par x ,
- si $|X| \geq 2$ alors :

$$\text{trie}(X) = \langle \bullet, \{\text{trie}(\underline{Q}_{[m]}(X))\}_{m \in \mathcal{M}} \rangle$$

Où $\bullet$ désigne le nœud interne.

Le trie de la figure 4.1 permet de distinguer 8 vecteurs binaires a à h , de longueur 5. Les bits utiles à la séparation des vecteurs apparaissent en gras. L'arbre sépare les vecteurs selon que leur premier bit vaut 0 (à gauche) ou 1 (à droite), puis d'après le second bit etc. Nous définissons maintenant les paramètres relatifs aux tries qui nous sont indispensables

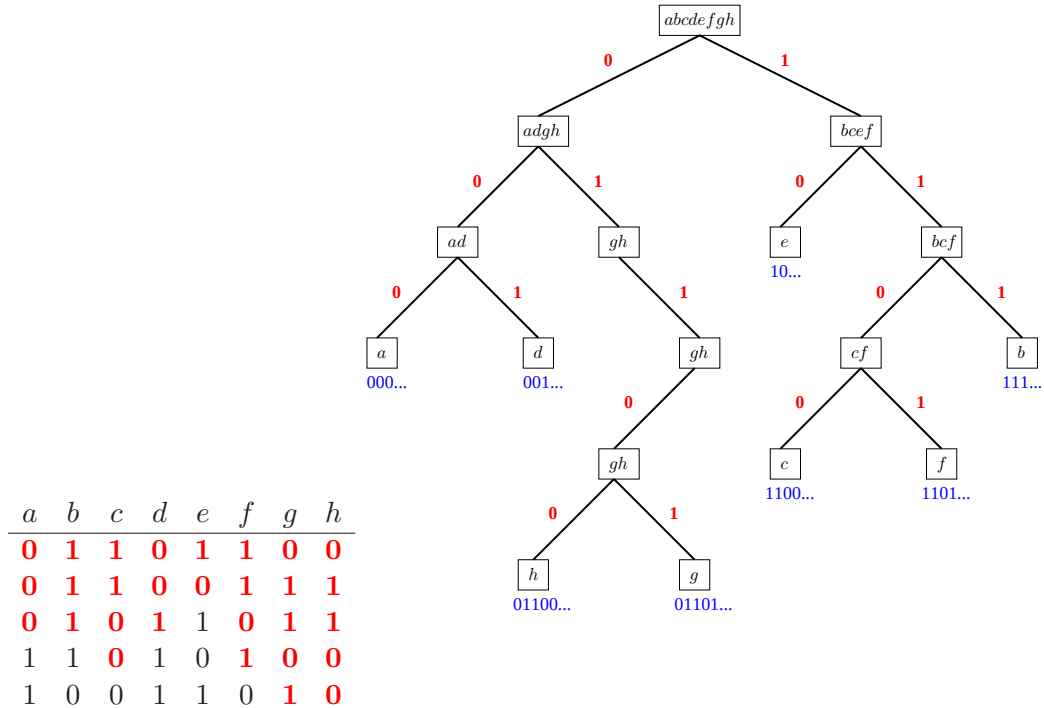


FIG. 4.2 – Exemple de trie fini

pour l'étude de complexité des algorithmes de reconstruction.

- La hauteur H du trie est la longueur maximale d'une branche du trie. Elle correspond au nombre minimum de comparaisons qu'il faut effectuer pour distinguer deux mots. C'est la longueur du plus long préfixe commun à deux mots.
- la taille N du trie est le nombre de nœuds internes du trie.
- la longueur de cheminement externe L est la somme des profondeurs des nœuds externes. Elle correspond au nombre de comparaisons nécessaires pour distinguer tous les mots.

Notons $X = \{x(j) \mid j = 1 \dots n\}$ et $Y = \{y(j) \mid j = 1 \dots n\}$ les ensembles de n mots infinis. D'après ce qui précède, pour retrouver σ à partir des $(x^i)_{i>0}$ et $(y^i)_{i>0}$, il faut et il suffit que les feuilles du trie associé à X soient étiquetées par un unique mot. À chaque top d'horloge l'observateur reçoit une nouvelle lettre pour chacun des n mots. Il remet ensuite à jour le trie associé à X et lorsque toutes les feuilles du trie sont étiquetées par un unique mot alors la σ peut être reconstruite de manière univoque.

4.2 Algorithmes de reconstruction

Dans ce chapitre, nous présentons les algorithmes de reconstruction d'un entrelaceur bloc connaissant des couples d'entrée et sortie. Nous nous intéressons dans un premier temps au cas où les entrées et sorties sont sans erreur. L'algorithme proposé consiste à construire les tries dynamiques pour les entrées et sorties. Ces deux tries sont isomorphes puisque les mots obtenus en sortie correspondent à une permutation des mots en entrée. Lorsque toutes les feuilles des tries sont finalement étiquetées par un unique mot, σ est retrouvé de façon univoque. En effet, σ est par définition l'isomorphisme qui permet de déduire le trie des sorties en fonction de celui des entrées. Dans un deuxième temps, nous traitons le cas où les entrées et sorties possèdent un certain nombre de bits erronés. Pour cela, nous remarquons que le bit j de x^i est égal au bit $\sigma(j)$ de y^i sauf dans le cas où l'un des deux est erroné. Si l'erreur n'est pas trop importante, ce cas apparaît peu fréquemment. À partir de cette remarque, nous proposons un algorithme d'élections qui attribue à chaque couple d'indices (p, q) un vote $+1$ lorsque $x^i(p) = y^i(q)$ et -1 sinon. Nous définissons alors σ par les couples $(p, q = \sigma(p))$ vainqueurs des élections.

4.2.1 Canal sans erreur

Le principe au cœur des algorithmes de reconstruction des entrelaceurs est la conservation du poids de Hamming. Soit à déterminer la valeur de $\sigma(i)$ pour un i donné et (x^l, y^l) un couple d'observations. D'après la relation (4.1) page 117,

$$\sigma(i) \in \{j \mid y^l(j) = x^l(i)\}.$$

Après k observations,

$$\sigma(i) \in \bigcap_{l=1}^k \{j \mid y^l(j) = x^l(i)\}.$$

Le fait que les observations soient issues de sources indépendantes nous assure que

$$\bigcap_{l=1}^k \{j \mid y^l(j) = x^l(i)\} \xrightarrow[k \rightarrow \infty]{} \{\sigma(i)\}.$$

Le principe est alors d'attendre le minimum d'observations pour que les limites des intersections correspondantes à chacun des indices $i = 1 \dots n$ soient atteintes. Cela équivaut donc à construire itérativement pour chaque indice i l'ensemble des indices des mots $y(j)$ de même préfixe que $x(i)$. Ceci revient alors à construire un trie $\text{trie}(X)$, respectivement $\text{trie}(Y)$, à partir des mots infinis, $x(j)$, resp. $y(j)$ pour $j = 1 \dots n$, issus des observations. Enfin, puisque les mots $y(j)$ correspondent à une permutation des mots $x(j)$, les tries sont identiques. À une feuille dans $\text{trie}(X)$ étiquetée par i correspond donc la même feuille dans $\text{trie}(Y)$ étiquetée par $\sigma(j)$.

Exemple 4.1

Soit à reconstruire un entrelaceur défini sur $\mathbb{F}_2^4$ par la donnée de $\sigma : (1, 2, 3, 4) \longrightarrow (3, 4, 2, 1)$. Les observations étant les suivantes : $(x^1, y^1) = (0110, 0101)$, $(x^2, y^2) = (0011, 1100)$, $(x^3, y^3) = (1010, 0110)$, $\dots$. Nous représentons ces observations dans un tableau, où t représente les *tops d'horloge*, afin de mettre en évidence dans les colonnes les différents mots infinis $x(j)$ et $y(j)$ pour $j = 1 \dots 4$.

t	$x(1)$	$x(2)$	$x(3)$	$x(4)$	$y(1)$	$y(2)$	$y(3)$	$y(4)$
1	0	1	1	0	0	1	0	1
2	0	0	1	1	1	1	0	0
3	1	0	1	0	0	1	1	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Les tries $\text{trie}(X)$ et $\text{trie}(Y)$ correspondants sont représentés sur la figure 4.3.

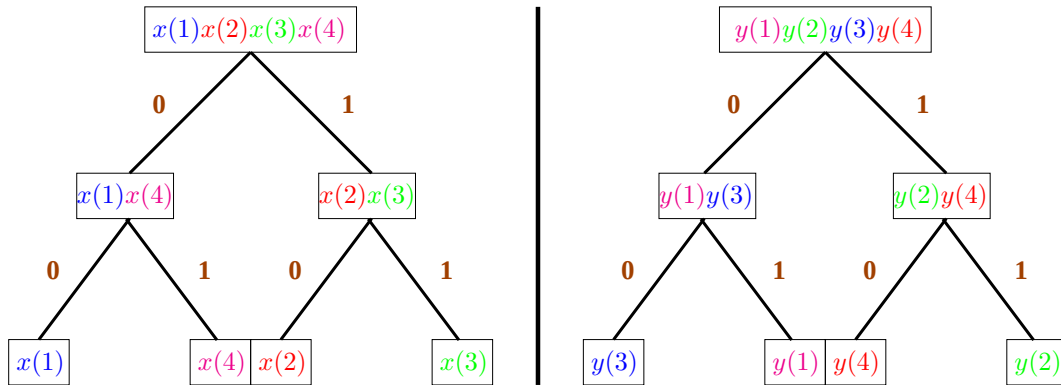


FIG. 4.3 – $\text{trie}(X)$ et $\text{trie}(Y)$

À la feuille de $\text{trie}(X)$ étiquetée par $x(1)$ correspond la feuille de $\text{trie}(Y)$ étiquetée par $y(3)$; nous en déduisons $\sigma(1) = 3$. De la même façon, nous concluons $\sigma(4) = 1$, $\sigma(2) = 4$ et $\sigma(3) = 2$. L'entrelaceur est donc reconstruit.

Dans la structure de trie, seules nous intéressent les feuilles et leur position relative; il n'est donc pas nécessaire de garder trace de l'ensemble du trie. D'autre part, nous pouvons remarquer qu'à chaque étape de la construction des tries, c'est-à-dire pour chacune des profondeurs du trie, les vecteurs sont classés dans l'ordre lexicographique (ou ordre inverse selon la convention). L'algorithme peut donc maintenir seulement l'état des feuilles

du trie à un instant donné ; celui-ci s'arrêtant lorsque à une étape toutes les feuilles sont étiquetées par un seul mot.

L'algorithme de reconstruction est le suivant.

- Nous regroupons tout d'abord dans ensembles identiques tous les indices i de 1 à n .
- À chaque nouvelle observation, nous formons une liste L_x triée d'ensembles d'indices i tels que les mots $x(i)$ possèdent le même préfixe.
- Nous construisons de même une liste triée L_y d'ensembles d'indices i tels que les mots $y(i)$ possèdent le même préfixe.
- L'algorithme s'arrête alors lorsque tous les ensembles de L_x sont des singletons. Il suffit ensuite de définir σ par

$$\forall i = 1 \dots n, \quad \sigma(L_x(i)) = L_y(i).$$

La relation d'ordre pour trier les listes d'ensembles est la suivante.

Définition 4.3 Soient un ensemble de mots finis $W = \{w_i \mid i = 1 \dots n\}$ et $\mathcal{E}_1, \mathcal{E}_2$ deux ensembles d'entiers inférieurs à n tels que pour tout entier i d'un même ensemble $\mathcal{E}_j$, $w_i = W_j$, $j \in \{1, 2\}$, où W_j est un mot fini. Nous définissons une relation d'ordre, $\preceq_{\mathcal{E}}$ portant sur les ensembles $\mathcal{E}_1$ et $\mathcal{E}_2$ par

$$\mathcal{E}_1 \preceq_{\mathcal{E}} \mathcal{E}_2 \text{ si et seulement si } W_1 \preceq W_2,$$

où $\preceq$ est la relation d'ordre lexicographique.

Exemple 4.2

En reprenant l'exemple précédent et en triant les mots infinis par ordre lexicographique nous obtenons le tableau suivant

t	$x(1)$	$x(4)$	$x(2)$	$x(3)$	$y(3)$	$y(1)$	$y(4)$	$y(2)$
1	0	0	1	1	0	0	1	1
2	0	1	0	1	0	1	0	1
3	1	0	0	1	1	0	0	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

En déroulant l'algorithme nous calculons L_x et L_y .

t	L_x	L_y
0	$L_x = (\{1, 2, 3, 4\})$	$L_y = (\{1, 2, 3, 4\})$
1	$L_x = (\{1, 4\}, \{2, 3\})$	$L_y = (\{1, 3\}, \{2, 4\})$
2	$L_x = (\{1\}, \{4\}, \{2\}, \{3\})$	$L_y = (\{3\}, \{1\}, \{4\}, \{2\})$

Nous concluons $\sigma(1) = 3$, $\sigma(4) = 1$, $\sigma(2) = 4$ et $\sigma(3) = 2$. L'entrelaceur est donc reconstruit.

Pour implémenter cet algorithme, nous avons seulement besoin de maintenir à jour trois listes de n éléments. La première contient les indices i de 1 à n dans l'ordre imposé par l'algorithme, la deuxième contient les images par σ de ces indices et enfin la dernière maintient les limites entres les ensembles. L'algorithme de reconstruction est résumé dans la figure 4.5 page 124 et la procédure de maintien des listes d'indices dans la figure 4.4 page ci-contre.

Procédure QuickCut

Entrées : une liste L de n entiers,
 m un mot binaire de n bits,
 min , max deux entiers inférieurs à n .

Sortie : L' liste de n entiers.
 cut un entier, $min \leq cut \leq max$

```

1   $E_0 \leftarrow \emptyset$ 
2   $E_1 \leftarrow \emptyset$ 
3  Pour  $i$  de  $min$  à  $max$ 
4    Si  $m(L(i)) = 1$ 
5       $E_1 \leftarrow E_1 \cup \{L(i)\}$ 
6    Sinon
7       $E_0 \leftarrow E_0 \cup \{L(i)\}$ 
8    Fin Si
9  Fin Pour
10  $n_0 \leftarrow \text{Card } E_0$ 
11 Si  $n_0 = 0$  ou  $n_0 = n$ 
12   Renvoyer  $L, 0$ 
13 Fin Si
14 Pour  $i$  de 0 à  $n_0 - 1$ 
15    $L(min + i) \leftarrow E_0(i + 1)$ 
16 Fin Pour
17 Pour  $i$  de 0 à  $\text{Card } E_1 - 1$ 
18    $L(min + n_0 + i) \leftarrow E_1(i + 1)$ 
19 Fin Pour
20 Renvoyer  $L, n_0$ 
```

FIG. 4.4 – Procédure de mise à jour de la liste des indices i de 1 à n

4.2.2 Canal avec erreur

Dans le cas d'un canal avec erreur, la conservation du poids entre x^i et y^i n'est plus assurée. En effet, si le canal introduit une erreur sur le $j^{\text{ème}}$ bit de x^i ou sur le $\sigma(j)^{\text{ème}}$ bit de y^i alors la relation (4.1) page 117 n'est plus vérifiée et nous avons

$$y^i(j) \neq x^i(\sigma^{-1}(j)).$$

Il est facile de voir que lorsque le canal introduit une erreur sur $x^i(\sigma^{-1}(j))$ et en même temps sur $y^i(j)$ alors les erreurs se compensent et la relation (4.1) page 117 reste vérifiée pour l'indice j . Dans cette étude, nous étudions le comportement de l'algorithme de reconstruction en considérant **l'erreur effective**, c'est-à-dire l'erreur résiduelle après compensation. Dans ce contexte, cela est équivalent à considérer les $(x^i)_{i>0}$ non bruités et les $(y^i)_{i>0}$ bruités par un canal binaire symétrique de TEB ε . Sous ces hypothèses, nous avons

**Algorithme de reconstruction des
entrelaceur blocs linéaires
- cas sans erreur -**

Entrées : (x^i, y^i) couples d'entrée et sortie de l'entrelaceur n de bits.

Sortie : liste des images respectives des indices $1, \dots, n$.

```

1   $L_x(j), L_y(j) \leftarrow j, \quad j = 1 \dots n$ 
2   $Cut(1) \leftarrow 1$ 
3   $Cut(2) \leftarrow n$ 
4   $i \leftarrow 0$ 
5   $n_c \leftarrow 2$ 
6  Tant que  $n_c \neq n + 1$ 
7     $i \leftarrow i + 1$ 
8    obtenir  $(x^i, y^i)$ 
9    Pour  $j$  de 1 à  $n_c - 1$ 
10      $L_x, c \leftarrow QuickCut(L_x, x^i, Cut(j), Cut(j + 1) - 1)$ 
11     Si  $c = 0$ 
12       Retour à l'étape 9
13     Fin Si
14      $L_y, c \leftarrow QuickCut(L_y, y^i, Cut(j), Cut(j + 1) - 1)$ 
15     Insérer  $c$  dans  $Cut$ 
16      $n_c \leftarrow n_c + 1$ 
17   Fin Pour
18 Fin Tantque
19 Pour  $j$  de 1 à  $n$ 
20    $\sigma(L_x(j)) \leftarrow L_y(j)$ 
21 Fin Pour
22 Renvoyer  $\sigma$ 

```

FIG. 4.5 – Algorithme de reconstruction des entrelaceurs dans le cas sans erreur

$$\mathcal{Pr}(y^i(j) = x^i(\sigma^{-1}(j))) = 1 - \varepsilon \quad \text{et} \quad \mathcal{Pr}(y^i(j) \neq x^i(\sigma^{-1}(j))) = \varepsilon. \quad (4.2)$$

En d'autres termes, la relation (4.1) page 117 reste presque toujours vérifiée sauf un petit nombre de fois qui correspond à l'introduction d'erreur résiduelle sur $y^i(j)$. Pour chaque couple d'indices i et j , $i, j = 1 \dots n$, nous allons faire une hypothèse $\mathcal{H}_i(j)$ définie par « $j = \sigma(i)$ ». Évidemment, seules les hypothèses $\mathcal{H}_i(\sigma(i))$ sont vraies. Pour distinguer les hypothèses vraies des autres, nous utilisons la relation (4.2). En effet, à chaque nouvelle observation (x^l, y^l) , nous attribuons un vote pour chaque hypothèses de la manière suivante. Pour chaque couple d'indices $i, j = 1 \dots n$, nous attribuons un vote +1 à l'hypothèse $\mathcal{H}_i(j)$ si et seulement si $x^l(i) = y^l(j)$. Dans le cas contraire, le vote attribué est -1. Notons maintenant $V_{ij}^{(k)}$ la variable aléatoire représentant le nombre de votes obtenus

par l'hypothèse $\mathcal{H}_i(j)$ après k observations. L'étude de ces n^2 variables aléatoires à valeurs dans $[-k, k]$ permet de discriminer les bonnes hypothèses des autres.

En effet, les lois de probabilités suivies par les $V_{ij}^{(k)}$ ne sont pas identiques suivant que $j = \sigma(i)$ ou non. Les $x^{(i)}$, $i = 1 \dots n$, sont des mots infinis générés par n sources binaires sans mémoire et indépendantes deux à deux. On en déduit que

$$\mathcal{P}r(x^l(i) = x^l(j)) = \frac{1}{2} \quad \forall l \text{ et } i \neq j.$$

La variable aléatoire représentant le nombre de fois qu'un tel événement survient en k observations suit une distribution binomiale de paramètres k et $\frac{1}{2}$. De même

$$\mathcal{P}r(x^l(i) = y^l(j)) = \frac{1}{2} \quad \forall l \text{ et } j \neq \sigma(i).$$

La variable aléatoire représentant le nombre de fois qu'un tel événement subvient en k observations, suit une loi binomiale de paramètre k et $\frac{1}{2}$. On en conclut que quel que soit i et quel que soit $j \neq \sigma(i)$ la variable aléatoire $V_{ij}^{(k)}$ suit une distribution binomiale de paramètre $\frac{1}{2}$ centrée en 0.

Dans le cas $j = \sigma(i)$, $V_{i\sigma(i)}^{(k)}$ suit une loi binomiale de paramètre $(1 - 2\varepsilon)$ d'après la relation (4.1) page 117. Lorsque le nombre d'observations augmente, un *pic* de votes apparaît pour les bonnes hypothèses ; pour les autres, le cumul des votes avoisine 0.

Le principe de l'algorithme de reconstruction des entrelaceurs blocs linéaire de profondeur n est le suivant. Pour chaque nouvelle observation (x^l, y^l) nous attribuons les votes pour chaque hypothèse conformément aux règles prédéfinies. Les votes sont maintenus dans la matrice cumulative des votes, V , pour laquelle le coefficient $[V]_{ij}$ correspond au nombre de votes cumulés pour l'hypothèse $\mathcal{H}_i(j)$. Puis nous recherchons dans la matrice des votes la présence de *grand maxima*, dont la définition est la suivante.

Définition 4.4 Soit V la matrice cumulative des votes. Les coefficient $[V]_{ij} = M$ est appelé *grand maxima* des V si et seulement si

- $\forall l = 1 \dots n, \quad [V]_{lj} \leq M,$
- $\forall l = 1 \dots n, \quad [V]_{il} \leq M,$
- $M - \max_{l \neq j} [V]_{il} \geq L,$
- $M - \max_{l \neq i} [V]_{lj} \geq L,$

où L est un paramètre de l'algorithme appelé *facteur de précision*.

Remarque 4.1 :

La propriété de grand maxima est un prolongement naturel de la notion de seuil pour discriminer un processus aléatoire suivant une certaine loi de probabilité, ici normale centrée en $(1 - 2\varepsilon)k$ et $2(n - 1)$ autres suivant la même loi, ici normale centrée en 0.

Si un tel grand maxima $[V]_{ij}$ est trouvé dans la matrice des votes alors l'algorithme décide que l'hypothèse $\mathcal{H}_i(j)$ est vraie et que pour tout $l \neq i$, $\mathcal{H}_l(j)$ est fausse ainsi que

$\mathcal{H}_i(l)$ pour tout $l \neq j$. L'algorithme s'arrête lorsque toutes les hypothèses ont été décidées. L'algorithme de reconstruction des entrelaceurs blocs linéaires dans le cas avec erreurs est résumé dans la figure 4.6 page ci-contre.

Quand un coefficient $[V]_{ij}$ de la matrice de vote est un grand maxima cela signifie que l'indice j a reçu le plus de voix pour l'élection de l'image de i par σ et que l'écart de voix avec son premier concurrent dépasse le seuil L . De même, cela signifie que l'indice i a reçu le plus de voix dans l'élection de l'antécédent de j par σ et que l'écart de voix avec son premier concurrent dépasse le seuil L .

4.3 Analyse des algorithmes

4.3.1 Canal sans erreur

L'algorithme de reconstruction des entrelaceurs blocs linéaires dans le cas sans erreur s'arrête lorsque qu'il comptabilise $(n + 1)$ coupures entre les indices, ce qui correspond à n singletons. Ces singletons sont obtenus lorsque les mots infinis $x(j)$, pour $j = 1 \dots n$ sont distincts deux à deux. Le nombre moyen d'observations nécessaire pour distinguer de manière univoque ces n mots est donc égal au nombre moyen de lettres nécessaire à distinguer tous les mots deux à deux, c'est-à-dire à la hauteur moyenne du trie associé. Ce nombre moyen d'observations est donné par le théorème suivant.

Théorème 4.2 *Soient n sources dynamiques S_i , $i = 1 \dots n$, binaires sans mémoire et indépendantes deux à deux, générant respectivement les n mots infinis $x^{(i)}$. Si, quel que soit $i = 1 \dots n$, la probabilité que S_i émette un « 1 » est $\frac{1}{2}$, alors le trie associé aux $x^{(i)}$ est de hauteur moyenne*

$$\mathcal{O}(\log_2 n).$$

Preuve :

La preuve de ce théorème est donnée en annexe C page 261, en prenant $p = \frac{1}{2}$. ■

La démonstration du théorème 4.2 explicite une approximation de la probabilité d'arrêt de l'algorithme avant l'étape k , $\mathcal{P}_a^{(k)}$,

$$\lim_{k \rightarrow +\infty} \sup_{k \geq 0} |\mathcal{P}_a^{(k)} - \exp(-\frac{n^2}{2^{k+1}})| = 0.$$

Cette probabilité d'arrêt est illustrée par la figure 4.7 page 128.

D'autre part, la mise à jour des différentes listes s'effectue en $\mathcal{O}(n)$ opérations élémentaires. La complexité de l'algorithme de reconstruction d'un entrelaceur bloc linéaire de profondeur n , dans le cas d'un canal sans erreur est

$$\mathcal{O}(n \log_2 n).$$

**Algorithme de reconstruction des
entrelaceur blocs linéaires
- cas avec erreur -**

Entrées : (x^l, y^l) couples d'entrée et sortie de l'entrelaceur n de bits,
 L le facteur de précision.

Sortie : ensemble des couples $(i, \sigma(i))$ pour $i = 1, \dots, n$.

```

1   $Ant \leftarrow \{1, \dots, n\}$ 
2   $Im \leftarrow \{1, \dots, n\}$ 
3   $\sigma \leftarrow \emptyset$ 
4   $V \leftarrow 0$ 
5   $l \leftarrow 0$ 
6  Tant que Card  $(\sigma) \neq n$ 
7     $l \leftarrow l + 1$ 
8    obtenir  $(x^l, y^l)$ 
9    Pour tout  $i$  de  $Ant$ 
10     Pour tout  $j$  de  $Im$ 
11       Si  $x^l(i) = y^l(j)$ 
12          $[V]_{ij} \leftarrow [V]_{ij} + 1$ 
13       Sinon
14          $[V]_{ij} \leftarrow [V]_{ij} - 1$ 
15       Fin Si
16     Fin Pour
17   Fin Pour
18   Pour tout grand maxima  $[V]_{ij}$  de  $V$ 
19      $\sigma \leftarrow \sigma \cup \{(i, j)\}$ 
20      $Ant \leftarrow Ant \setminus \{i\}$ 
21      $Im \leftarrow Im \setminus \{j\}$ 
22      $V \leftarrow V$  privé de sa  $i^{\text{ème}}$  ligne et  $j^{\text{ème}}$  colonne
23   Fin Pour
24 Fin Tantque
25 Renvoyer  $\sigma$ 

```

FIG. 4.6 – Algorithme de reconstruction des entrelaceurs dans le cas avec erreur

4.3.2 Canal avec erreur

L'analyse de l'algorithme de reconstruction dans le cas d'un canal sans erreur ne nous a malheureusement pas permis de mettre en évidence une expression exacte de la probabilité de succès et du nombre moyen d'observations nécessaire. Le problème est pour l'instant encore ouvert. Néanmoins, nous avons une bonne approximation du nombre moyen de couples nécessaires en fonction du TEB. Cette approximation a été validée expérimentalement (cf. paragraphe 4.4). Le nombre moyen de couples d'entrée et de sortie d'un entrelaceur bloc

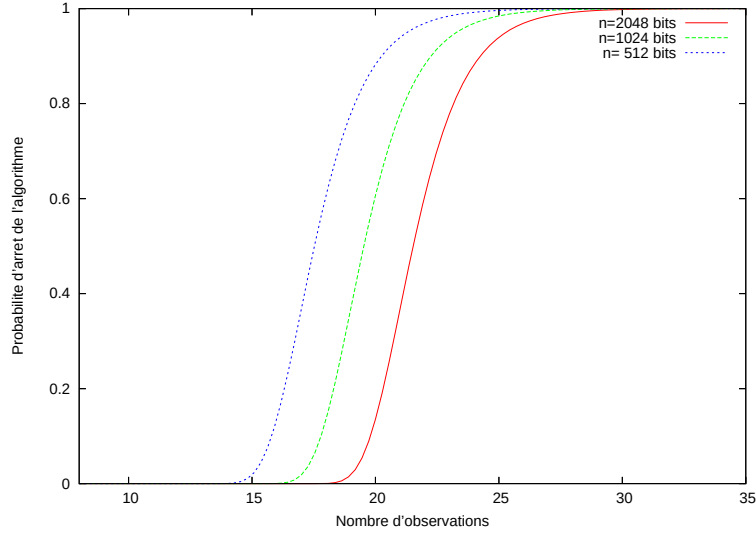


FIG. 4.7 – Probabilité d'arrêt de l'algorithme de reconstruction dans le cas sans erreur

linéaire dans le cas d'un canal bruité de TEB ε est environ de

$$\mathcal{O}\left(\frac{\log n}{(\frac{1}{2} - \varepsilon)^2}\right),$$

La mise à jour des votes à chaque observation ainsi que la recherche d'un grand maxima dans la matrice des votes nécessitent $\mathcal{O}(n^2)$ opérations élémentaires. La complexité de l'algorithme est

$$\mathcal{O}\left(\frac{n^2 \log n}{(\frac{1}{2} - \varepsilon)^2}\right).$$

4.4 Résultats expérimentaux

Les simulations présentées dans ce paragraphe ont été effectuées en simulant des sources dynamiques indépendantes émettant des 0 et des 1 avec même probabilité dans un canal binaire symétrique (CBS) de paramètre ε . Le résultat de ces simulations sont complétées pour d'autres types de sources et de canaux dans [8]. Les tests ont été effectués sur un PIII 533 Mhz avec 100 essais pour chaque configuration. Dans les cas avec erreurs, afin de détecter les échecs, une borne par défaut de 600 couples a été fixée avant de conclure à un échec.

% d'erreur	précision	nb moyen de couples	tps moyen	nb d'échecs
0	-	18	0.0003s	0
10	15	69	6.88s	0
20	30	154	12.08s	0
30	40	346	23.15s	0

TAB. 4.1 – Simulations dans un CBS $n = 512$ bits

% d'erreur	précision	nb moyen de couples	tps moyen	nb d'échecs
0	-	20	0.01s	0
10	15	74	42.5s	0
20	30	164	67.05s	0
30	40	367	116.5s	0

TAB. 4.2 – Simulations dans un CBS $n = 1024$ bits

% d'erreur	précision	nb moyen de couples	tps moyen	nb d'échecs
0	-	22	0.0243s	0
10	15	79	263.04s	0
20	30	175	365.45s	0
30	40	396	588.51s	0

TAB. 4.3 – Simulations dans un CBS $n = 2048$ bits**Remarque 4.2 :**

Les valeurs observées lors des simulations sont peu dispersées ; l'écart type très proche de 0. D'autre part, si on prend un taux d'erreur égal à 0 et une précision de 10, on retrouve un nombre d'observations proche de celui de l'algorithme sans erreur.

Les résultats montrent que moyennant le choix d'un bon paramètre de précision, l'algorithme retrouve avec forte probabilité la permutation, quel que soit le contexte d'erreur, en peu de temps et avec peu d'observations. Les simulations présentées dans le tableau 4.4 exhibent de « bonnes » valeurs du facteur de précision dans un voisinage de la limite d'erreur de 0.5. Elles ont été effectuées sur PIII 500 Mhz avec 100 essais pour chaque configuration.

% d'erreur	précision	nb moyen de couples	tps moyen	nb d'échecs
40	50	1 114	12.78s	0
44	70	2 552	28.54s	0
46	130	6 341	64,38s	0
48	410	31 729	380s	0

TAB. 4.4 – Simulations dans un CBS $n = 256$ bits

La courbe 4.8 illustre le peu d'observations nécessaires par rapport aux 2^n valeurs que peut produire la source, même dans des environnements extrêmement bruités. Nous validons expérimentalement le comportement asymptotique du nombre moyen de couples d'entrée et de sortie nécessaires annoncé au paragraphe 4.3.2 page ci-contre. La figure 4.8 illustre la concordance entre l'expérimentation et la courbe théorique pour $n = 256$ bits. L'équation intuitée est

$$\frac{10}{(\frac{1}{2} - \varepsilon)^2}.$$

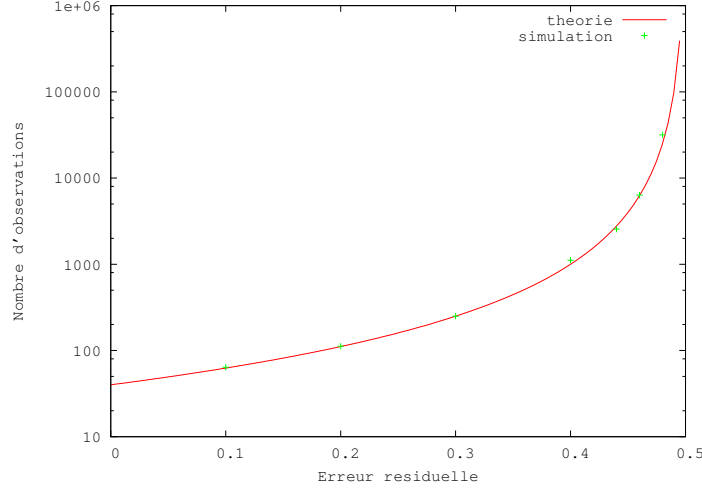


FIG. 4.8 – Évolution du nombre d'observations en fonction de l'erreur pour $n = 256$

4.5 Conclusion et perspectives

Nous avons présenté dans ce chapitre des techniques permettant de retrouver des entrelaceurs de taille quelconque grâce à des algorithmes de reconstruction de permutation sur $\mathbb{F}_2^n$. Ces algorithmes se généralisent de façon naturelle à n'importe quel alphabet $\mathcal{M}$ différent de $\mathbb{F}_2$. L'efficacité de tels algorithmes croît avec $\text{Card } \mathcal{M}$. En effet, le cas favorable pour différencier n mots de longueur infinie se présente lorsqu'à chaque nouvelle lettre observée, les groupes de l mots ayant le même préfixe sont divisés en sous-groupe de même taille, $l/\text{Card } \mathcal{M}$. Dans le cas optimal, le nombre de couples d'entrée et de sortie nécessaires est en $\mathcal{O}(\log(n)/\log(\text{Card } \mathcal{M}(n)))$. Les résultats mis en avant dans ce chapitre constituent donc une borne inférieure pour ceux espérés dans le cas de sources non binaires.

Ces algorithmes de reconstruction permettent ainsi de reconstruire des entrelaceurs dans un contexte d'erreurs largement supérieur à celui des canaux de transmissions habituels et dans un intervalle de temps acceptable pour de nombreuses applications. La faible quantité d'observations nécessaire à la reconstruction ainsi que la robustesse aux erreurs rendent ces deux derniers efficaces. Ils sont d'autant plus efficaces dans le cadre de la reconstruction des turbo-codes car l'erreur prise en compte n'est plus celle introduite par le canal mais l'erreur moyenne résiduelle après décodage de l'entrée et de la sortie par des décodeurs convolutifs. Nous nous trouvons donc dans un contexte favorable pour la reconstruction de l'entrelaceur, l'entrée et la sortie étant faiblement bruitées.

Enfin, contrairement à la reconstruction des codes en blocs linéaires et des codes convolutifs, nous sommes dans la possibilité de reconstruire non seulement les paramètres des turbo-codes mais aussi de construire des décodeurs équivalents. La partie systématique du premier codeur convolutif permet de lever les deux indéterminés intrinsèques à la reconstruction des codes convolutifs et de retrouver les paramètres exactes de ce codeur. La connaissance de l'entrée du schéma entrelaceur-codeur convolutif nous permet de retrouver une matrice du codeur à une permutation près de ses colonnes. Enfin, la reconstruction de la permutation permet de trouver un schéma entrelaceur-codeur convolutif équivalent.

Conclusion et perspectives

*« Ne parlez pas dans la rue, il y a des oreilles
sous les pavés. »*

Proverbe chinois

DANS cette partie, nous avons abordé l'étude des canaux de communication dans un contexte non coopératifs sous l'angle des codes correcteurs d'erreurs. Dans ce domaine, notre contribution a été variée. Nous avons tout d'abord présenté une analyse très fine de l'algorithme de Sicot-Houcke [173] en utilisant une approche différente [19]. Cet algorithme est la brique de base qui s'est imposée comme socle pour concevoir les algorithmes de reconstruction des codes convolutifs. Dans ce contexte, nous avons proposé [9] un nouvel algorithme de reconstruction des codes convolutifs dans le cas d'un canal sans erreur, de meilleure complexité que ceux existants [71, 162]. Nous avons aussi amélioré notablement la complexité théorique des techniques de reconstruction proposées par É. Filiol [71] et nous avons pu expliquer pourquoi les résultats expérimentaux qu'il obtenait étaient meilleurs que ceux annoncés en théorie. De plus, nous avons pu améliorer ses techniques en proposant un processus totalement automatisé. Cette amélioration est essentiellement due à l'approche algébrique qui nous a permis de simplifier sensiblement les équations existantes et d'en mettre en évidence de nouvelles [9]. Par ailleurs, nous avons pu démontré la conjecture d'É. Filiol sur le caractère canonique des matrices génératrices renvoyées par les algorithmes de reconstruction des codes convolutifs. Pour la reconstruction des codes en blocs linéaires et les codes convolutifs, nous nous sommes attachés à mettre en œuvre un formalisme unificateur, c'est-à-dire l'approche algébrique [19], en symétrie avec le formalisme algébrique unificateur proposé par G.D. Forney [87] et repris par R.J. McEliece [143]. Enfin, nous avons proposé une technique effective de reconstruction des turbo-codes et des entrelaceurs blocs [8, 14, 9]. Contrairement aux algorithmes de reconstruction des codes en blocs linéaires et des codes convolutifs, nous sommes en mesure, non seulement de reconstruire le code mais aussi de trouver un décodeur équivalent.

Les algorithmes de reconstruction tirent parti de la structure induite lors du codage. Cette même structure, nécessaire à la correction d'erreur signe fortement le type de schéma de codage utilisé et favorise les techniques de reconstruction. Nous avons apporté une réponse au problème de reconstruction des codes linéaires, des codes convolutifs et des turbo-codes. D'autres codes fortement structurés, BCH ou Reed-Solomon par exemple, semblent être aussi intéressants à étudier dans un contexte de reconstruction d'algorithme, notamment parce qu'ils sont largement déployés. Malheureusement, la complexité des algo-

algorithmes de reconstruction est exponentielle en la taille des codes à reconstruire mais aussi en l'erreur du canal. Aux vues des simulations, les algorithmes présentés sont néanmoins efficaces jusqu'à des niveaux de bruit de l'ordre de 10^{-3} . Il en résulte que sur des canaux dont le TEB est supérieur à ce seuil, la reconstruction est impossible en pratique tandis que les utilisateurs légitimes peuvent encore décoder. Le facteur limitant est bien le TEB du canal. L'approche de M. Cluzeau [51] consistant à corriger des erreurs lors du processus de reconstruction semble être prometteuse. Dans une autre direction, l'utilisation d'information souple dans les algorithmes de reconstruction devrait permettre de diminuer artificiellement le taux d'erreur binaire. En effet, s'il semble au premier abord difficile d'adapter l'algorithme de Gauss pour résoudre un système dont les coefficients sont souples, nous pouvons plus facilement imaginer sélectionner préférentiellement les équations les plus favorables afin d'obtenir un taux d'erreur dans le système inférieur à celui du canal. Dans un contexte non coopératif, l'observateur tire avantage du fait que les paramètres des codes correcteurs d'erreurs implantés dans un système de communication le sont pour plusieurs années. Même sur un canal un peu bruité, celui-ci peut attendre assez de temps pour obtenir suffisamment d'information. De plus, même si le processus de reconstruction prend plusieurs mois dans le cas le plus défavorable, il reste négligeable devant la durée de vie d'un système de communication. La reconstruction s'effectue une fois pour toute et reste valable durant toute la vie du système. Néanmoins, il reste un important travail de comparaison et de synthèse des techniques existantes afin tout d'abord de choisir la plus efficace en pratique mais aussi éventuellement afin de coupler le meilleur de chacune des approches.

Dans le cadre de cette étude, nous nous sommes placés dans le contexte le plus défavorable pour un observateur, c'est-à-dire le contexte non coopératif. Il en découle que sans information *a priori*, il n'est pas toujours possible de trouver un décodeur équivalent. Or, les indéterminées se lèvent facilement avec très peu d'information supplémentaire. Il est alors assez naturel de se poser la question de l'intérêt d'adapter les techniques de reconstruction de codeurs correcteurs d'erreurs au contexte coopératif. En effet, celui-ci propose un contexte favorable aux algorithmes de reconstruction et permet d'intégrer sans surcoût l'information nécessaire à lever les indéterminées éventuelles. Une première application peut consister à économiser l'envoi de nouveaux paramètres dans des systèmes changeant fréquemment de paramètres de système de codage [12]. Ces nouveaux paramètres sont alors retrouvés avec des algorithmes de reconstruction. Ces algorithmes peuvent être utilisés comme compromis entre l'économie de la bande passante et la charge de calcul au niveau émetteur et récepteur. Dans des systèmes où la latence n'est pas primordiale, on peut même imaginer coder de l'information supplémentaire dans le choix des paramètres de codage [12], par exemple dans le choix de l'entrelaceur d'un turbo-codeur parallèle. Les algorithmes de reconstruction lorsqu'ils nous permettent de retrouver exactement les paramètres initiaux peuvent aussi être vus comme des algorithmes de décodage. Dans un contexte coopératif, de tels algorithmes peuvent ainsi être intégrés comme part entière dans un schéma de codage de canal [13].

Deuxième partie

Étude de techniques de stéganalyse



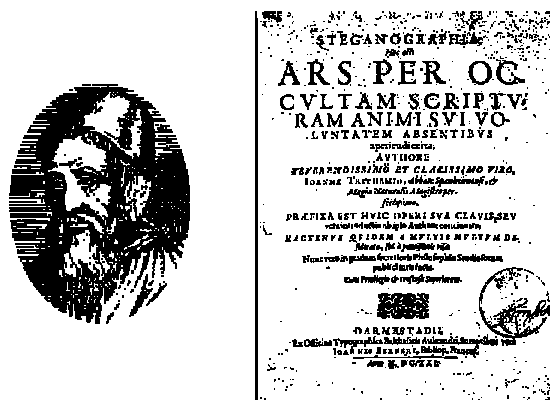
Introduction

« *J'aurais voulu être un espion, mais il fallait avaler des micro films et mon médecin me l'a interdit.* »

Woody Allen

S'IL est une discipline connue du grand public émulant l'imagination et la curiosité, c'est bien la cryptographie. Du code de César au « Da Vinci code », la cryptographie fascine ; tantôt elle est l'apanage des militaires et espions au secours de l'histoire ou d'amours impossibles, tantôt elle préoccupe les mathématiciens par les énigmes qu'elle offre et tantôt elle alimente les romans grand public [35, 36, 180]. Peut-être plus ancienne et souvent amalgamée à la cryptographie, la *stéganographie* vit dans l'ombre des « codes secrets », dissimulée derrière un objectif et un formalisme à la fois proches et différents de ceux de la cryptographie. Son étymologie grecque « *stego* », le secret et « *graphia* », l'écriture, l'enracine dans l'antiquité. La stéganographie est donc l'art de l'écriture secrète. Tout au long de l'histoire, elle tient au même titre que la cryptographie, une place importante dans des événements marquants. Ainsi, Hérodote relate, dans son œuvre *l'Enquête*, comment en 480 av. J.-C., Démarate réussit à prévenir les Grecs d'une invasion imminente du roi de Perse Xerxès 1^{er} en envoyant un message gravé dans le bois d'une tablette d'écriture recouverte de cire, d'apparence vierge. En 300 av. J.-C., Énée le tacticien dans ses *Mémoires sur la stratégie*, décrit le premier système stéganographique qui consiste à marquer d'un trou les lettres d'un texte constitutives d'un message. En Chine, la coutume veut que le signal de la révolte des chinois contre la dynastie mongole Yuan lors de la *fête de la lune*, a été donné par des messages cachés dans des *gâteaux de lune*. Plus technique, l'invention de l'encre sympathique est attribuée au naturaliste Plinie l'Ancien, romain du 1^{er} siècle av. J.-C. et est encore utilisée de nos jours. Les techniques de stéganographie devenant de plus en plus savantes, très tôt les premiers ouvrages traitant du sujet voient le jour à partir du XVI^e siècle. En 1499, l'abbé Jean Trithème (1462-1516) publie le premier traité de stéganographie, intitulé *Steganographia* et composé de trois livres qui ne livrèrent tous leurs secrets que récemment. Le troisième livre n'a été finalement « décodé » par Thomas Ernst qu'en 1996 et indépendamment par Jim Reeds [160] en 1998.

Un scientifique allemand, Gaspart Schott (1608-1666) explique dans son livre *Schola Steganographica* comment dissimuler des messages en utilisant des notes de musique. Souvent taxés d'ésotérisme, certains de ces ouvrages, à l'exemple de *Steganographia* ont été interdits en leur temps. Néanmoins, l'intérêt vif du public pour les sciences du secret a

FIG. 4.9 – Jean Trithème et *Steganographia*

rendu possible la diffusion de ces livres, sous le manteau. La littérature en tant que vecteur de diffusion d'information est elle-même support servant à dissimuler des messages. Qu'elle soit médiévale ou moderne, elle foisonne de figures de styles telles *l'acrostiche*¹ mais aussi d'exemples plus croustillants les uns que les autres. Les plus célèbres d'entre eux sont notamment le poème de Boccaccio (1313-1375), *Amorosa visione*, long d'environ 1500 vers et la correspondance privée entre George Sand et Alfred de Musset en 1883.

Les techniques se sont complexifiées avec le temps et l'invention du micro-film en 1857 par Sir Brewster, puis du micro-point, ont redonné un nouveau souffle à la stéganographie. Elles permettent ainsi de réduire des photos à la taille d'un point sur un *i* et de les dissimuler dans un texte. Ces techniques ont été largement employées par les militaires pendant les différentes guerres franco-allemandes mais aussi les services de renseignement. La stéganographie a aussi marqué de son empreinte l'histoire contemporaine, notamment celle de la France. Le *message de Verlaine*, diffusé en deux parties sur les ondes de la BBC le 5 juin 1944 à 21h15, « *Les sanglots longs des violons de l'automne* » et « *Blessent mon cœur d'une langueur monotone* », annonce le débarquement imminent des alliés. Plus tard, dans les années 80, Margaret Thatcher réussit à identifier la source de nombreuses fuites de documents en traçant ceux-ci à l'aide de techniques de dissimulation d'information. Enfin, plus récemment, de nombreux spécialistes relayés par les média [22, 116, 117, 176] avancent l'hypothèse selon laquelle Bin Laden aurait coordonné les attentats du 11 septembre 2001 en utilisant des messages cachés dans des images de sites à caractère pornographique. Le lecteur féru d'épistémologie trouvera son bonheur dans [111, 112, 113, 122, 159].

Paradoxalement, la stéganographie dite *moderne*, c'est-à-dire adaptée aux données numériques, est relativement jeune. En pleine expansion, elle suit depuis le milieu des années 90 un essor corrélé à celui d'Internet ; le nombre de conférences scientifiques proposant des sessions dédiées à la dissimulation d'information augmentant chaque année. Si on se reconnaît dans une communauté par les points communs que l'on partage avec ses membres, parler le même langage est un point de passage obligé. De ce fait, on peut situer avec bonne approximation la naissance de la communauté des *stéganographes* en 1996, lors de la première édition d'*Information Hiding* et l'adoption d'un corpus relatif à la dissi-

¹L'acrostiche est un poème dont les premiers mots, lettres ou syllabes de chaque vers forment un message.

George Sand à Alfred de Musset

Je suis très émue de vous dire que j'ai bien compris l'autre soir que vous aviez toujours une envie folle de me faire danser. Je garde le souvenir de votre baiser et je voudrais bien que ce soit là une preuve que je puisse être aimée par vous. Je suis prête à vous montrer mon affection toute désintéressée et sans calcul, et si vous voulez me voir aussi vous dévoiler sans artifice mon âme toute nue, venez me faire une visite. Nous causerons en amis, franchement. Je vous prouverai que je suis la femme sincère, capable de vous offrir l'affection la plus profonde comme la plus étroite en amitié, en un mot la meilleure preuve dont vous puissiez rêver, puisque votre âme est libre. Pensez que la solitude où j'habite est bien longue, bien dure et souvent difficile. Ainsi en y songeant j'ai l'âme grosse. Accourrez donc vite et venez me la faire oublier par l'amour où je veux me mettre.

La réponse d'Alfred de Musset

Quand je mets à vos pieds un éternel hommage
Voulez-vous qu'un instant je change de visage ?
Vous avez capturé les sentiments d'un cour
Que pour vous adorer forma le Créateur.
Je vous chéris, amour, et ma plume en délire
Couche sur le papier ce que je n'ose dire.
Avec soin, de mes vers lisez les premiers mots
Vous saurez quel remède apporter à mes maux.
Bien à vous, Éric Jarrigeon

La réponse de George Sand

Cette insigne faveur que votre cour réclame
Nuit à ma renommée et répugne mon âme.

FIG. 4.10 – Correspondance entre George Sand et Alfred de Musset

mulation d'information [153]. C'est d'ailleurs en 1997 qu'est soutenue l'une des premières thèses [48] dans le domaine. À la lumière de l'histoire de cette discipline, on s'aperçoit que très longtemps, la stéganographie est restée l'exclusivité de gens cultivés voir instruits. De nos jours, cela ne semble plus être le cas. En effet, Internet a fait tomber les barrières et offre à qui le veut des outils très performants et « prêts à l'emploi » en quelques clics. Entre 2004 et 2006, nous avons répertorié environ 120 outils de stéganographie facilement disponibles (cf. annexe D). Indépendamment de l'intérêt scientifique, l'étude de techniques de *stéganalyse*, c'est-à-dire des techniques visant à détecter la présence d'information cachée, a un impact certain dans le domaine de recherches informatiques [121], dans la lutte contre la pornographie infantile [4, 5, 161] et le terrorisme [1, 2]. Nous nous proposons dans cette étude, d'aborder la stéganographie sous l'angle du *stéganalyste* et présentons plusieurs algorithmes permettant de détecter avec une forte probabilité la présence d'information cachée dans des images fixes.

La première étape est bien évidemment de définir précisément l'objet que l'on va étudier. Le lecteur pourra se référer à d'excellents ouvrages [110, 114, 201] traitants de dissimulation d'information. Bien que la communauté des stéganographes se soit constituée dans les années 90, G.J. Simmons pose en 1983 le socle de la stéganographie moderne en définissant la notion de *canal subliminal*. Pour illustrer son propos, il reprend le *problème du prisonnier*. Le contexte général est le suivant. Soient Alice et Bob, deux protagonistes partageant un secret commun et désirant communiquer ensemble de façon « sécurisée » ; Wendy une amie indiscreète qui voudrait bien avoir accès au contenu de leur correspondance. Un premier moyen pour Alice et Bob de protéger leurs communications est d'utiliser

la cryptographie afin d'assurer notamment la confidentialité, l'intégrité, l'authenticité des messages qu'ils s'échangent. En employant la cryptographie, ils mettent ainsi en œuvre la *sécurité de communication* (COMSEC). Dans de nombreux cas de figure, cette seule protection est suffisante. Prenons maintenant l'exemple d'un agent infiltré dans une organisation mafieuse qui doit rester en contact avec un agent de liaison de la police. Dans ce cas très précis, les deux agents doivent évidemment protéger leurs communications afin qu'un tiers interceptant le message ne puisse apprendre aucune information. De plus, l'existence même de leurs communications, indépendamment de leur contenu, peut compromettre la couverture de l'agent infiltré. En effet, la présence du numéro de la police sur le portable d'un membre de la pègre le désignerait rapidement comme suspect. Ils doivent alors rendre furtif leur canal de transmission, en mettant en œuvre de la *sécurité de transmission* (TRANSEC). Dans le contexte du *problème du prisonnier*, Alice et Bob sont deux détenus qui communiquent par l'intermédiaire de Wendy, le gardien. Si Wendy soupçonne qu'ils élaborent un plan pour s'échapper, celle-ci s'autorise à mettre fin à la communication entre les deux détenus. De plus, Wendy peut aussi modifier les messages si elle le désire. L'utilisation de messages chiffrés éveillerait les soupçons ; ils seraient de plus, contraints par les autorités à divulguer leur clé de chiffrement. La seule alternative d'Alice et Bob est donc de s'envoyer des messages innocents et de dissimuler l'information compromettante dans ceux-ci. De fait, ils mettent en place un canal de transmission (par l'intermédiaire des messages eux-mêmes) qui n'est pas visible pour Wendy ; ce canal est appelé *canal subliminal*. La stéganographie permet alors de généraliser les techniques classiques de TRANSEC, telles l'étalement de spectre ou l'évasion de fréquence, à tout type de données. Réciproquement, l'étalement de spectre et l'évasion de fréquence peuvent être vus comme des techniques de stéganographie, dissimulant un signal dans de la bande passante ou le spectre des fréquences. Ces techniques visent par ailleurs à rendre furtives les transmissions mais aussi à se protéger contre un attaquant actif qui brouillerait le canal.

Nous supposons tout d'abord qu'Alice et Bob se sont échangés au préalable une clé *secrète cryptographique* (ou ont accès à un serveur de clés publiques cryptographiques) ainsi qu'une clé *secrète stéganographique* (ou ont accès à un serveur de clés publiques stéganographiques). Nous appelons dans la suite *médium support* ou *support de couverture* le médium qui va contenir le message caché et *stégo médium* tout médium contenant de l'information cachée. Par abus de langage, nous utilisons aussi le terme de support et nous disons qu'un médium est *stégo* si c'est un stégo médium et *non stégo* dans le cas contraire. La mise en œuvre d'un schéma de stéganographie s'effectue alors en deux étapes distinctes. Pour envoyer un message à Bob, Alice effectue les opérations suivantes :

1. elle compresse son message et le chiffre avec la clé cryptographique,
2. elle génère un support de couverture,
3. l'algorithme de stéganographie sélectionne les sous-parties du support favorables à la dissimulation,
4. il dissimule ensuite aléatoirement, à l'aide de la clé stéganographique, le message chiffré dans les parties favorables,
5. Alice envoie le stégo médium par un canal classique.

Cette étape, appelée aussi *dissimulation*, est illustrée par la figure 4.11 page suivante.

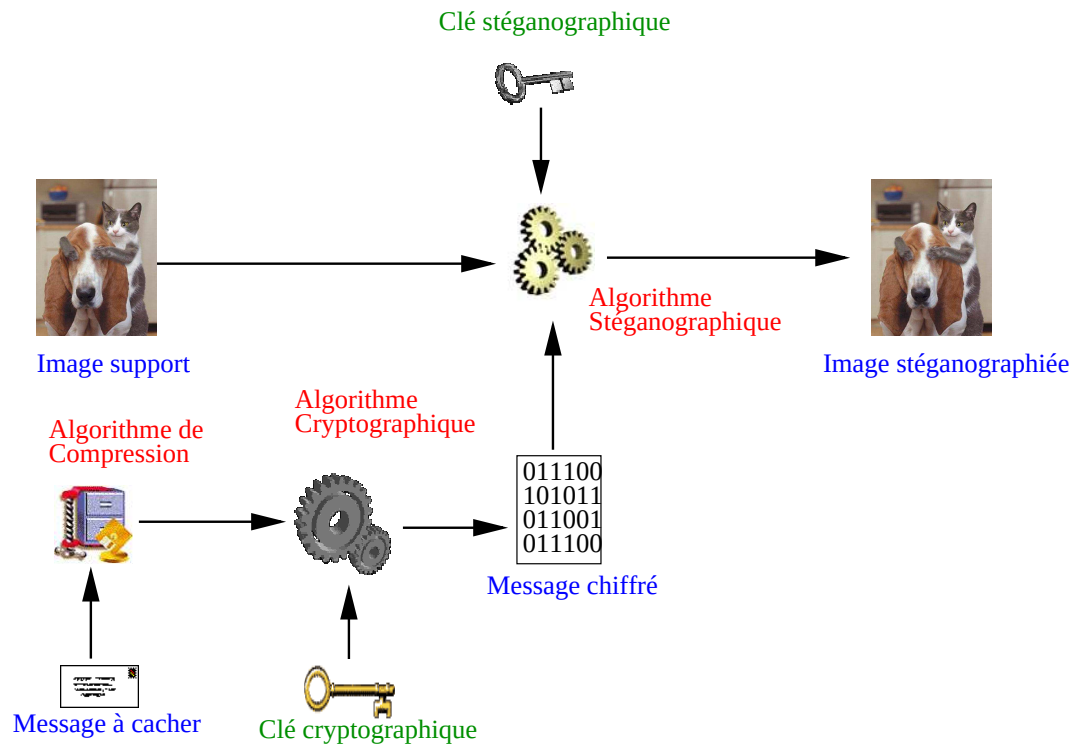


FIG. 4.11 – Étape de dissimulation pour une image fixe

Pour lire le message d'Alice, Bob effectue les opérations suivantes :

1. Bob reçoit le stégo médium par le canal classique,
2. l'algorithme de stéganographie sélectionne les sous-parties du support favorables à la dissimulation,
3. il retrouve la position du message chiffré dans les parties favorables, à l'aide de la clé stéganographique,
4. Bob déchiffre le message à l'aide de la clé cryptographique et le décompresse.

Cette étape, appelée aussi *extraction*, est illustrée par la figure 4.12 page suivante. Comme la cryptographie, la stéganographie peut être abordée sous l'angle de la théorie de l'information. Dans cet esprit, des définitions de schémas de stéganographie ont été proposées par C. Cachin [38, 39, 40], J. Zöllner *et al.* [209] et R. Chandramouli [44, 45, 46].

Quelques règles de base doivent être respectées pour éviter de mettre en défaut le schéma par des attaques triviales. Tout d'abord, c'est l'émetteur qui génère le support. Celui-ci doit n'être utilisé qu'une seule fois et détruit après utilisation, afin d'éviter les attaques par différence. En effet, tout attaquant possédant le support original est capable avec une probabilité égale à 1 de détecter tout stégo médium issu du support. De même, pour éviter les attaques visuelles, l'algorithme de stéganographie ne doit pas détériorer visuellement le support. En général, les valeurs du support que l'algorithme de stéganographie modifie pour dissimuler l'information possèdent une distribution uniforme (par exemple les bits de poids faible (LSB)). Chiffrer permet d'une part d'assurer le COMSEC et d'autre part d'uniformiser la distribution des bits du message effectivement dissimulé. Le but étant d'obtenir une distribution des valeurs du stégo médium identique à celle des valeurs du support. De

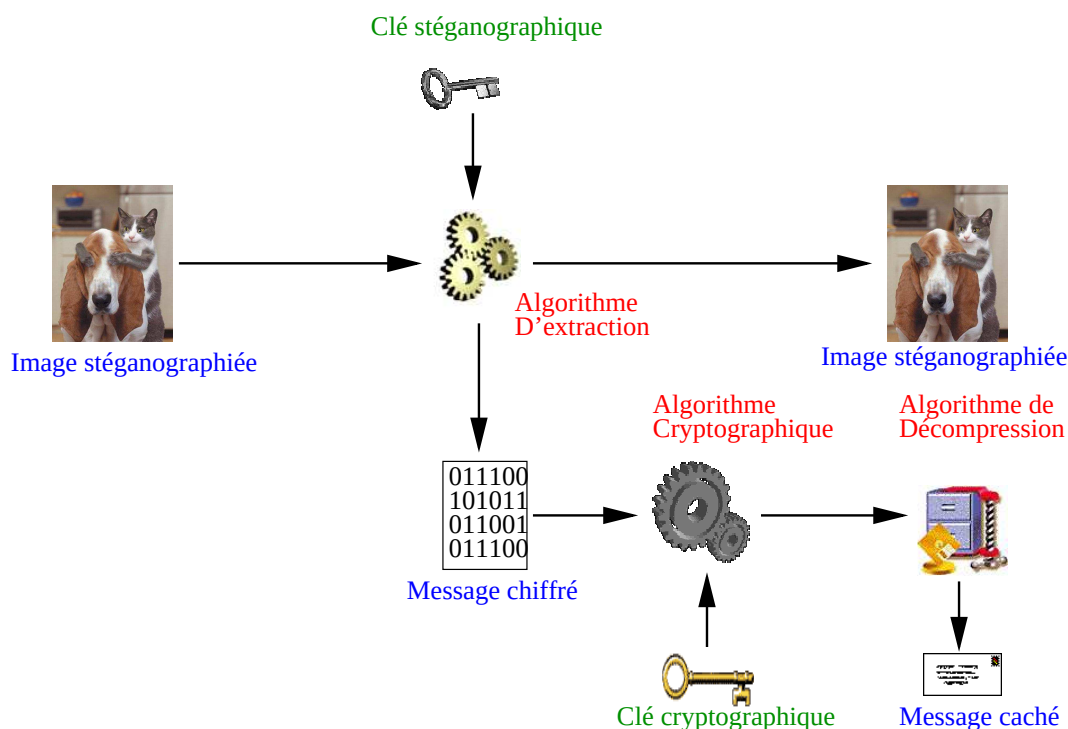


FIG. 4.12 – Étape d'extraction pour une image fixe

plus, afin de se prémunir contre des attaques classiques sur les moments d'ordre supérieur, comme un test du χ^2 par exemple, on demande aux algorithmes de stéganographie de préserver les statistiques des valeurs qu'il modifie, à l'ordre 1 et au-delà. Enfin, la quantité d'information à dissimuler doit être petite. Le support peut être en effet considéré comme un canal au sens de Shannon [44], avec une capacité limitée. Intuitivement, plus on dissimule d'information dans un support, plus celui-ci subit de changements et plus le stégo médium risque d'être détecté. Dans le domaine de la dissimulation d'information, il faut composer avec un compromis entre la *capacité*, c'est-à-dire la quantité d'information dissimulée, l'*indéfectabilité*, c'est-à-dire la probabilité que le stégo médium soit déclaré non stégo par un attaquant et la *robustesse*, c'est-à-dire la quantité d'information dissimulée résiduelle après un certain nombre de transformations sur le stégo médium. Ce compromis est traditionnellement représenté par un triangle comme illustré sur la figure 4.13 page suivante. En stéganographie, le compromis qui nous intéresse est celui entre la capacité et l'indéfectabilité. En effet, on considère que si le message est altéré, il sera ré-émis. L'objectif du stéganographe est bien d'envoyer le maximum d'information sans qu'un attaquant puisse le détecter. La notion de robustesse est plutôt importante pour le tatouage ou le marquage ; ceux-ci ne rentrant pas dans le cadre de notre étude. Le lecteur intéressé par le marquage d'image pourra trouver une bonne introduction dans les ouvrages [73, 110, 114].

Nous prenons maintenant la place de l'analyste. Comme nous venons de le voir, les schémas modernes de stéganographie intègrent deux niveaux de sécurité indépendants et répondant à deux besoins de sécurité différents. Le Saint Graal du stéganalyste est bien évidemment d'avoir accès à l'information en clair échangée par Alice et Bob. Pour ce faire, il doit tout d'abord distinguer les stégo média des autres, puis extraire l'information dissimulée et enfin cryptanalyser le message chiffré. Dans un contexte réaliste, selon

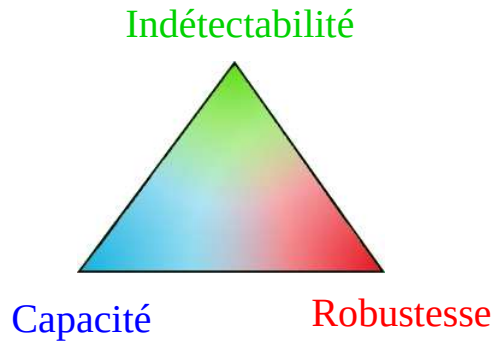


FIG. 4.13 – Compromis entre la capacité, l'indétectabilité et la robustesse

les principes énoncés par A. Kerckhoffs [120], l'analyste ne dispose que des spécifications des schémas de stéganographie utilisés par Alice et Bob. Cryptanalyser le message chiffré est alors équivalent à une attaque à chiffré seul. Extraire l'information est équivalent à reproduire la suite de pseudo-aléa générée à l'aide de la clé stéganographique sans aucune connaissance ni de cette clé, ni même de la suite. Seule une attaque par recherche exhaustive sur la clé semble convenir. Enfin, distinguer les stégo média des autres est équivalent à trouver au moins une mesure statistique sur les média dont la distribution est différente suivant que le médium est stégo ou non. Aux vues des étapes que doit franchir le stéganalyste, il semble que l'avantage soit définitivement acquis au stéganographe. Supposons de plus, que celui-ci dissimule dans un même support de couverture C deux messages m_1 et m_2 avec les clés stéganographiques respectives k_1 et k_2 pour obtenir le stégo médium S . Supposons aussi qu'il existe un distingueur stéganographique idéal ; c'est-à-dire capable de détecter les stégo média avec une probabilité égale à 1. Une analyse de S avec ce distingueur indiquera qu'il contient de l'information dissimulée. Confondu, le stéganographe sera contraint de révéler une clé k_i et donc un message m_i , $i \in \{1, 2\}$. Or, l'extraction de m_i consiste en une lecture de S ; S étant inchangé après l'extraction, le distingueur classifera toujours S comme stégo médium, qu'il contienne plus d'information dissimulée ou non. En d'autres termes, quel que soit le distingueur stéganographique, celui-ci ne peut pas distinguer un stégo médium contenant exactement un message caché, d'un autre possédant plus d'un message dissimulé. Nous avons traduit cette propriété, *plausible deniability* en anglais, par *indistingabilité indéniable*. Aussi surprenant que cela puisse paraître, peu d'implémentations stéganographiques offrent cette fonctionnalité. Parmi les trois schémas de stéganographie que nous étudions, *Outguess* [155], *F5* [203] et *JPHide and JPSeek* [124], seul *Outguess* propose de dissimuler deux messages en même temps, dans le même support et avec deux clés stéganographiques différentes. À la vue de cet exemple, il apparaît naturellement une règle d'or du bon usage de la stéganographie : il faut dissimuler un message sans importance en plus du message à envoyer, ce avec une clé stéganographique différente.

Dans ce contexte extrêmement défavorable au stéganalyste, nous avons choisi de nous focaliser sur la première étape de l'analyse, c'est-à-dire la conception de distingueurs stéganographiques. Nous nous sommes attachés principalement à étudier des techniques de stéganographie dédiées aux images fixes et plus particulièrement aux formats non compressés et au format JPEG. Bien que tout vecteur d'information soit potentiellement

support d'une technique stéganographique, les images sont le vecteur le plus usité par les algorithmes de stéganographie, comme le montre le recensement effectué en annexe D. De plus, le nombre d'images numériques disponibles sur Internet favorise l'effet de masse et rend la recherche de stégo média encore plus difficile comme l'illustre la tentative infructueuse de N. Provos. En 2001, celui-ci analyse quelques trois millions d'images issues d'eBay et de Usenet [157] avec son logiciel *stegdetect*, implémentant toutes les stéganalyses à l'état de l'art, sans y trouver la moindre image stéganographiée. Nous décrivons dans un premier temps, au chapitre 5, les formats non compressé et JPEG et détaillons précisément les algorithmes de stéganographie pour lesquels nous avons spécifié des distingueurs stéganographiques. Au chapitre 6, nous nous intéressons aux modèles classiques de sécurité en stéganographie et proposons deux nouveaux modèles d'attaquant très faible afin de modéliser l'attaquant passif réel. Nous les relierons aux modèles classiques, d'une part, et aux performances des stéganalyses pratiques évaluées expérimentalement, d'autre part. Ce sont ces modèles d'attaquant que nous mettons en œuvre par la suite. Nous introduisons ensuite succinctement la théorie de la discrimination sous l'angle de l'analyse discriminante de Fisher. Ces deux chapitres introductifs sont essentiels car ils présentent les objets que nous analysons, le modèle dans lequel s'inscrit cette analyse, les règles pour évaluer les performances des distingueurs dans ce modèle et enfin les notions de statistiques au centre de l'élaboration de nos distingueurs stéganographiques. Au chapitre 7, nous montrons qu'il est illusoire de croire qu'il suffit de ne pas utiliser des parties d'une image, favorables à une technique de stéganalyse pour s'en protéger. Nous illustrons notre point de vue en proposant une technique efficace de stéganalyse de l'algorithme *Multi Bit Plane Image Steganography* (MBPIS), proposé par B.C. Nguyen *et al.* [147] à IWDW en 2006. Les auteurs annoncent que leur technique est robuste à la stéganalyse RS [76, 77] en excluant de la dissimulation des zones de l'image qui sont plutôt favorables à cette stéganalyse. Une simple adaptation de celle-ci, restreinte aux zones sélectionnées par MBPIS, nous permet de détecter efficacement des stégo média générés avec cet algorithme. Enfin, au chapitre 8, nous présentons une approche nouvelle pour la stéganalyse JPEG. Nous proposons d'étudier des statistiques des coefficients DCT quantifiés après codage entropique, dans ce que nous appelons le *domaine fréquentiel compressé* (DFC). Cette approche met en évidence une nouvelle classe de fonctions qui permettent d'obtenir, en pratique, des taux de détection quasi-indépendants de la quantité d'information dissimulée. Nous illustrons cette technique par la conception de distingueurs stéganographiques universels, d'une part, et spécifiques, d'autre part, pour les algorithmes *Outguess* [156], *F5* [203] et *JPHide and JPSeek* [124]. Enfin, nous validons expérimentalement l'indépendance des taux de détection et de la quantité d'information dissimulée.

Chapitre 5

Stéganographie adaptée aux images non compressées et au JPEG

« Si tu révèles ton secret au vent, tu ne dois pas lui reprocher de le révéler à l'arbre. »

Khalil Gibran (Le sable et l'écume)

Sommaire

5.1	Stéganographie dans le domaine spatial	144
5.1.1	Les images non compressées	144
5.1.2	Stéganographie dans des plans de bits multiples	145
5.2	Le format JPEG	151
5.2.1	Changement de l'espace des couleurs	151
5.2.2	Transformation DCT (Discrete Cosinus Transform)	152
5.2.3	Quantification	154
5.2.4	Codage RLE	156
5.2.5	Codage de Huffman	157
5.3	Stéganographie adaptée au format JPEG	159
5.3.1	Outguess	159
5.3.2	F5	161
5.3.3	JPHide and JPSeek	168

DANS ce chapitre, nous nous intéressons à la stéganographie dédiée aux images fixes. Ce support est l'un des vecteurs d'information les plus usités et les plus présents sur Internet. Le stéganographe souhaitant dissimuler l'existence même de l'information qu'il veut transmettre va naturellement se tourner vers les supports les plus représentés afin de « noyer » son message caché dans la masse. Les images fixes sont donc les média de couverture les plus prisés par les logiciels de stéganographie. De nombreux formats d'image sont disponibles sur Internet, mais là encore, deux d'entre eux semblent être majoritaires. En premier lieu, les formats non compressés permettent d'échanger des images sans dégradation. Les fichiers associés sont néanmoins de taille beaucoup plus importante mais de nombreuses applications nécessitent des images de qualité supérieure (images

médicales, photographie de qualité, photos satellites . . .). Parmi les formats d'images compressées, le JPEG [200] est bien sûr le plus répandu et s'est imposé comme standard *de facto*. Le format JPEG est en effet un bon compromis entre la qualité et la taille du fichier. Il est de plus supporté par la majorité des applications et laisse facilement le choix à l'utilisateur de la qualité qu'il souhaite obtenir.

La plupart des algorithmes de stéganographie sont adaptés pour les formats non compressés et le format JPEG. Dans un premier temps, nous décrivons le domaine spatial, *i.e.* l'image non compressée ainsi que l'algorithme de stéganographie pour lequel nous présentons une stéganalyse au chapitre 7. Nous détaillons ensuite finement le format de compression JPEG afin de comprendre les mécanismes sous-jacents et les contraintes imposées pour utiliser les fichiers JPEG comme média de couverture. Enfin, nous rappelons les algorithmes de stéganographie adaptés au format JPEG que nous stéganalysons au chapitre 8. Les descriptions du format JPEG et des algorithmes sont centrales dans notre étude car elles permettent non seulement de se familiariser avec les algorithmes attaqués mais aussi d'appréhender la méthodologie que nous avons adoptée pour notre stéganalyse ; celle-ci étant au premier abord contre-intuitive.

5.1 Stéganographie dans le domaine spatial

5.1.1 Les images non compressées

Les images fixes non compressées apparaissent dans de nombreux formats, notamment BMP, Raw, X Pixmap Chaque format correspond à une structure particulière de représentation et de stockage des informations relatives à l'image (données, taille, nombre de bits par donnée . . .). L'image non compressée est composée d'une succession de points appelés *pixels*. Elle est alors en noir et blanc, en niveaux de gris ou en couleurs selon le nombre de bits nécessaires à coder chaque pixel.

Pour une image en noir et blanc, chaque pixel est codé sur 1 bit valant 0 pour un pixel noir et 1 pour un pixel blanc. Pour une image en niveau de gris un octet permet de coder les 256 niveaux de gris que peut prendre le pixel. Les pixels des images couleurs sont codés sur au moins 24 bits sous forme de coordonnées dans un *espace de couleurs*. Il existe plusieurs façons de coder la couleur et donc plusieurs espaces de couleurs. Le choix d'un espace de couleurs dépend essentiellement des applications mises en œuvre, comme par exemple la photographie, la télévision, l'impression, etc. Le lecteur intéressé par une description exhaustive des différents modèles de représentation de la couleur pourra se référer à [34, ch. 6]. Un des espaces de couleur le plus usité pour les images fixes est l'espace RVB. Dans cet espace, chaque couleur possède trois composantes qui correspondent respectivement à une intensité de rouge (R), de vert (V) et de bleu (B). Chaque composante est additionnée pour donner la couleur finale. L'espace RVB est un espace en 3 dimensions et peut se représenter par un cube RVB, comme l'illustre la figure 5.2 page ci-contre.

Il est néanmoins possible de stocker des images couleurs avec des pixels codés sur moins de 24 bits. Par exemple, pour stocker des images couleur sur 16 bits nous devons constituer un dictionnaire de 2^{16} entrées, appelé *palette*. À chaque pixel de 16 bits est associé un entier de 16 bits qui fait référence à une entrée de la palette et à chaque entrée est associé un triplet (r, v, b) de 24 bits. La palette doit évidemment être codée et stockée dans le format



FIG. 5.1 – Même image en noir et blanc, niveaux de gris et en couleurs



FIG. 5.2 – Cube RVB

de l'image. Les fichiers GIF, par exemple, utilisent une palette de 256 entrées. L'image de la figure 5.3 possède une palette de 16 couleurs.

5.1.2 Stéganographie dans des plans de bits multiples

La stéganographie LSB (Least Significant Bit) consiste à dissimuler l'information dans des bits de poids faibles d'un support. Cette technique est un cas particulier de stéganographie $\pm k$, qui incrémente ou décrémente les valeurs du support de $\pm k$. Historiquement, peut-être par facilité d'implémentation, la stéganographie LSB adaptée aux images fixes non compressées est l'une des premières techniques stéganographiques et peut-être même l'une des plus employées encore aujourd'hui.

Malheureusement, la stéganographie LSB est sujette à de nombreuses attaques, tout comme la stéganographie $\pm k$. Parmi les innombrables stéganalyses, nous pouvons notam-



FIG. 5.3 – Image couleur avec une palette de 16 couleurs

Dans le cas d'une image non compressée codée sur 24 bits, chaque pixel est décrit par trois octets.

```
(00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)
⋮
```

Pour dissimuler la chaîne de bits **10000011**, on utilise le bit de poids faible de chaque octet.

```
(00100111 11101000 11001000)
(00100110 11001000 11101000)
(11001000 00100111 11101001)
⋮
```

FIG. 5.4 – Stéganographie LSB pour une image non compressée

ment citer la stéganalyse RS due à J. Fridrich [77], qui permet non seulement de détecter l'usage de stéganographie LSB mais aussi d'estimer la longueur du message, les attaques du type χ^2 dues à A. Westfeld et A. Pfitzmann [204], l'analyse par paires de S. Dumitrescu *et al.* [59] et améliorée par P. Lu *et al.* [134] ou d'autres toutes aussi efficaces proposées par S. Lyu et H. Farid [137] ou A.D. Ker [118]. La stéganalyse RS est notamment détaillée au chapitre 7.

Pour contourner ces attaques, B.C. Nguyen *et al.* ont proposé à IWDW'06 (International Workshop on Digital Watermarking) [147] de ne pas se limiter aux bits de poids faible mais d'utiliser des plans de bits d'ordre supérieur (les auteurs préconisent 4 au maximum), mais en restreignant la dissimulation à des zones non homogènes afin de ne pas détériorer

le support de façon visible. Une caractéristique particulière de cette technique est de dissimuler l'information non pas dans les plans de bits usuels mais dans des plans de bits après un *Codage de Gray Canonique* (CGC). L'algorithme proposé est ainsi robuste à la stéganalyse RS et par analyse de paires. Ce paragraphe décrit la méthode de B.C. Nguyen afin de comprendre les mécanismes sous-jacents ainsi que la stéganalyse décrite au chapitre 7.

Avant de décrire plus en détails, nous rappelons la définition du Codage de Gray Canonique (CGC). Soit à coder un pixel $b_N b_{N-1} \dots b_1$ de N bits dans les plans de bits usuels, en CGC par une valeur $g_N g_{N-1} \dots g_1$ de N bits définie par

$$\begin{cases} g_N = b_N, \\ g_i = b_i \oplus b_{i+1}, 1 \leq i \leq N-1. \end{cases} \quad (5.1)$$

Le résultat de la transformation d'une image dans le CGC est représenté par la figure 5.5.

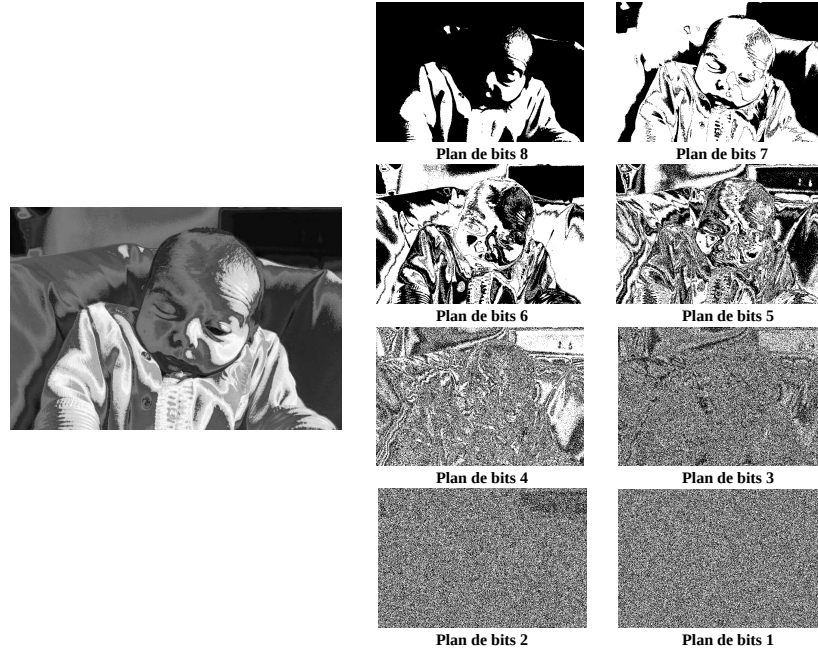


FIG. 5.5 – Décomposition en plans de bits d'une image codée en CGC

La transformation inverse est alors définie par

$$\begin{cases} b_N = g_N, \\ b_i = g_i \oplus g_{i+1}, 1 \leq i \leq N-1. \end{cases} \quad (5.2)$$

Il en résulte que tout changement de la valeur d'un bit g_i d'un pixel dans le CGC implique une variation de la valeur du pixel comprise dans $[1, 2^i - 1]$. L'image est tout d'abord décomposée en N plans de bits $B_N B_{N-1} \dots B_1$ dans le CGC. Afin d'éviter les attaques visuelles, le nombre maximum i_{max} de plans de bits utilisés lors de la dissimulation est limité à 4. Les messages sont alors insérés dans le support du plan de bits le plus élevé vers le plan de bits le plus faible, *i.e.* de $B_{i_{max}}$ à B_1 .

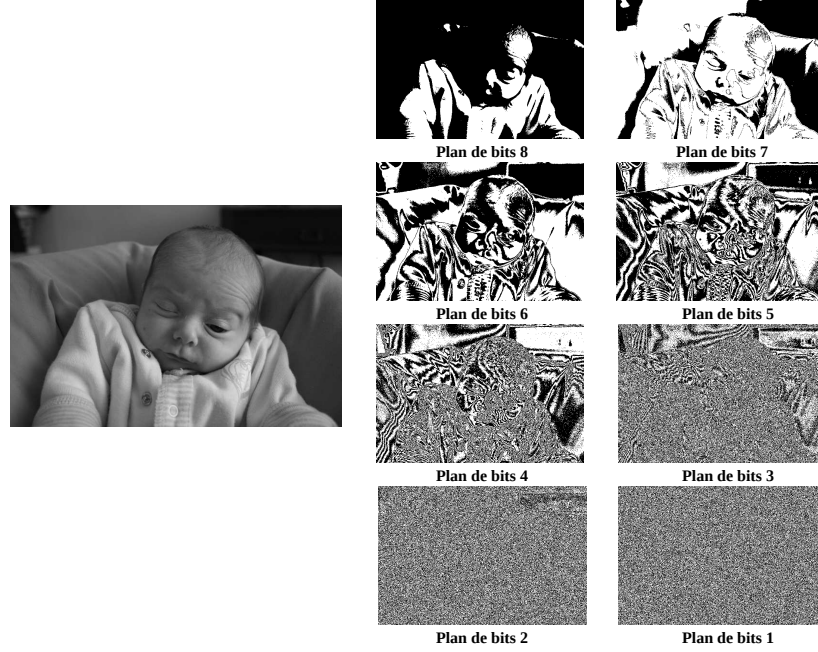


FIG. 5.6 – Décomposition en plans de bits d'une image codée en plans de bits usuels

Les zones homogènes pour un certain plan de bits B_i , sont obtenues en combinant des zones homogènes plus petites entre elles. Chaque plan de bits est divisé en fenêtres W non recouvrantes et de taille $n \times n$. Posons par exemple $n = 2$ et considérons une telle fenêtre.

$$W = \begin{bmatrix} p_1 & p_2 \\ p_3 & p_4 \end{bmatrix}, \quad (5.3)$$

où p_i représente la valeur d'un pixel. Nous calculons alors

$$\begin{bmatrix} |p_1 - p_1| & |p_1 - p_2| \\ |p_1 - p_3| & |p_1 - p_4| \end{bmatrix} = \begin{bmatrix} p'_1 & p'_2 \\ p'_3 & p'_4 \end{bmatrix}. \quad (5.4)$$

Finalement, si $\forall j, \lfloor p'_j/2^i \rfloor \leq t$, où t est le seuil, alors W est dite *homogène* et *non homogène* dans le cas contraire. Cette définition peut être étendue à des fenêtres de taille quelconque $m \times n$. Le message est alors inséré dans les zones non homogènes du plan de bits B_i à l'aide d'une séquence pseudo-aléatoire produite par un Générateur Pseudo-Aléatoire (GPA), par insertion, en fixant la valeur des bits de B_i à celles des bits du message. Les algorithmes d'insertion et d'extraction sont résumés dans les figures 5.7 page ci-contre et 5.8 page 150 respectivement.

La notion de *capacité d'un algorithme stéganographique* est très proche de celle donnée par Shannon [171]. En effet, si l'on considère le support comme un canal de transmission, la quantité d'information que l'on peut dissimuler est donc bornée. Dans cet esprit, un algorithme de stéganographie peut être vu comme un algorithme de codage et la capacité du support est donc soumise à la borne de Shannon. Une approche de la stéganographie sous l'angle de la théorie de Shannon a été proposée par C. Cachin [38, 39, 40] et dans ce modèle, R. Chandramouli et N.D. Nemon [44, 47] ont défini la notion de capacité. En pratique, un même support possède une capacité différente selon l'algorithme de stéganographie

Algorithme d'insertion MBPIS

Entrées : un message secret M de l bits,
 I une image non compressée,
 K_e, K_s les clés cryptographique et stéganographique.

Sortie : une image stéganographiée ou *échec*.

Paramètres : i_{max} le plan de bits maximum,
 t , le seuil,
 $m \times n$, la taille de la fenêtre.

```

1  Coder  $I$  en  $I'$  dans le CGC selon (5.1)
2  Décomposer  $I'$  en  $N$  plans de bits
3  Compresser et Chiffrer  $M$  en  $M'$  avec  $K_e$ 
4  Initialiser le GPA à l'aide de  $K_s$ 
5  Pour  $i$  de  $i_{max}$  à 1
6    Déterminer les zones non homogènes de  $B_i$  taille  $m \times n$ 
      avec le seuil  $t$  selon (5.4)
7    Insérer les bits de  $M'$  dans les bits du plan  $B_i$  des zones
      non homogènes à l'aide du GPA
8  Fin Pour
9  Si il reste des bits de  $M'$  non insérés alors
10   Renvoyer échec
11 Fin Si
12 Coder  $I'$  dans le plan de bits usuel selon (5.2)
13 Renvoyer  $I'$ 

```

FIG. 5.7 – Algorithme d'insertion MBPIS

utilisé. Afin de pouvoir comparer la quantité d'information que peuvent dissimuler les algorithmes de stéganographie, nous utilisons le *taux stéganographique*, c'est-à-dire rapport entre la taille du message à dissimuler et la taille du support. Ce taux peut s'exprimer en pourcentage ou en nombre de bits de message dissimulés par pixel (bpp). Par la suite, nous avons fait le choix de l'exprimer en pourcentage. La figure 5.9 page suivante met en évidence les pixels affectés par l'insertion d'un message par MBPIS. Les images supports (ligne du haut) sont comparées avec les stégo images (ligne du milieu) pour différents taux stéganographiques. Les images de la dernière ligne, correspondent à la différence entre le support et le stégo médium. Les pixels blancs représentent les pixels inchangés, ceux de couleur apparaissent pour des pixels changés par MBPIS selon la composante de même couleur.

Algorithme d'extraction MBPIS

Entrées : une stégo image I' ,
 K_e, K_s les clés cryptographique et stéganographique.

Sortie : le message M dissimulé.

Paramètres : i_{max} le plan de bits maximum,
 t , le seuil,
 $m \times n$, la taille de la fenêtre.

- 1 **Coder** I' dans le CGC selon (5.1)
- 2 **Décomposer** I' en N plans de bits
- 3 **Initialiser** le GPA à l'aide de K_s
- 4 **Pour** i de i_{max} à 1
- 5 **Déterminer** les zones non homogènes de B_i taille $m \times n$
 avec le seuil t selon (5.4)
- 6 **Lire** les bits de M' dans les bits du plan B_i des zones
 non homogènes à l'aide du GPA
- 7 **Fin Pour**
- 8 **Déchiffrer** M' avec k_e puis le décompresser en M
- 9 **Renvoyer** M

FIG. 5.8 – Algorithme d'extraction MBPIS

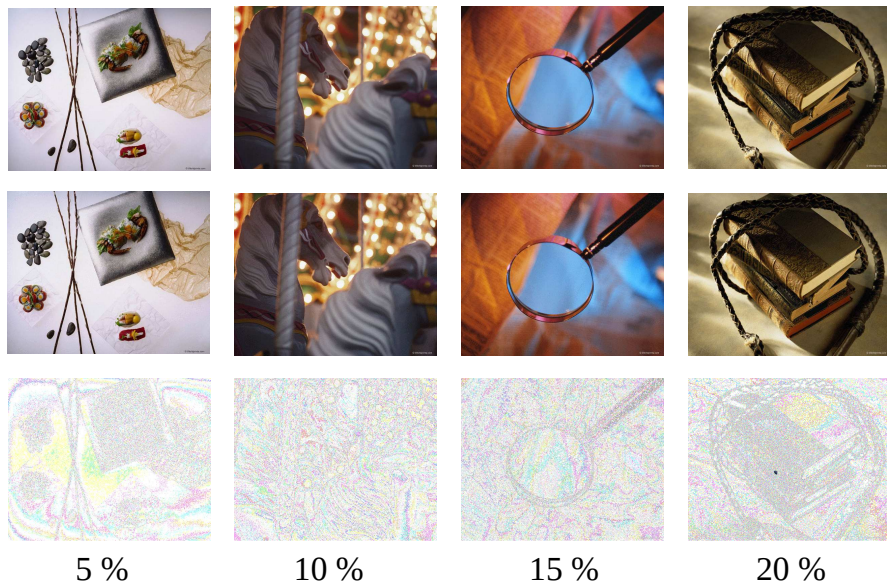


FIG. 5.9 – Images stéganographiées par MBPIS pour différents taux

5.2 Le format JPEG

JPEG est l'acronyme de « Joint Picture Experts Group ». Ce groupe d'experts s'est formé en 1986 sous l'impulsion de l'ISO (International Standards Organisation) et de l'ITU (International Telecommunication Union) afin de travailler à l'élaboration d'un standard de compression pour les images fixes en nuances de gris ou en couleurs. Ses travaux ont débouché sur deux standards de compression : la norme T.81 pour l'ITU et la norme 10918-1 pour l'ISO. Par abus de langage, JPEG désigne aujourd'hui ces standards internationaux de compression.

Ce chapitre présente succinctement les grandes lignes des étapes qui composent la compression JPEG. Il a surtout vocation à être pédagogique et son objectif est de permettre la compréhension des techniques de stéganographie et de stéganalyse adaptées au format JPEG. Dans cet esprit, certaines approximations ont été effectuées et certains détails techniques ont été omis pour ne pas noyer le lecteur. Les puristes nous en excuseront et pourront retrouver l'ensemble des informations et précisions concernant le format JPEG, notamment dans les normes sus-citées mais aussi dans [200, 34]. Le lecteur pourra aussi se référer à une description très complète et en français du format JPEG [142]. Ce paragraphe s'organise suivant les cinq grandes étapes de la compression JPEG qui sont : le changement de l'espace des couleurs, la transformation en cosinus discrète (DCT), la quantification, le codage *Run Length Encoding* (RLE) et la compression de Huffman.

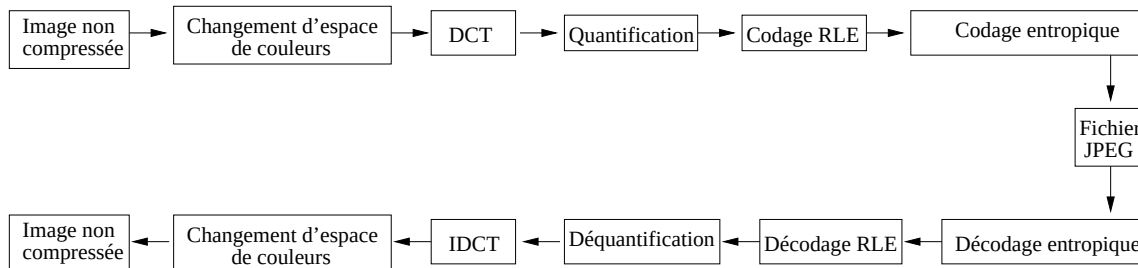


FIG. 5.10 – Schéma de compression/décompression JPEG

5.2.1 Changement de l'espace des couleurs

Soit une image I que l'on veut compresser au format JPEG. Chaque pixel p_i , est représenté par un triplet (R_i, V_i, B_i) dans l'espace RVB. La première étape de la compression JPEG consiste en un changement de l'espace des couleurs de RVB vers l'espace de couleurs YCbCr.

Un pixel sera donc codé par un triplet (Y_i, Cb_i, Cr_i) , où Y_i désigne la *luminance*, c'est-à-dire l'intensité lumineuse, Cb_i la *chrominance bleue*, c'est-à-dire l'intensité de la couleur bleue et Cr_i la *chrominance rouge*, c'est-à-dire l'intensité de la couleur rouge. Le changement d'espace s'effectuent en appliquant les équations suivantes.

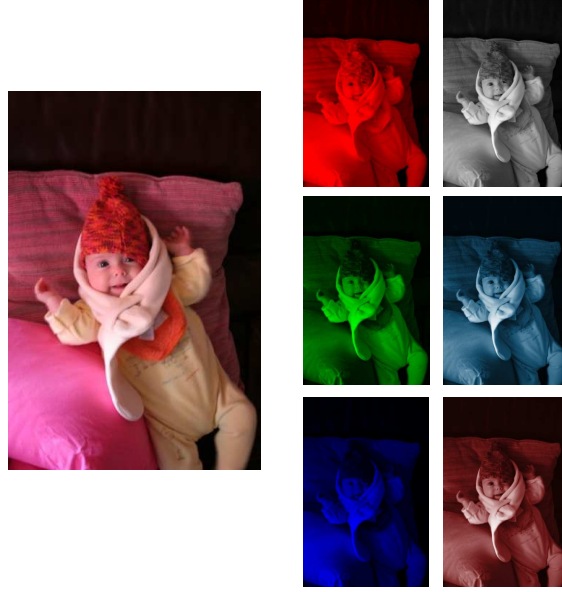


FIG. 5.11 – Décomposition suivant les composantes RVB et YCbCr

Les équations de changement de RVB vers YCbCr :

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0,299 & 0,587 & 0,114 \\ -0,1687 & -0,3313 & 0,5 \\ 0,5 & -0,4187 & -0,0813 \end{pmatrix} \times \begin{pmatrix} R \\ V \\ B \end{pmatrix} + \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix}.$$

Les équations de changement de YCbCr vers RVB :

$$\begin{aligned} R &= Y + 1,402(Cr - 128), \\ V &= Y - 0,34414(Cb - 128) - 0,71414(Cr - 128), \\ B &= Y + 1,772(Cb - 128). \end{aligned}$$

Le changement d'espace de couleurs se justifie par le fait que l'espace YCbCr est très proche du fonctionnement de l'œil humain. De plus, ce dernier est très sensible aux variations de luminance et très peu sensible aux variations de chrominance. De ce fait, on pourra effectuer des modifications sur les composantes Cb et Cr afin de compresser l'information visuelle et cela, sans que l'œil ne détecte la différence. Pour ce faire, les pixels sont regroupés en blocs de 4×4 pixels. Les quatre valeurs de chrominance bleue sont remplacées par leur moyenne; la même transformation est effectuée sur les quatre valeurs de chrominance rouge. L'œil ne perçoit pas les changements effectués. Le gain de stockage est de 50% et la transformation appliquée est non réversible, la compression est dite *avec perte*. Cette transformation est appelée *sous-échantillonnage* et illustrée par la figure 5.12 page suivante.

Chacune des valeurs Y , Cb , Cr est un nombre codé sur P bits, compris entre 0 et $2^P - 1$, où P est appelé *précision*. Les valeurs Y , Cb et Cr sont alors ramenées sur l'intervalle $[-2^{P-1} + 1; 2^{P-1} - 1]$, par une translation de (-2^{P-1}) .

5.2.2 Transformation DCT (Discrete Cosinus Transform)

De même, l'œil est sensible aux basses fréquences et peu sensible aux hautes fréquences. Le JPEG tire avantage de cette sensibilité en passant tout d'abord dans le domaine

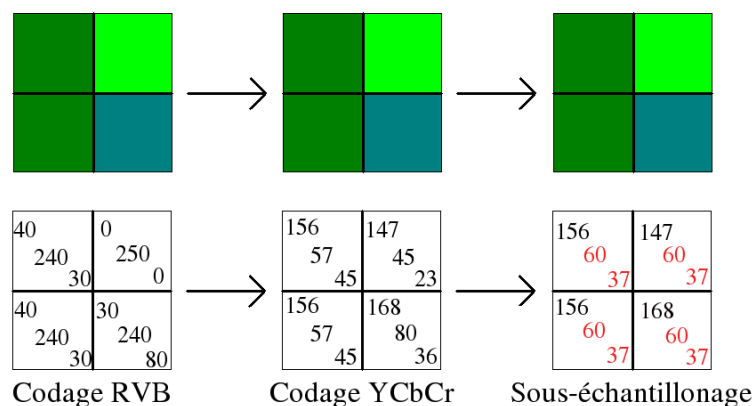
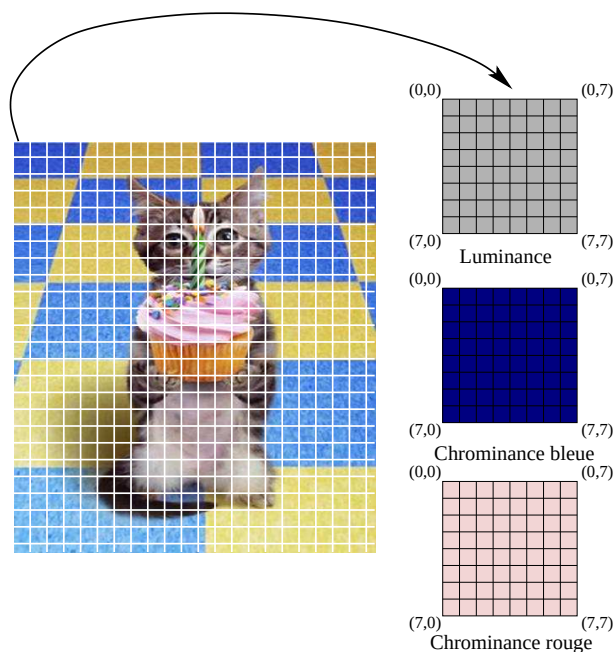


FIG. 5.12 – Étape de sous-échantillonnage

fréquentiel en appliquant une transformée de Fourier discrète. Des approximations sont effectuées sur les hautes fréquences ; elles ne sont donc pas perçues par l'œil humain. Pour ce faire, chaque composante Y, Cb et Cr est découpée en blocs de 8×8 valeurs indexée de 0 à 7 de haut en bas et de gauche à droite comme illustré par la figure 5.13.

FIG. 5.13 – Découpage en blocs de 8×8 valeurs

Chacun de ces blocs appartenant au domaine spatial est ensuite transformé en un bloc de 8×8 valeurs dans le domaine fréquentiel par la transformation DCT. Un coefficient DCT S_{uv} de coordonnées (u, v) dans le domaine fréquentiel s'exprime en fonction des 64 valeurs

du bloc dans le domaine spatial s_{xy} de coordonnées (x, y) à partir de la formule

$$S_{uv} = \frac{1}{4} C(u) C(v) \sum_{x=0}^7 \sum_{y=0}^7 s_{xy} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16},$$

avec

$$C(0) = \frac{1}{\sqrt{2}} \text{ et } C(u) = 1 \text{ si } u \neq 0.$$

Cette transformation est en fait la partie réelle de la transformée de Fourier discrète. La transformée en cosinus discrète inverse, (IDCT) permet de retrouver, lors de la décompression, les blocs dans le domaine spatial à partir des coefficients DCT, à l'aide de la formule

$$s_{xy} = \frac{1}{4} \sum_{u=0}^7 \sum_{v=0}^7 C(u) C(v) S_{uv} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}.$$

Parmi les coefficients DCT, on distingue le coefficient $S_{0,0}$, aussi appelé *coefficient DC*, des autres que l'on nomme *coefficients AC*. Le coefficient DC est le coefficient des basses fréquences, c'est donc lui qui est porteur de la majorité de l'information. Ce coefficient étant généralement le plus grand ; on ne stocke que sa différence avec le coefficient DC du bloc précédent. La figure 5.14 illustre un bloc de coefficients DCT.

235,6	-1	-12,1	-5,2	2,1	-1,7	-2,7	1,3
-22,6	-17,5	-6,2	-3,2	-2,9	-0,1	0,4	-1,2
-10,9	-9,3	-1,6	1,5	0,2	-0,9	-0,6	-0,1
-7,1	-1,9	0,2	1,5	0,9	-0,1	0,6	1,3
0,6	-0,8	1,5	1,6	-0,1	-0,7	0,6	1,3
1,8	-0,2	1,6	-0,3	-0,8	1,5	1	1
-1,3	-0,4	-0,3	-1,5	-0,5	1,7	1,1	-0,8
-2,6	1,6	-3,8	-1,8	1,9	1,2	-0,6	-0,4

FIG. 5.14 – Exemple d'un bloc de coefficients DCT de la luminance

Le coin en haut à gauche des blocs DCT contient les valeurs de basses fréquences, tandis que les hautes fréquences se situent dans le coin bas à droite. Dans un bloc de coefficients DCT, les droites d'équation $u + v = cte$ regroupent les coefficients pour une fréquence donnée.

5.2.3 Quantification

La quantification est une étape importante de la compression JPEG. C'est lors de cette étape que l'information est la plus dégradée. Chaque bloc de coefficients DCT est divisé par une *table de quantification*. Cette division s'effectue coefficient à coefficient et le résultat est arrondi par défaut. Les tables de quantification sont construites à partir d'un *facteur de qualité* Q et de tables de référence représentées dans le tableau 5.1 page suivante.

Les tables de quantification tab_quant sont calculées à partir des tables de référence tab_ref (une pour la luminance et une pour les chrominances) à partir de la formule

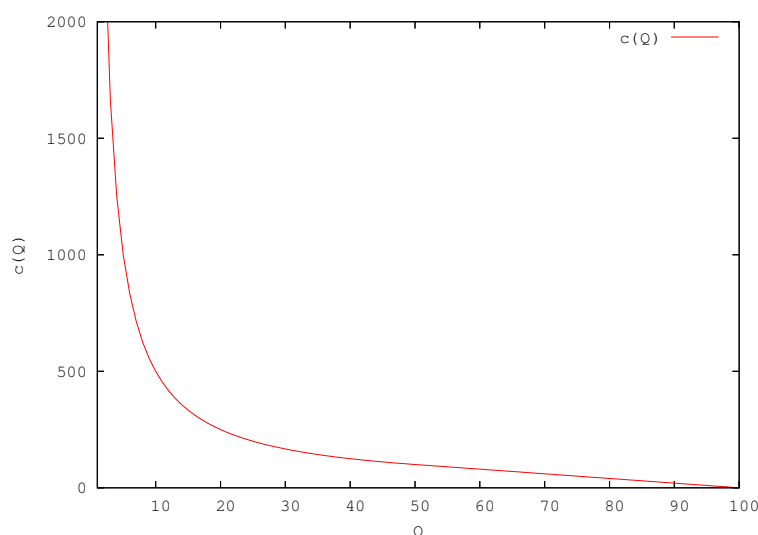
$$tab\_quant(i, j) = \lceil (tab\_ref(i, j) \times c(Q) + 50) / 100 \rceil \quad \forall i, j = 0 \dots 7,$$

16	11	10	16	24	40	51	61	17	18	24	47	99	99	99	99
12	12	14	19	26	58	60	55	18	21	26	66	99	99	99	99
14	13	16	24	40	57	69	56	24	26	56	99	99	99	99	99
14	17	22	29	51	87	80	62	47	66	99	99	99	99	99	99
18	22	37	56	68	109	103	77	99	99	99	99	99	99	99	99
24	35	55	64	81	104	113	92	99	99	99	99	99	99	99	99
49	64	78	87	103	121	120	101	99	99	99	99	99	99	99	99
72	92	95	98	112	100	103	99	99	99	99	99	99	99	99	99

TAB. 5.1 – Tables de référence pour la luminance (à gauche) et la chrominance (à droite)

où $c(Q)$ se déduit à partir du facteur de qualité de la manière suivante.

$$c(Q) = \begin{cases} \frac{5000}{Q} & \text{si } Q < 50, \\ 200 - 2Q & \text{sinon.} \end{cases} \quad (5.5)$$

FIG. 5.15 – $c(Q)$ en fonction du facteur de qualité

Le facteur de qualité prend des valeurs comprises entre 1 et 100, la valeur 1 correspondant à une faible qualité, c'est-à-dire de dégradation maximale et la valeur 100 à la qualité la plus forte, c'est-à-dire sans dégradation. En effet, si $Q = 100$, alors les coefficients des tables de quantification sont tous égaux à 1 d'après l'équation (5.5) ; les coefficients DCT sont juste arrondis lors de l'étape de quantification.

La quantification a pour effet de favoriser les basses fréquences, c'est-à-dire celles qui contiennent le plus d'information. En effet, les coefficients les plus faibles se trouvent en haut à gauche des tables et les coefficients les plus grands en bas à droite. De la même manière, les coefficients DCT de la luminance sont favorisés par rapport à ceux des composantes de chrominance.

En conclusion, la quantification introduit majoritairement des coefficients DCT quantifiés à 0 dans les hautes fréquences et les composantes de chrominance. La figure 5.2 nous

Valeur	Conversion	Complément à 2	Bits conservés
-7	$(-7-1)=-8$	1111000	000
-6	$(-6-1)=-7$	1111001	001
-5	$(-5-1)=-6$	1111010	010
-4	$(-4-1)=-5$	1111011	011
4		0000100	100
5		0000101	101
6		0000110	110
7		0000111	111

TAB. 5.3 – Exemple de valeurs codées sur 3 bits

Valeurs	14	0	4	-8
Codage décimal	0 :4 :14	-	1 :3 :4	0 :4 :7
Codage binaire	0000 :0100 :1110	-	0001 :0011 :100	0000 :0100 :0111

TAB. 5.4 – Exemple de codage de 4 coefficients DCT quantifiés

obtenons une suite binaire de longueur variable comme représentée dans le tableau 5.4.

L'étude des distributions des triplets (RL,S,V) montre que les valeurs V semblent distribuées aléatoirement, tandis que certaines paires RL et S apparaissent plus fréquemment que d'autres. Le format JPEG prévoit alors un codage entropique pour compresser sans perte les paires RL et S. Suivant les modes du format JPEG, ce codage entropique peut être un codage de Huffman ou un codage arithmétique. Le codage de Huffman étant majoritairement utilisé, nous ne détaillerons que celui-ci. Le lecteur intéressé par le codage arithmétique pourra se référer à [105].

Les coefficients DC sont les termes de plus basse fréquence ; généralement non nuls, ils sont codés différemment des coefficients AC. De grande amplitude, seule la différence entre deux coefficients DC successifs est codée. Cette différence est ensuite codée par une paire (S,V) avec S de 0 à 11, codé sur 4 bits représentant le nombre de bits pour coder V et V la valeur de la différence codée sur S bits. Seule la valeur S sera ensuite codée par l'algorithme de Huffman.

5.2.5 Codage de Huffman

Le codage de Huffman [102] est un *codage à longueur variable* (VLC), c'est-à-dire dont les mots du code sont de longueur variable. De tels codes sont aussi appelés *codage entropique*. De plus, celui-ci est dit *préfixé*, c'est-à-dire que chacun des mots du code ne peut être le début d'un autre mot du même code. Dans ce paragraphe, nous décrivons l'algorithme de construction du code de Huffman dans la figure 5.17 page suivante illustré par un exemple adapté de [105]. Pour plus de détails, le lecteur pourra se référer à [105].

La figure 5.18 page suivante montre le déroulement de l'algorithme de Huffman pour la séquence « abracadabra » sur l'alphabet $\mathcal{A} = \{ a, b, c, d, r \}$. Le code de Huffman pour la séquence $\mathcal{S}$ est représenté dans le tableau 5.2.5 page 159.

Le décodage de l'algorithme se fait en lisant les lettres du code jusqu'à trouver un mot de

Algorithme de Huffman

Entrées : une séquence S de N symboles.

Sortie : code de Huffman.

- 1 Ordonner les N symboles par ordre croissant de leur nombre d'occurrences dans S , chaque couple (symbole, occurrence) étiquettant un arbre réduit à sa racine.
- 2 **Tant qu'**il reste plus d'un arbre binaire
- 3 Retirer de la liste les deux arbres binaires de poids minimum
- 4 Fusionner ces arbres en un troisième de poids la somme des poids de ces fils
- 5 Affecter à l'arc vers le fils de droite la valeur 1 et vers celui de gauche la valeur 0
- 6 Insérer ce nouvel arbre dans la liste
- 7 **Fin Tant que**
- 8 Affecter à chaque feuille la concaténation de la valeur des arcs depuis la racine
- 9 **Renvoyer** pour chaque feuille, le couple (symbole, valeur binaire)

FIG. 5.17 – Algorithme de construction du code de Huffman

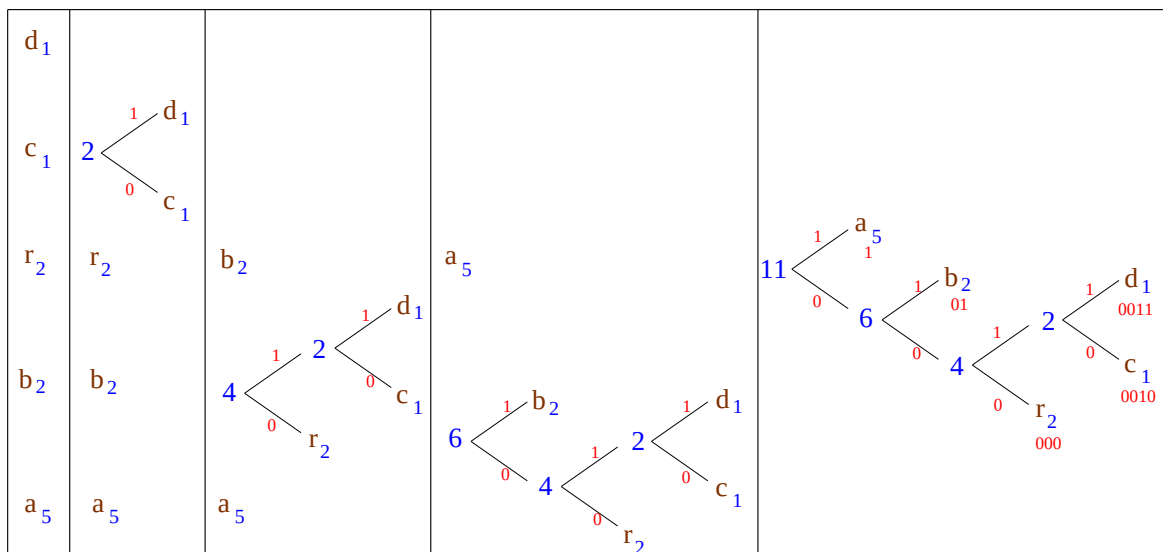


FIG. 5.18 – Algorithme de Huffman pour $S = \text{« abracadabra »}$

code présent dans le dictionnaire. Le code étant préfixé, aucune ambiguïté sur le mot de code n'est possible et le décodage est univoque. Le dictionnaire étant construit en fonction des données à coder, il doit être fourni au décodeur.

La dernière étape du format JPEG est un codage entropique de Huffman de la valeur S de la différence des coefficients DC et des paires (RL, S) pour les coefficients AC. Deux cas sont alors possibles. Soit les dictionnaires sont transmis dans l'entête du fichier JPEG

symbole	code
a	1
b	01
r	000
c	0010
d	0011

TAB. 5.5 – Code de Huffman pour $\mathcal{S}$ =« abracadabra »

pour le décodage, soit ce sont les numéros de dictionnaires prédéfinis par la norme. Ces dictionnaires prédéfinis ont été déterminés expérimentalement par le groupe d'experts JPEG et sont en moyenne très performants. La plupart des applications manipulant le format JPEG utilisent les dictionnaires prédéfinis.

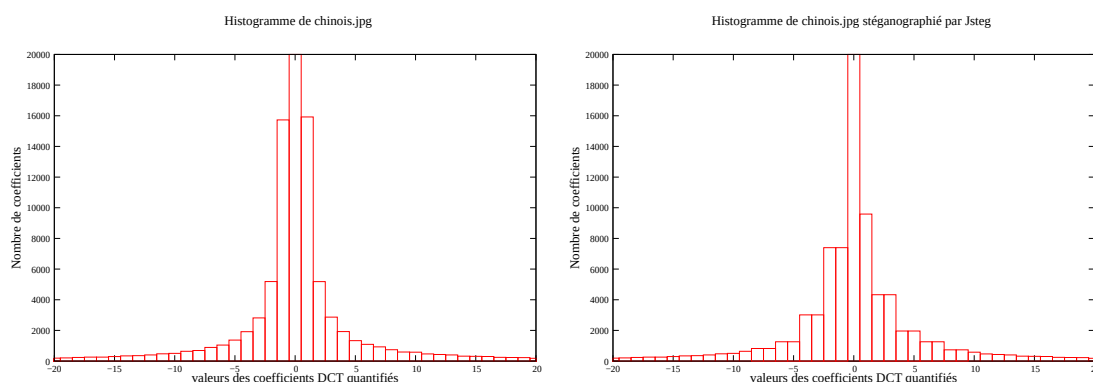
5.3 Stéganographie adaptée au format JPEG

Les étapes présentées au paragraphe précédent peuvent se regrouper en deux étapes. La première étape prend en entrée une image non compressée, appartenant au domaine spatial et transforme celle-ci en coefficients DCT quantifiés, appartenant au domaine fréquentiel. Cette première transformation n'est pas bijective, elle correspond à une étape de compression avec perte d'information. La seconde étape prend en entrée des coefficients DCT du domaine fréquentiel et les transforme en une suite de données binaires dans le domaine fréquentiel compressé. Cette transformation est bijective et correspond à une compression sans perte d'information. Le format JPEG ne laisse donc pas le choix quant aux données qui peuvent recevoir l'information à dissimuler. En effet, la première étape étant avec perte, nous ne pouvons pas insérer de l'information au cours de celle-ci sans risque de ne pouvoir l'extraire. D'autre part, modifier ne serait ce que quelques bits dans les données binaires du fichier JPEG implique nécessairement un décodage de Huffman suffisamment erroné pour que l'image stéganographiée ne ressemble en rien à une image naturelle. La seule possibilité est alors de dissimuler le message à l'aide des coefficients DCT.

Lorsque les premiers algorithmes de stéganographie tels *Jsteg* [193] ou *Outguess* [156] dans sa version première, ont été spécifiés, une attaque proposée par A. Westfeld et A. Pfitzmann [204] a mis en évidence des déviations statistiques triviales sur l'histogramme des coefficients DCT quantifiés. Par exemple, comme l'illustre les figures 5.19 page suivante et 5.20 page suivante, *Jsteg* a tendance à égaliser des paires de coefficients DCT quantifiés. À partir d'un simple test du χ^2 , ils ont alors mis au point des détecteurs stéganographiques permettant de discriminer les stégo média des supports de couverture. Des rustines ou de nouveaux algorithmes ont alors été proposés, tenant compte des nouvelles contraintes imposées par cette attaque. Nous présentons dans ce paragraphe, trois algorithmes de stéganographie dédiés au format JPEG conçus pour préserver les statistiques du premier ordre des coefficients DCT quantifiés. Ces algorithmes nous servent au chapitre 8 à illustrer nos techniques de stéganalyse.

5.3.1 Outguess

Outguess [155] est l'un des tout premiers algorithmes de stéganographie dédié aux images et issu de la communauté scientifique. Sa version première, (0.13), de 1998, est

FIG. 5.19 – Image *chinois.jpg*FIG. 5.20 – Modifications de l’histogramme de *chinois.jpg* par Jsteg

malheureusement sensible à l’attaque de A. Westfeld et A. Pfitzmann [204]. Son auteur, N. Provos propose alors l’ajout d’une deuxième passe visant à corriger les distortions introduites lors de l’insertion [156]. *Outguess* dissimule l’information dans les LSB des coefficients DCT quantifiés du format JPEG. Il utilise RC4 pour chiffrer le message et sélectionner aléatoirement les coefficients DCT et propose d’utiliser un codage de parité.

Dans un premier temps, le message M de longueur l à dissimuler est chiffré avec l’algorithme de chiffrement flot, RC4 [169], en un message M' . *Outguess* propose en option l’utilisation d’un code de parité après le chiffrement. Dans un deuxième temps, l’algorithme simule l’insertion de M' avec plusieurs Vecteurs d’Initialisation (IV), IV_i . L’IV qui minimise le nombre de changements dans les LSB des coefficients DCT est alors retenu. *Outguess* insère tout d’abord un premier registre de 32 bits constitué de la concaténation de l’IV et de la longueur du message, chacun codé sur 16 bits. Le GPA est ensuite ré-initialisé avec l’IV. L’insertion s’effectue en forçant les LSB des coefficients DCT choisis à la valeur des bits à insérer. L’ordre des coefficients est fixé par la suite des indexes des coefficients DCT (pos_i) définie par

$$\begin{aligned} pos_0 &= 0, \\ pos_i &= pos_{i-1} + R(x(i)), \end{aligned} \tag{5.6}$$

où $R(x(i))$ est une valeur tirée aléatoirement dans $[1, x(i)]$ et

$$x(i) = \begin{cases} x(i-1) & \text{si } i \bmod 8 \neq 0, \\ 2 \times \frac{\text{nombre de coefficients DCT non utilisés}}{\text{nombre de bits restant à insérer}} & \text{sinon.} \end{cases}$$

À la fin de l'étape d'insertion du message chiffré, *Outguess* corrige dans une seconde étape les distortions introduites sur l'histogramme des coefficients DCT. Pour ce faire, chaque modification est compensée par une modification inverse. Une modification par insertion LSB change un coefficient DCT $2i$ en $2i+1$ et inversement. Si chaque inversion est corrigée alors l'histogramme est globalement inchangé. L'objectif des corrections apportées par *Outguess* est de maintenir identiques les écarts $f_{2i+1} - f_{2i}$ entre deux valeurs adjacentes $2i$ et $2i+1$ de l'histogramme, avant et après l'insertion tout en restant proche de f_{2i+1} et f_{2i} . Pour chaque valeur f_i de l'histogramme un nombre maximum de changements T_i est toléré au cours du processus de correction. Si ce seuil est dépassé alors l'algorithme essaie de compenser la modification courante. À la fin, *Outguess* essaie de corriger les modifications résiduelles. L'algorithme **corriger**(*pos*, *val*) consiste à trouver un coefficient DCT d'index inférieur à *pos* de valeur adjacente à *val*, non déjà utilisé pour l'insertion ou la correction et à inverser son LSB.

D'autre part, N. Provos associe à chaque coefficient DCT une valeur de *déteçtabilité*, D qui n'est pas définie dans [156] mais qui apparaît clairement lorsque l'on regarde de plus près les sources de *Outguess* [155]. D est donnée par la relation

$$D(DCT) = \begin{cases} 1 & \text{si } DCT \leq 16, \\ 0 & \text{si } 16 < DCT < 240, \\ -1 & \text{si } DCT \geq 240. \end{cases}$$

La déteçtabilité de l'insertion est la somme des déteçtabilités des coefficients DCT modifiés par la dissimulation du message moins la somme des déteçtabilités des coefficients DCT modifiés par la correction statistique. La *distortion* introduite par *Ougtuess* est alors définie comme la somme du nombre de coefficients modifiés et de la déteçtabilité de l'insertion. Une autre optimisation qui apparaît aussi dans les sources consiste à utiliser prioritairement les coefficients les moins déteçtables pour la correction statistique. Le processus de correction est résumé dans la figure 5.21 page suivante. Les algorithmes d'insertion et d'extraction sont résumés aux figures 5.22 page 163 et 5.23 page 164. Le lecteur doit néanmoins avoir à l'esprit que l'IV est un paramètre public et peut donc être choisi comme étant celui qui entraîne la distortion la plus faible. D'autre part, les algorithmes *Enc*(.) et *Dec*(.) correspondent aux applications de codage et de décodage par un code correcteur d'erreurs. Ces applications sont soit l'identité (absence de codage correcteur) soit un code parité.

La figure 5.25 page 165 représente l'évolution de l'histogramme de la figure 5.24 page 164 tandis que la figure 5.26 page 165 illustre la différence des histogrammes avant et après l'insertion par *Outguess*. D'après la figure 5.27 page 165, les pixels modifiés se répartissent sur l'ensemble de l'image et préférentiellement autour des zones non homogènes, sur les contours.

5.3.2 F5

F5 est un algorithme de stéganographie ± 1 sur les coefficients DCT quantifiés, conçu par A. Westfeld [203, 202] en 1999. *F5* conjugue différentes techniques pour conserver l'his-

Algorithme de correction statistique d'Outguess

Entrées : f l'histogramme des coefficients DCT,
 $(DCT_i)_{i=1\dots n}$ les coefficients DCT.

Sortie : $(DCT'_i)_{i=1\dots n}$ les coefficients DCT corrigés.

```

1   $T_i \leftarrow (150/n)f_i$ 
2   $Mod[i] \leftarrow 0$ 
3  Pour  $i$  de 1 à  $n$ 
4    Si  $DCT_i$  non modifié alors continuer en 3
5     $Adj \leftarrow DCT_i \oplus 1$ 
6    Si  $Mod[Adj] \neq 0$  alors
7       $Mod[Adj] \leftarrow Mod[Adj] - 1$ 
8      continuer en 3
9    Fin Si
10   Si  $Mod[DCT_i] < T_{DCT_i}$  alors
11      $Mod[DCT_i] \leftarrow Mod[DCT_i] + 1$ 
12     continuer en 3
13   Fin Si
14   Si corriger( $i, DCT_i$ ) échoue alors
15      $Mod[DCT_i] \leftarrow Mod[DCT_i] + 1$ 
16     continuer en 3
17   Fin Si
18 Fin Pour
19 Pour chaque  $Mod[i] \neq 0$ 
20   Tant que  $Mod[i] \neq 0$ 
21      $Mod[i] \leftarrow Mod[i] - 1$ 
22     corriger( $n, DCT_i$ )
23   Fin Tant que
24 Fin Pour
25 Renvoyer  $(DCT'_i) = (DCT_i)$ 

```

FIG. 5.21 – Algorithme de correction statistique d'Outguess

togramme des coefficients DCT quantifiés proche de celui d'une image naturelle, d'une part et minimiser le nombre de changements à effectuer sur les coefficients DCT, d'autre part.

Afin de préserver l'histogramme des coefficients DCT quantifiés, l'information est insérée en fixant le LSB du coefficient DCT porteur à la valeur du bit du message à insérer. Si le LSB du coefficient et le bit du message sont identiques, le coefficient est laissé inchangé. Dans le cas contraire, la valeur absolue du coefficient est décrémentée de 1. De plus, les histogrammes des images naturelles présentent plus de coefficients non nuls impairs que pairs. Pour conserver cette propriété, les coefficients DCT positifs et im-

Algorithme d'insertion d'Outguess

Entrées : I une image non compressée,
 M un message,
 K_s une clé stéganographique,
 IV un IV.

Sortie : I' une stégo image,
 d distortion.

```

1  Compresser  $I$  jusqu'à la quantification
2   $(DCT'_i) \leftarrow (DCT_i)$ 
3   $M' \leftarrow Enc(RC4(K_s, M))$ 
4   $l' \leftarrow \text{longueur}(M')$ 
5  Initialiser le GPA à l'aide de  $K_s$ 
6  Générer  $(pos_i)$  selon (5.6)
7   $M_1 \leftarrow IV || l'$ 
8  Pour  $i$  de 1 à 32
9     $LSB(DCT'_{pos_i}) \leftarrow M_1(i)$ 
10 Fin Pour
11 Initialiser le GPA à l'aide de  $IV$ 
12 Générer  $(pos_i)$  selon (5.6)
13 Pour  $i$  de 1 à  $l'$ 
14    $LSB(DCT'_{pos_i}) \leftarrow M'(i)$ 
15 Fin Pour
16  $E_1 \leftarrow \{DCT'_i \neq DCT_i\}$ 
17  $f \leftarrow \text{histogramme des } (DCT'_i)$ 
18  $DCT' \leftarrow \text{Correction}(f, (DCT'_i))$ 
19 Terminer la compression JPEG
20  $E_2 \leftarrow \{DCT'_i \neq DCT_i\} \setminus E_1$ 
21  $d \leftarrow \text{Card}(E_1 \cup E_2) + \sum_{c \in E_1} D(c) - \sum_{c \in E_2} D(c)$ 
22 Renvoyer  $I'$  et  $d$ 

```

FIG. 5.22 – Algorithme d'insertion d'Outguess

pairs ainsi que les coefficients négatifs et pairs codent un bit de message de valeur 1, les autres codant un bit de message à 0, comme l'illustre la figure 5.28 page 166. De même, la symétrie devant être conservée, les coefficients DCT égaux à 0 sont donc ignorés. D'autre part, les coefficients 0 étant ignorés, le récepteur ne peut faire la différence entre un 0 de l'image et un 0 produit par le changement d'un -1 ou 1. Si un tel changement survient lors de l'insertion, le bit du message est ré-inséré de nouveau dans le prochain coefficient DCT. On parle alors d'*effondrement*.

D'autre part, pour minimiser les changements introduits, une randomisation et un co-

Algorithme d'extraction d'Outguess

Entrées : I' une stégo image JPEG,
 K_s une clé stéganographique.

Sortie : M un message.

```

1  Décompresser  $I'$  jusqu'à la quantification
5  Initialiser le GPA à l'aide de  $K_s$ 
6  Générer  $(pos_i)$  selon (5.6)
8  Pour  $i$  de 1 à 32
9     $M_1(i) \leftarrow LSB(DCT'_{pos_i})$ 
10 Fin Pour
7   $M_1 \rightarrow IV || l$ 
11 Initialiser le GPA à l'aide de  $IV$ 
12 Générer  $(pos_i)$  selon (5.6)
13 Pour  $i$  de 1 à  $l$ 
14    $M(i) \leftarrow LSB(DCT'_{pos_i})$ 
15 Fin Pour
3   $M \leftarrow Dec(RC4^{-1}(K_s, M))$ 
22 Renvoyer  $M$ 

```

FIG. 5.23 – Algorithme d'extraction d'*Outguess*



FIG. 5.24 – Image *tahiti.jpg*

dage par syndrome sont mis en œuvre. En utilisant un GPA, les coefficients DCT sont mélangés avant l'insertion. Cela a pour effet d'uniformiser la répartition des changements sur l'ensemble des coefficients DCT. Sans cette étape de randomisation, les changements sont localement concentrés sur les premiers coefficients. L'emploi du codage par syndrome en stéganographie a initialement été proposé par R. Crandall [56] en 1999. Depuis, l'utili-

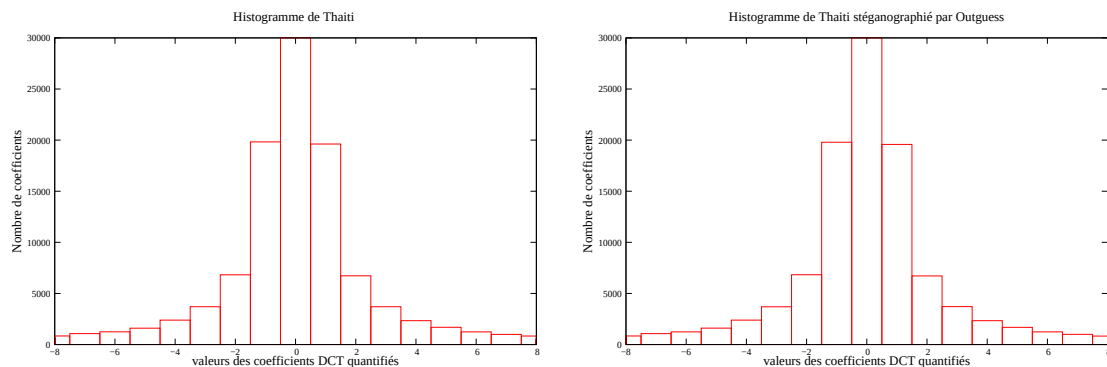
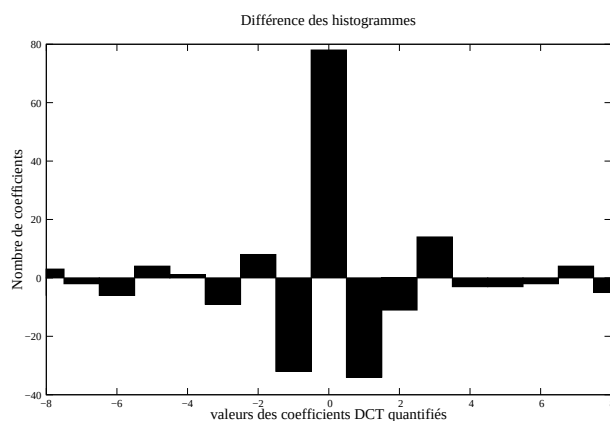
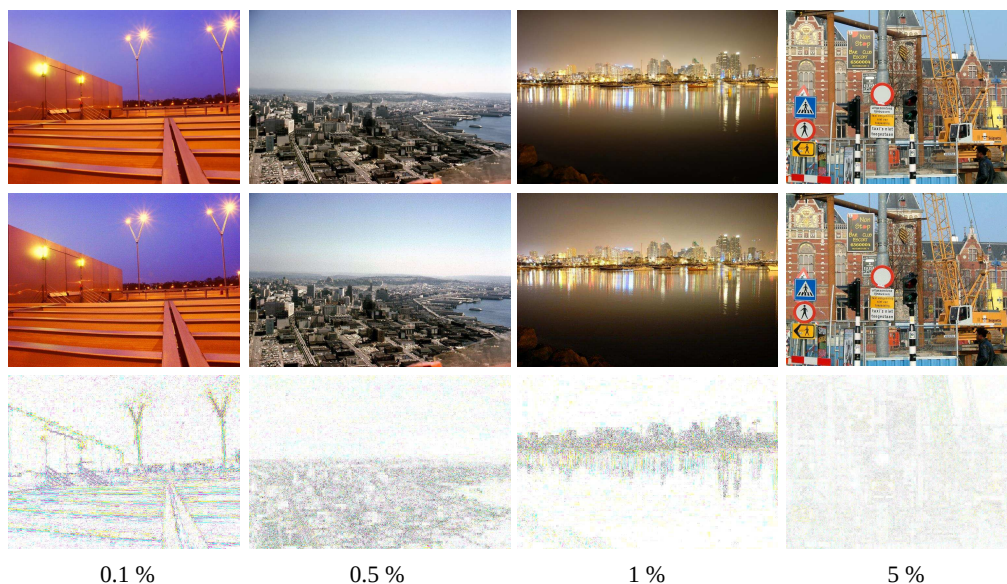
FIG. 5.25 – Modifications de l'hitogramme de *tahiti.jpg* par *Outguess*

FIG. 5.26 – Différence des histogrammes avant et après insertion

FIG. 5.27 – Images stéganographiées par *Outguess* pour différents taux stéganographiques

sation de codes parfaits et de codes MDS (*Maximum Distance Separable*) a été largement étudiée. Nous en donnons ici les grands principes. Pour plus de détails, notamment pour

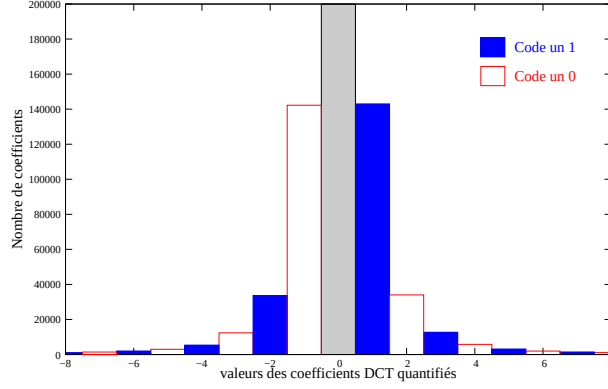


FIG. 5.28 – Codage des bits du message par les coefficients DCT

le choix de « bons » codes, le lecteur pourra se référer à [74, 80, 81, 82, 84, 85, 167, 168].

Notons $\mathbb{F}_q = GF(q)$, le corps de Galois à q éléments (en général, $q = 2$). Soit $\mathcal{C}$ un code linéaire de longueur n , de dimension k et de matrice de parité H . Nous rappelons que par définition, $\mathcal{C} = \{c \mid cH^t = 0\}$ est un sous-espace vectoriel de $\mathbb{F}_q^n$ de dimension k . Nous appelons *syndrome* du vecteur $y \in \mathbb{F}_q^n$ le vecteur $S(y) = yH^t$ de longueur $r = (n - k)$. Dans un schéma classique de communication, l'émetteur code le message x en $y = C(x)$, où C est l'application de codage définie de $\mathbb{F}_q^k$ sur $\mathbb{F}_q^n$. Le mot de code y est alors envoyé sur le canal. Le récepteur reçoit $y' = y + e$, où e est le vecteur de bruit introduit par le canal. Pour retrouver le message x , le récepteur doit tout d'abord évaluer l'erreur e et la soustraire de y' puis décoder $(y' - e)$ et retrouver enfin x . Par ailleurs, nous avons une relation liant les syndromes de e et de y' .

$$S(y') = S(y + e) = (y + e)H^t = yH^t + eH^t = eH^t = S(e) \in \mathbb{F}_q^{(n-k)}.$$

Tout le problème est donc de retrouver e à partir de son syndrome. En règle générale, mettre en évidence une telle application est un problème difficile. Le lecteur intéressé par la théorie des codes pourra se référer à [103] pour plus de précisions concernant les choix de codes et l'existence ou non d'algorithmes de décodage pour des codes donnés.

Le principe du codage de syndrome est d'associer à un message à dissimuler m , de longueur $r = (n - k)$ symboles de $\mathbb{F}_q$, en utilisant un vecteur v de n symboles du support de couverture. Pour ce faire, v est transformé par l'algorithme d'insertion en v' vecteur de n symboles, tel que $S(v') = m$. En d'autres termes, on cherche v' dont le syndrome est exactement le message que l'on souhaite dissimuler. D'autre part, nous voulons que le nombre de modifications pour passer de v à v' soit minimum, *i.e.* $(v' - v)$ de poids faible. Les algorithmes d'insertion et d'extraction, $\mathcal{Emb}$ et $\mathcal{Ext}$, du schéma de stéganographie vérifient alors

$$\begin{aligned} \mathcal{Emb}(v, m) &= v + \Delta(m, v) = v' \text{ avec } w(\Delta(v, m)) \leq \rho, \\ \mathcal{Ext}(v') &= S(v') = m, \end{aligned}$$

où ρ est le nombre maximum de modifications autorisées pour n symboles du support. Notons D l'application qui, à un syndrome s , associe un vecteur de poids borné de syndrome

s ,

$$\begin{aligned} D : \mathbb{F}_q^{(n-k)} &\longrightarrow \mathbb{F}_q^n \\ y &\longrightarrow D(y), \end{aligned}$$

telle que $S(D(y)) = y$ et $w(D(y)) \leq \rho$. De plus, nous appelons *rayon de couverture d'un code* C , noté $\rho(C)$, l'entier défini par

$$\rho(C) = \min\{\rho \text{ tel que } \{S(y) \mid w(y) \leq \rho\} = \mathbb{F}_q^{(n-k)}\}.$$

Tout comme D , le rayon de couverture est en général difficile à évaluer. Afin de minimiser les distortions introduites lors de l'insertion, nous aurons plutôt tendance, lorsque les contraintes le permettent, à utiliser des codes pour lesquels le rayon de couverture est minimum. De tels codes sont appelés *codes parfaits*. C. Fontaine et F. Galand [74] préconisent préférentiellement des codes MDS lorsque certains symboles doivent impérativement demeurer inchangés au cours du processus d'insertion et qu'il est impossible de les déterminer à l'avance. Les codes MDS sont des codes pour lesquels la distance minimale est maximale. Pour répondre à nos contraintes, nous prenons

$$\Delta(v, m) = D(m - S(v)) \text{ avec } d_H(v, D(m - S(v))) \leq \rho, \quad (5.7)$$

où d_H est la distance de Hamming. Le problème du codage de syndrome est alors équivalent au problème de décodage du syndrome de poids borné, qui est en général un problème difficile.

L'algorithme *F5* utilise des codes parfaits, les codes de Hamming. Ces codes sont paramétrés par s , leur longueur vaut $n = 2^s - 1$ et leur dimension $k = n - s$. De plus, l'algorithme de décodage est connu ainsi que son rayon de couverture qui vaut 1. Le taux stéganographique est inférieur à $s/(2^s - 1)$ et pour dissimuler un message de s bits, au plus un seul bit parmi n est modifié. En fonction de la taille du message et du support, *F5* choisit le paramètre s adapté. Le code est défini par sa fonction syndrome,

$$\begin{aligned} S : \mathbb{F}_2^n &\longrightarrow \mathbb{F}_2^s \\ y &\longrightarrow S(y) = \bigoplus_{i=1}^n y_i \cdot i, \end{aligned} \quad (5.8)$$

où i est sous sa forme binaire, codé sur s bits. Soit à insérer le message m de s bits dans le vecteur support v de n bits, nous évaluons alors

$$m - S(v) = m \oplus S(v), \quad (5.9)$$

et $j = D(m - S(v))$ est défini comme étant l'entier correspondant à $m - S(v)$. Si $j = 0$ alors $m = f(v)$ et aucune modification n'est nécessaire, dans le cas contraire le bit v_j doit être inversé. Pour ce faire, la valeur absolue du coefficient DCT de LSB v_j est décrétementée. Si un effondrement apparaît, v_j est remplacé par v_{j+1} , v_{j+1} par v_{j+2} , etc ... et le LSB du prochain coefficient DCT non nul est inséré en v_n . Les algorithmes d'insertion et d'extraction sont résumés dans les figures 5.29 page suivante et 5.30 page 169.

La figure 5.32 page 170 représente l'évolution de l'histogramme de la figure 5.31 page 169 tandis que la figure 5.33 page 170 illustre la différence des histogrammes avant et après l'insertion par *F5*. D'après la figure 5.34 page 171, les pixels modifiés se répartissent sur l'ensemble de l'image et préférentiellement autour des zones non homogènes, sur les contours.

Algorithme d'insertion de F5

Entrées : un message secret $M = m_1 m_2 \dots$ de l bits,
 I une image non compressée,
 K_s la clé stéganographique.

Sortie : une image stéganographiée.

```

1  Compresser  $I$  au format JPEG, jusqu'à la quantification
2  Initialiser le GPA à l'aide de  $K_s$ 
3  Mélanger aléatoirement les coefficients DCT à l'aide du GPA
4  Déterminer le paramètre  $s$  à partir de  $l$  et  $I$ 
5  Pour chaque bloc de message  $m_i$  de longueur  $s$ 
6      Remplir un buffer  $v$  avec les  $n$  LSB des coefficients DCT non nuls suivants
7      Calculer  $j = m_i \oplus S(v)$  selon (5.9)
8      Si  $j \neq 0$  alors décrémenter la valeur absolue du coefficient DCT de LSB  $v_j$ 
9      Fin Si
10     Si un effondrement apparaît alors
11          $v_k \leftarrow v_{k+1}$  pour  $k = j \dots (n-1)$ 
12          $v_n \leftarrow$  LSB du prochain DCT non nul
13     Retour à l'étape 7
14 Fin Si
15 Fin Pour
16 Finir la compression JPEG en  $I'$ 
17 Renvoyer  $I'$ 
```

FIG. 5.29 – Algorithme d'insertion de $F5$

5.3.3 JPHide and JPSeek

JPHide and *JPSeek* est un algorithme de stéganographie implémenté par A. Latham [124] en 1999 selon deux versions, 0.3 et 0.5. La version 0.5 intègre en plus un algorithme de compression du message à dissimuler. Malheureusement, les spécifications de cet algorithme n'ont jamais été publiées. Nous en donnons ici une description, bien moins détaillée que celles d'*Outguess* et de *F5*, à partir de notre compréhension des codes sources et de [157].

Le message à dissimuler est tout d'abord chiffré avec l'algorithme Blowfish [166]. La clé cryptographique est un mot de passe tronqué à 120 caractères. Dans un premier temps, les 8 premiers coefficients DCT sont réduits modulo 256 et concaténés en un bloc de 64 bits. Le mécanisme de dérivation de clé est initialisé avec le mot de passe et le premier bloc est chiffré. Dans un deuxième temps, la taille du message à insérer est codée sur les 24 bits premiers bits du bloc chiffré ; les 40 restants constituant un vecteur d'initialisation (IV). Ce bloc de 64 bits est ensuite concaténé au message chiffré avant l'insertion.

Algorithme d'extraction de F5

Entrées : I' une image JPEG,
 K_s la clé stéganographique,
 s le paramètre du code,
 l la longueur du message.

Sortie : le message $M = m_1m_2\dots$

- 1 **Décompresser** I' jusqu'à la quantification
- 2 **Initialiser** le GPA à l'aide de K_s
- 3 **Mélanger** aléatoirement les coefficients DCT à l'aide du GPA
- 5 **Pour** les l blocs de message m_i de longueur s
- 6 **Remplir** un buffer v' avec les n LSB des coefficients DCT non nuls suivants
- 7 **Calculer** $m_i = S(v')$ selon (5.8)
- 8 **Fin Pour**
- 9 **Renvoyer** $M = m_1m_2\dots$

FIG. 5.30 – Algorithme d'extraction de F5



FIG. 5.31 – Image *soldat.jpg*

JPHide dissimule le message dans les bits de poids faible des coefficients DCT mais il s'autorise aussi à modifier ponctuellement certains bits du plan 1. D'autre part, les coefficients -1, 0 et 1 bénéficient d'un traitement particulier. Enfin, l'ordre de parcours des coefficients DCT est fixé par une table statique et n'est pas randomisé. En revanche, avec une certaine probabilité dépendant de la longueur du message à dissimuler et de la taille de l'image, le coefficient DCT courant est ignoré. Ces coefficients sont choisis à l'aide du GPA.

La figure 5.36 représente l'évolution de l'histogramme de la figure 5.35 tandis que la figure 5.37 page 172 illustre la différence des histogrammes avant et après l'insertion par

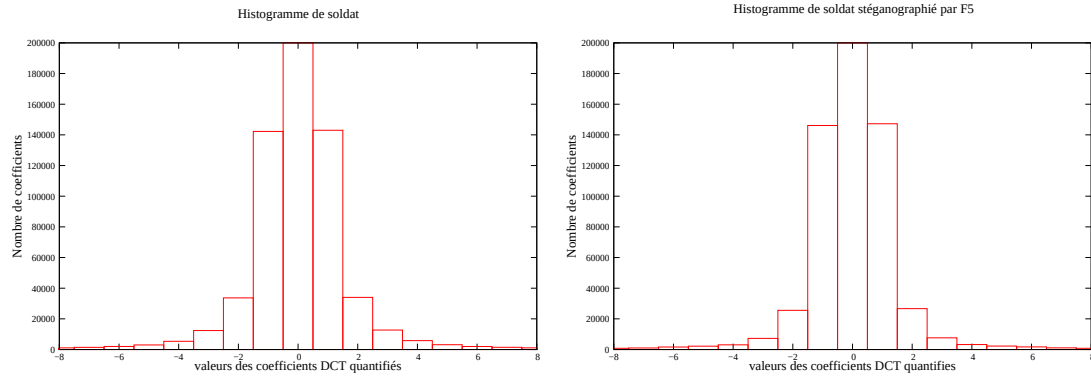


FIG. 5.32 – Modifications de l'histogramme de l'image *soldat.jpg* par *F5*

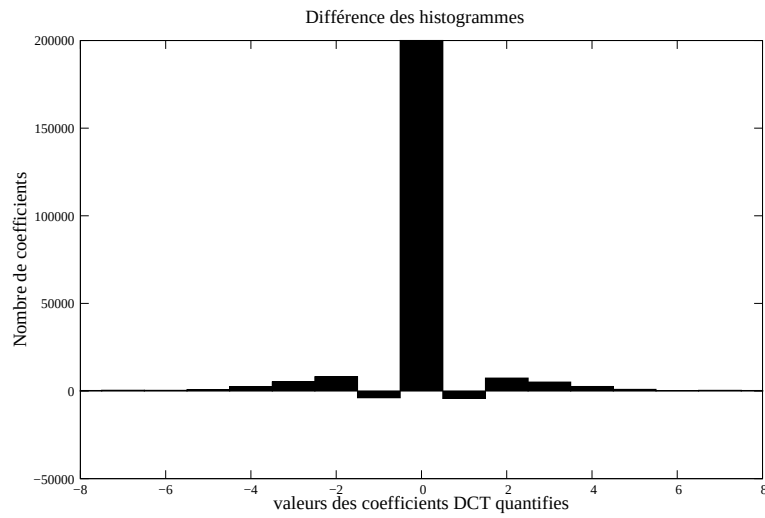
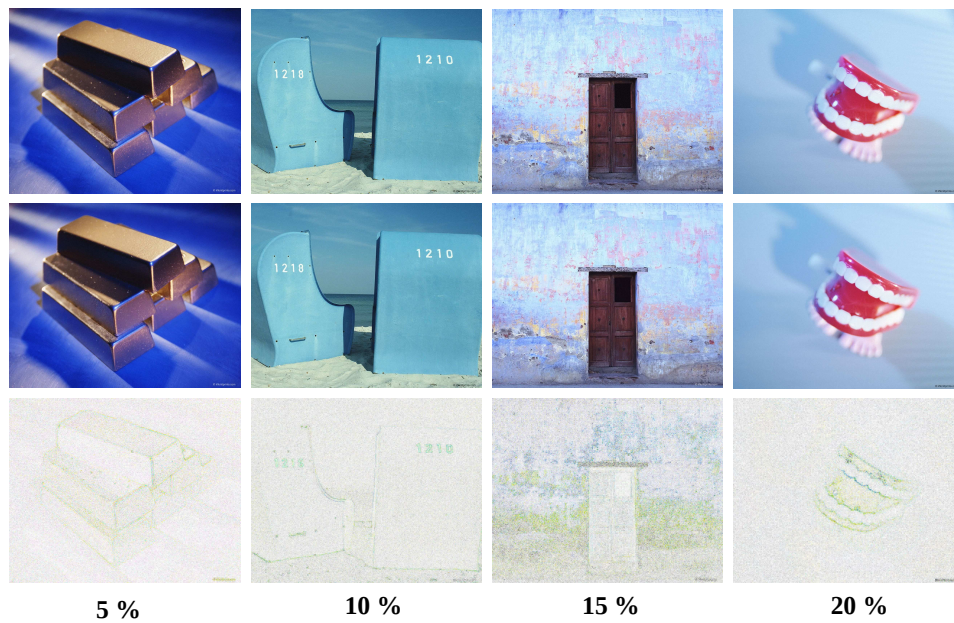
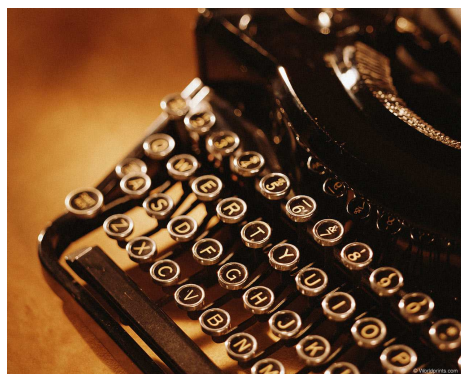
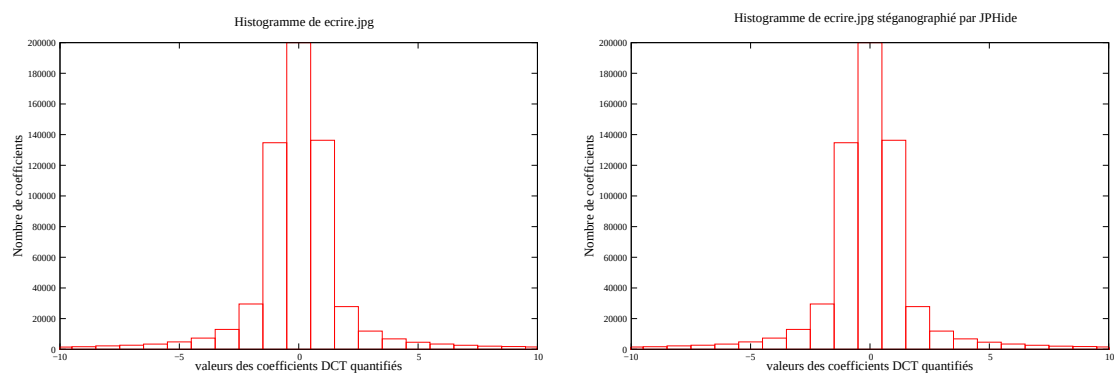


FIG. 5.33 – Différence de l'histogramme de l'image *soldat.jpg* après et avant insertion

JPHide. D'après la figure 5.34 page suivante, les pixels modifiés se répartissent sur l'ensemble de l'image et préférentiellement autour des zones non homogènes, sur les contours.

FIG. 5.34 – Images stéganographiées par *F5* pour différents tauxFIG. 5.35 – Image *ecrire.jpg*FIG. 5.36 – Modifications de l'histogramme de l'image *ecrire.jpg* par *JPHide*

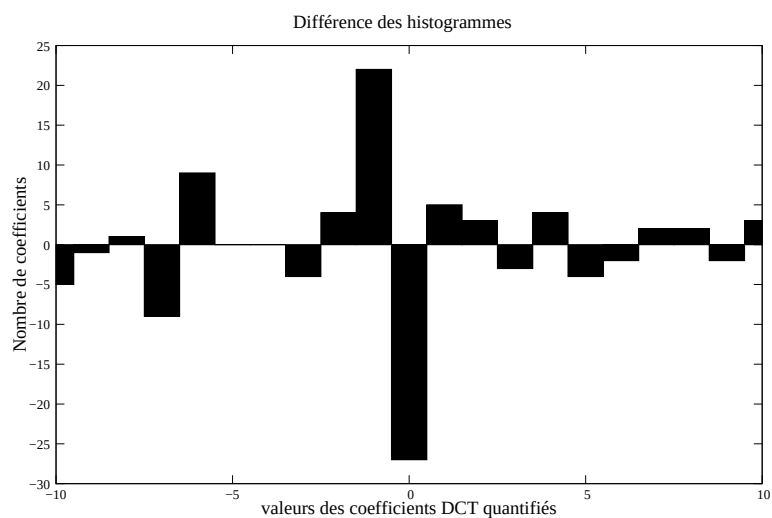


FIG. 5.37 – Différence de l'histogramme de l'image *ecrive.jpg* après et avant insertion

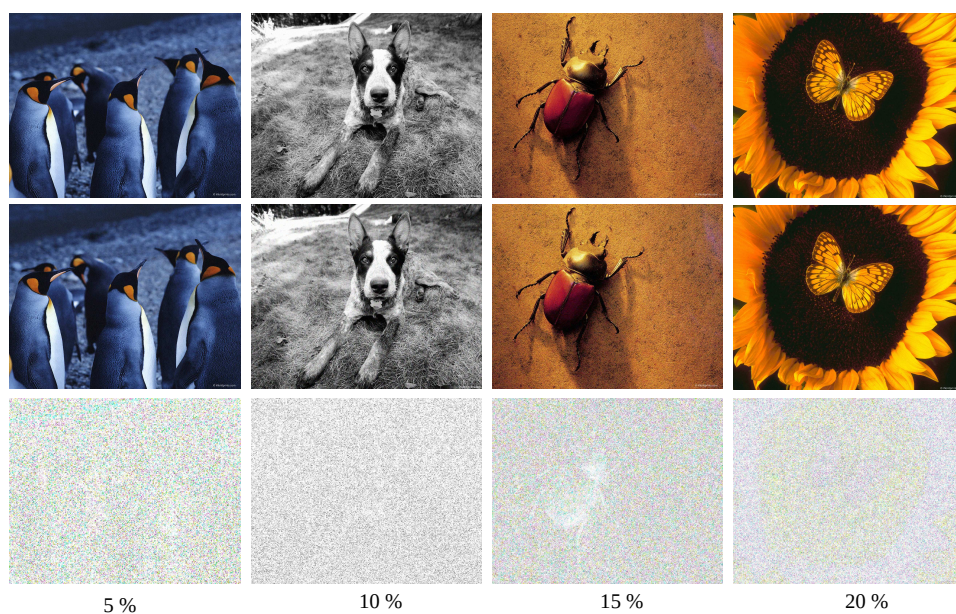


FIG. 5.38 – Images stéganographiées par *JPHide* pour différents taux

Introduction à la stéganalyse

*« Connais l'adversaire et surtout connais toi
toi-même et tu seras invincible. »*

Sun Tzu

Sommaire

6.1	Un problème de discrimination	175
6.2	Modèles d'attaquants	178
6.2.1	Modèle d'indistingabilité	178
6.2.2	Attaquant spécifique	182
6.2.3	Attaquant universel	185
6.2.4	Évaluation de la sécurité contre une attaque	187
6.3	Distingueur de Fisher	189
6.3.1	L'analyse discriminante	189
6.3.2	Discrimination particulière restreinte à deux classes	192
6.4	Conclusion	193

NOUS avons présenté au chapitre précédent une introduction aux techniques de stéganographie, notamment dédiées à l'image fixe, compressée ou non. Par ailleurs, dans l'introduction, nous avons expliqué que la stéganographie apporte un service de sécurité supplémentaire aux données numériques. La sécurité de transmission (TRANSEC), jusque là réservée aux communications sur des canaux physiques, trouve ainsi son expression dans le domaine numérique. Tout naturellement, nous étudions dans ce chapitre comment évaluer la sécurité TRANSEC des algorithmes décrits au chapitre 5. Pour bien appréhender ce nouveau service de sécurité, nous pouvons considérer que dissimuler de l'information dans un support est un processus similaire à l'émission d'un signal numérique dans un canal physique. Cette analogie fait correspondre le message caché au signal émis dans le canal, le support numérique au canal lui-même, l'algorithme d'insertion à la partie codage de canal (codage et modulation du signal) et enfin l'algorithme d'extraction à la partie réception et décodage de canal (démodulation et décodage du signal). Par ailleurs, la borne de Shannon nous indique que la capacité d'un canal est limitée et que cette limite dépend du modèle de canal considéré et de la façon de coder l'information transmise. De la même manière, la capacité d'un support numérique est bornée et cette limite dépend

support et de l'algorithme d'insertion. Cette analogie forte entre la stéganographie et le monde des communications laisse supposer une adaptation aisée des concepts de la théorie de l'information pour modéliser les mécanismes stéganographiques. C. Cachin [38, 39, 40] relayé par R. Chandramouli [45, 46, 47] ont abordé la sécurité et la capacité des algorithmes de stéganographie¹ sous cet angle. C. Cachin est le premier à définir la notion de sécurité d'un algorithme de stéganographie. Soit $\mathcal{C}$, l'ensemble des supports de couverture de distribution $P_{\mathcal{C}}$, $\mathcal{S}$, l'ensemble des stégo média de distribution $P_{\mathcal{S}}$, alors la sécurité d'un système stéganographique est donnée par l'entropie relative $D(P_{\mathcal{C}}||P_{\mathcal{S}})$ entre $P_{\mathcal{C}}$ et $P_{\mathcal{S}}$. Un système stéganographique est alors qualifié de ε -sûr contre un attaquant passif si et seulement si

$$D(P_{\mathcal{C}}||P_{\mathcal{S}}) = \sum_{c \in \mathcal{C}} P_{\mathcal{C}}(c) \log \frac{P_{\mathcal{C}}(c)}{P_{\mathcal{S}}(c)} \leq \varepsilon.$$

De plus si $\varepsilon = 0$ alors le système est qualifié de *parfaitement sûr*. Cette pseudo-distance est aussi appelée *distance de Kullbak-Liebler*. En d'autres termes, la sécurité d'un schéma de stéganographie dépend de l'incapacité de l'adversaire à distinguer deux distributions de probabilité, $P_{\mathcal{C}}$ et $P_{\mathcal{S}}$. Depuis, de nombreux modèles de sécurité en stéganographie ont été proposés. Différents types de schéma ont été étudiés, à clé secrète [39, 58, 101, 115] ou à clé publique [198, 40, 130] et différents types d'attaquant ont été envisagés, passif [39, 101, 130, 198] ou actif [40, 100]. La thèse de N. Hopper [99] est la première qui traite de la sécurité des schémas de stéganographie. Les différents modèles et leur différentes notations y sont rigoureusement définis en analogie avec la sécurité des schémas de cryptographie.

Dans le cadre de l'analyse des canaux dans un contexte non coopératif, notre attaquant passif se transpose dans le monde de la stéganographie en un attaquant passif dont l'objectif est d'avoir accès à l'information échangée sur le canal. Pour ce faire, notre *stéganalyste* intercepte un flux d'images qui transitent sur un réseau de communication. Il doit tout d'abord détecter la présence, ou non, d'information dissimulée dans les images. Dans un deuxième temps, celui-ci doit extraire (ou décoder) l'information insérée dans le support. Enfin, il doit cryptanalyser le message chiffré extrait. Dans le cadre de notre étude, nous nous intéressons exclusivement à la partie détection et évaluons la sécurité des algorithmes de stéganographie sous l'angle de la sécurité basée sur l'indistingabilité.

L'objectif de ce chapitre est tout d'abord de définir clairement et proprement le modèle d'attaquant que nous utilisons dans le cadre des stéganalyses présentées aux chapitres 7 et 8. Nous nous sommes fixés deux contraintes. Tout d'abord, le modèle défini doit être le plus fidèle possible à l'attaquant réel, c'est-à-dire celui invoqué implicitement dans tous les papiers de stéganalyse effective. Deuxièmement, ce modèle doit s'intégrer dans les modèles classiques de sécurité. Tandis que l'approche traditionnelle consiste à se placer du côté du concepteur et à évaluer la sécurité du schéma spécifié dans un modèle donné, nous nous plaçons du côté de l'attaquant réel et définissons ensuite le modèle de sécurité qui le caractérise le mieux. Cette démarche inverse s'explique par le fait que l'attaquant réel est extrêmement contraint et donc le modèle de sécurité associé est très faible. Cela implique que les stéganalyses effectives, notamment celles des chapitres 7 et 8 mettent en évidence des bornes sur l'insécurité pour le modèle considéré mais aussi pour tous les modèles plus

¹Par abus de langage nous parlons indifféremment de schéma ou système de stéganographie et d'algorithme de stéganographie.

forts. Enfin, en reliant notre modèle d'attaquant aux modèles classiques, nous montrons que les algorithmes que nous avons stéganalysés ne sont sûrs dans aucun des modèles classiques. Les récents travaux d'A.D. Ker [119] s'inscrivent dans une démarche similaire. Celui-ci propose en effet une méthodologie commune pour comparer les stéganalyses effectives entre elles et vise ainsi à réduire l'écart entre les modèles pratiques et théoriques.

En s'appuyant sur les modèles proposés par C. Cachin [39], S. Katzenbeisser et F. Petitcolas [115] et N. Hopper [99], nous formalisons les concepts d'*attaque par discrimination*, de *distingueur* et de *stéganalyse par discrimination* ; puis nous énonçons ce dernier comme un problème de discrimination statistique. Dans un deuxième temps, nous décrivons le modèle d'indistingabilité en cryptographie duquel nous nous inspirons pour définir le jeu de l'adversaire en stéganographie. Nous dégradons ensuite ce modèle pour être le plus proche de l'attaquant réel et définissons ainsi, au paragraphe 6.2, deux modèles d'attaquant réel, l'*attaquant spécifique* et l'*attaquant universel*. Chacun de ces attaquants peut être vu comme un attaquant très faible dans les modèles classiques. L'efficacité des stéganalyses effectives est finalement reliée à l'*insécurité* des algorithmes de stéganographie analysés dans les modèles proposés, mais aussi dans les modèles classiques. Enfin, nous détaillons au paragraphe 6.3, l'analyse discriminante de Fisher à la base de la conception des distingueurs mis en œuvre dans nos stéganalyses. Les résultats de ce chapitre feront l'objet d'une présentation lors de la conférence internationale Information Hiding 08 [11].

6.1 Un problème de discrimination

Nous prenons maintenant la place du stéganalyste et interceptons en aveugle des média numériques échangés entre Alice et Bob. Nous savons par ailleurs que ceux-ci utilisent des algorithmes de stéganographie pour sécuriser leurs communications. Avant de préciser notre attaquant, il nous faut définir formellement ce qu'est un schéma de stéganographie à clé secrète.

Définition 6.1 [115] *Un système de stéganographie à clé secrète Σ est la donnée d'un ensemble $\mathcal{C}$ de supports de couverture et de trois algorithmes polynomiaux, dont*

- *un algorithme probabiliste $\mathcal{K}$ de génération des clés. Il prend en entrée un paramètre de sécurité 1^k et renvoie une clé secrète K_s .*
- *un algorithme probabiliste $\mathcal{Emb}$ d'insertion. Il prend en entrée la clé K_s , un message clair $m \in \mathcal{M} \subset \{0,1\}^*$, un support de couverture $I \in \mathcal{C}$ et renvoie un stégo médium I' de $\mathcal{S}$, l'ensemble des stégo média.*
- *un algorithme déterministe $\mathcal{Ext}$ d'extraction. Il prend en entrée la clé privée K_s , un médium I' et renvoie le message clair m si I' appartient à $\mathcal{S}$ ou $\perp$, un message d'erreur sinon.*

Remarque 6.1 :

Tout comme en cryptographie, cette définition s'adapte aisément à un schéma de stéganographie à clé publique. Dans le cadre de notre étude, nous nous intéressons exclusivement à des schémas de stéganographie à clé secrète. Néanmoins, tout ce qui suit reste vrai ou se transpose sans difficulté pour des schémas à clés publiques. Dans un contexte plus général, cette transposition n'est en générale pas aussi directe et s'effectue souvent au prix

de quelques adaptations et beaucoup de précautions. Cette définition formelle s'accorde notamment parfaitement avec la définition pratique d'un système de stéganographie que nous avons énoncée dans notre introduction.

Dans ce contexte, nous jouons le rôle d'un attaquant passif qui ne modifie pas les informations échangées entre Alice et Bob. Notre objectif est, dans un premier temps, de différencier, parmi les média échangés, les supports de couverture des stégo média. Plus simplement, étant donné un médium I , nous devons répondre à la question « I est-il un stégo médium ? ». Ce problème est équivalent à distinguer un support de couverture $c \in \mathcal{C}$, connaissant $P_{\mathcal{C}}$ d'un stégo médium $s \in \mathcal{S}$ connaissant $P_{\mathcal{S}}$. La sécurité d'un schéma de stéganographie peut être définie à l'aide de n'importe quelle distance définie sur l'ensemble des distributions de probabilité définies sur un même support, comme par exemple la distance de Kullbak-Liebler proposée par C. Cachin. Pour notre modèle, nous utilisons la distance statistique $D(.,.)$ introduite par N. Hopper [99], définie par

$$D(P_{\mathcal{C}}, P_{\mathcal{S}}) = \frac{1}{2} \sum_{c \in \mathcal{C} \cup \mathcal{S}} |P_{\mathcal{C}}(c) - P_{\mathcal{S}}(c)|.$$

Nous qualifions alors un schéma de stéganographie d' ε -sûr si et seulement si

$$D(P_{\mathcal{C}}, P_{\mathcal{S}}) \leq \varepsilon,$$

et de parfaitement sûr si et seulement si $\varepsilon = 0$. Malheureusement, en pratique, nous ne pouvons pas évaluer $P_{\mathcal{C}}$, $P_{\mathcal{S}}$ et donc mesurer la sécurité du système stéganographique étudié. Néanmoins, nous pouvons confronter la robustesse de notre système aux attaques connues. Dans un premier temps, nous introduisons la notion d'*attaque par discrimination*.

Définition 6.2 Soit Σ un schéma de stéganographie. Nous appelons *attaque par discrimination contre Σ* , la donnée d'une fonction

$$\begin{aligned} V : \quad \mathcal{I} = \mathcal{C} \cup \mathcal{S} &\longrightarrow \mathcal{V}_1 \times \cdots \times \mathcal{V}_n \\ I &\longrightarrow V(I) = (V_1(I), \dots, V_n(I)), \end{aligned}$$

où V_i est une variable aléatoire définie sur $\mathcal{I}$ à valeurs dans $\mathcal{V}_i$ et calculable en temps polynomial.

Selon la définition 6.2, une attaque par discrimination contre un système de stéganographie est équivalente à la donnée de n mesures, coordonnées d'un *vecteur statistique*. Une attaque est efficace si elle permet de distinguer $\mathcal{C}$ de $\mathcal{S}$ ou de façon équivalente $V(\mathcal{C})$ de $V(\mathcal{S})$. L'objectif d'un attaquant passif est de mettre en évidence une attaque V par discrimination efficace contre Σ afin de répondre à la question « I est-il un stégo médium ? ». Clairement une attaque est efficace si et seulement si elle vérifie au moins l'un des deux critères suivants. Il existe $i \in [1, n]$ tel que $P_{V_i(\mathcal{C})} \neq P_{V_i(\mathcal{S})}$ ou il existe $i \in [1, n]$ tel que $P_{V_i|\{V_k, k \neq i\}(\mathcal{C})} \neq P_{V_i|\{V_k, k \neq i\}(\mathcal{S})}$. Le premier critère implique que $I \longrightarrow V_i(I)$ est aussi une attaque par discrimination efficace, ce qui n'est pas le cas pour le second. Si $V_i(I)$ n'est pas une attaque efficace, cela signifie que nous devons aussi observer les autres coordonnées de V pour obtenir de l'information sur I à partir de V_i . Une attaque efficace permet alors d'obtenir des distributions que l'on peut distinguer, encore faut-il construire un distingueur qui exploite une différence de distributions marginales, P_{V_i} , ou conditionnelles, $P_{V_i|\{V_k, k \neq i\}}$, évaluées sur les ensembles $\mathcal{C}$ et $\mathcal{S}$. En adaptant la définition de C. Cachin [39], nous proposons la suivante.

Définition 6.3 Soit Σ un schéma de stéganographie et V une attaque par discrimination contre Σ . Nous appelons *distingueur compatible avec V* , la donnée d'une fonction définie par

$$D_V : \mathcal{V}_1 \times \cdots \times \mathcal{V}_n \longrightarrow \{0, 1\},$$

calculable en temps polynomial. Par convention 0 est associé à non stégo et 1 à stégo.

Celle-ci nous permet alors d'annoncer la définition de la *stéganalyse par discrimination*, qui semble être implicitement la plus usitée dans la communauté.

Définition 6.4 Soit Σ un schéma de stéganographie. Nous appelons *stéganalyse par discrimination contre Σ* tout couple (V, D_V) , où V est une attaque contre Σ et D_V un distingueur compatible avec V .

La définition formelle de stéganalyse par discrimination que nous proposons reflète parfaitement le comportement de l'adversaire réel. En effet, un attaquant effectif procède en deux étapes distinctes. Il doit tout d'abord mettre en évidence un ensemble de mesures dont au moins l'une des distributions marginales ou conditionnelles diffère sur les média de couverture et sur les stégo média, comme par exemple les coefficients RS [77]. Dans un second temps, il doit mettre au point un distingueur qui va décider si un médium est stégo ou non en fonction des mesures effectuées. Il existe différents procédés pour construire des distingueurs, notamment les machines à support de vecteurs, une analyse en composante principale ou bien une analyse discriminante de Fisher. Le choix des mesures relève de l'expérience du stéganalyste tandis que la conception d'un distingueur relève de techniques statistiques dédiées aux tests d'hypothèses. De même, il convient de définir la sécurité d'un schéma contre une stéganalyse par discrimination.

Définition 6.5 Soient Σ un schéma de stéganographie et (V, D_V) une stéganalyse par discrimination contre Σ . On dit que Σ est ε -sûr contre (V, D_V) si et seulement si

$$D(P_{(D_V \circ V)(C)}, P_{(D_V \circ V)(S)}) \leq \varepsilon.$$

De plus si $\varepsilon = 0$ alors le système est qualifié de parfaitement sûr contre (V, D_V) .

La proposition suivante relie la sécurité contre une stéganalyse par discrimination donnée et la sécurité inconditionnelle classique.

Proposition 6.1 Soit Σ un schéma de stéganographie. Alors

$$\Sigma \text{ est } \varepsilon\text{-sûr} \implies \Sigma \text{ est } \varepsilon\text{-sûr contre } (V, D_V), \forall (V, D_V) \text{ stéganalyse par discrimination.}$$

Preuve :

D'après [99, p. 12], pour toute f fonction définie sur $\mathcal{I}$ alors

$$D(P_{f(C)}, P_{f(S)}) \leq D(P_C, P_S). \blacksquare$$

La sécurité inconditionnelle d'un algorithme de stéganographie est une donc une borne de sécurité sur l'ensemble des stéganalyses par discrimination. De plus, si un schéma de stéganographie Σ est parfaitement sûr alors cela signifie que Σ est parfaitement sûr contre toute stéganalyse par discrimination. Dans ce cas, cela implique qu'il n'existe pas de stéganalyse par discrimination efficace contre Σ .

6.2 Modèles d'attaquants

Dans ce paragraphe, nous présentons tout d'abord, le modèle de sécurité *d'indistingabilité* issu du monde de la sécurité des primitives cryptographiques. Ce modèle s'adapte aisément à l'évaluation des schémas de stéganographie et modélise de façon réaliste l'attaquant passif réel. Nous déclinons ensuite ce modèle en deux modèles d'attaquant stéganographique, *l'attaquant spécifique* et *l'attaquant universel* selon que l'algorithme attaqué est connu ou non. Du point de vue du concepteur, un schéma de stéganographie est évalué sous l'angle de sa sécurité. Pour montrer qu'un schéma est ε -sûr dans un modèle d'attaquant donné, celui-ci doit montrer que $\forall (V, D_V)$, stéganalyse par discrimination, le schéma est ε -sûr contre (V, D_V) . En revanche, en nous plaçant du côté de l'attaquant, nous cherchons à quantifier préférentiellement *l'insécurité* du schéma étudié. Dans ce cadre, il faut mettre en évidence une borne minimum sur l'insécurité, c'est-à-dire, trouver une attaque efficace contre le schéma et un distingueur adapté. Nous expliquons au paragraphe 6.2.4, comment à partir d'une attaque V et d'un distingueur D_V adapté, calculer une borne minimum sur l'insécurité du schéma. Bien évidemment, plus cette borne est grande, plus l'insécurité l'est aussi et plus la sécurité du schéma est faible.

6.2.1 Modèle d'indistingabilité

Le modèle présenté vise à prouver la sécurité d'un schéma de cryptographie sous l'angle de l'indistingabilité. Pour ce faire, il met en jeu des attaquants adaptatifs ou non. Pour un algorithme de chiffrement, l'objectif de cet attaquant est de pouvoir distinguer des messages chiffrés connaissant les messages clairs. Au paragraphe précédent, nous avons montré que la stéganalyse par discrimination se réduit à un problème de discrimination statistique, le modèle d'indistingabilité est parmi les modèles de sécurité cryptographique, celui qui est le plus à même de fournir les briques nécessaires à la modélisation de notre attaquant réel. Tout d'abord, définissons formellement notre système cryptographique à clé secrète.

Définition 6.6 [95] *Un système cryptographique à clé secrète Γ est la donnée de trois algorithmes polynomiaux, dont*

- *un algorithme probabiliste $\mathcal{K}$ de génération des clés. Il prend en entrée un paramètre de sécurité 1^k et renvoie K_s une la clé secrète.*
- *un algorithme probabiliste $\mathcal{E}$ de chiffrement. Il prend en entrée la clé K_s , un message clair $m \in \mathcal{M} \subset \{0, 1\}^*$ et renvoie un message chiffré $m' \in \{0, 1\}^*$.*
- *un algorithme déterministe $\mathcal{D}$ de déchiffrement. Il prend en entrée la clé K_s , un élément m' de $\{0, 1\}^*$ et renvoie le message clair m si m' est un chiffré valide ou $\perp$, un message d'erreur sinon.*

Dans le modèle d'indistingabilité, l'attaquant A est représenté par un couple (A_1, A_2) d'algorithmes probabilistes et polynomiaux. Chaque algorithme A_i peut faire appel à un oracle $\mathcal{O}_i$ qui lui est propre. Pour évaluer la sécurité en indistingabilité, nous simulons une expérience entre un challenger et l'attaquant. Au début de l'expérience, le challenger génère à l'aide de $\mathcal{K}$ une clé secrète K_s . Lors d'une première phase, appelée *phase d'apprentissage*, il effectue autant de requêtes qu'il le souhaite à l'oracle $\mathcal{O}_1$. À la fin de cette étape, il sauvegarde son état dans une variable s et génère deux messages m_0, m_1 de même

longueur mais distincts. Dans la deuxième phase, le challenger choisit aléatoirement un bit b et donne à l'attaquant le message m_b chiffré par la clé secrète K_s . Durant cette étape, l'attaquant peut effectuer autant de requêtes à l'oracle $\mathcal{O}_2$ qu'il le souhaite hormis une requête de déchiffrement avec le challenge c . Enfin, à partir de l'état interne, des messages m_i , du message chiffré, celui-ci doit essayer de deviner lequel des messages m_i a été chiffré. Il fait alors un pari sur b et renvoie b' .

Les oracles utilisés lors de l'expérience sont des oracles de chiffrement et/ou déchiffrement. Les oracles de chiffrement, paramétrés par K_s , prennent en entrée un message clair et renvoient soit le texte chiffré soit rien suivant la puissance de l'attaquant. Les oracles de déchiffrement, paramétrés par K_s , prennent en entrée un message chiffré et renvoient soit le texte en clair ou un message d'erreur, soit rien suivant la puissance de l'attaquant. Nous distinguons trois types d'attaquant de puissance croissante :

- *L'attaquant CPA (Chosen Plaintext Attack)*. $\mathcal{O}_1$ est un oracle de chiffrement et $\mathcal{O}_2$ ne renvoie rien. L'attaquant peut faire chiffrer à l'oracle des messages qu'il choisit.
- *L'attaquant CCA1 (Chosen Cipher Attack)*. $\mathcal{O}_1$ est un oracle de chiffrement et de déchiffrement et $\mathcal{O}_2$ est un oracle de chiffrement. L'attaquant peut faire déchiffrer à l'oracle des messages chiffrés qu'il choisit.
- *L'attaquant CCA2 (Chosen Cipher Adaptive Attack)*. Les oracles $\mathcal{O}_1$ et $\mathcal{O}_2$ sont tous les deux des oracles de chiffrement et de déchiffrement. L'attaquant peut faire à l'oracle déchiffrer des messages chiffrés qu'il choisit. De plus, il peut orienter ses requêtes de déchiffrement en fonction du challenge.

Ces définitions, ainsi que celles qui suivent, ont d'abord été introduites pour des schémas à clés publiques par S. Goldwasser et S. Micali [95], puis adaptées par M. Bellare *et al.* [25] pour des schémas à clé secrète. Bien évidemment, plus un attaquant possède de moyens (d'oracles lui donnant de l'information), plus celui-ci est puissant et moins son attaque affectera la sécurité du schéma attaqué. Soit $ATK \in \{CPA, CCA1, CCA2\}$, alors l'expérience en indistinguabilité, décrite précédemment, avec un attaquant A de type ATK contre Γ et un paramètre de sécurité 1^k , est notée $\text{Exp}_{\Gamma}^{\text{IND-ATK}}(A, k)$ et est résumée par la figure 6.1.

Remarque 6.2 :

Il est important de noter que Γ est public selon les principes de Kerckhoffs [120], ce qui implique que l'attaquant a aussi accès à Γ et peut donc lui-même générer des clés, des messages et des messages chiffrés. Néanmoins, il n'a pas accès à la clé secrète générée par le challenger. D'autre part, puisque c'est l'attaquant qui génère le couple (m_0, m_1) , il est nécessaire que le chiffrement soit randomisé, sinon il lui suffit de demander à $\mathcal{O}_1$ ($\mathcal{E}(K_s, m_0), \mathcal{E}(K_s, m_1)$) avant de donner (m_0, m_1) et de comparer avec le challenge c dans la phase 2.

Définition 6.7 *L'efficacité d'un attaquant A de type IND-ATK contre Γ est défini par son avantage $\text{Adv}_{\Gamma}^{\text{IND-ATK}}(A, k)$, tel que*

$$\text{Adv}_{\Gamma}^{\text{IND-ATK}}(A, k) = 2|\Pr(\text{Exp}_{\Gamma}^{\text{IND-ATK}}(A, k) = 1) - \frac{1}{2}|.$$

L'avantage quantifie la puissance de l'attaque, *i.e.* la proportion de fois où l'attaquant répond bien relativement à un tirage à pile ou face. Du point de vue de l'attaquant, on

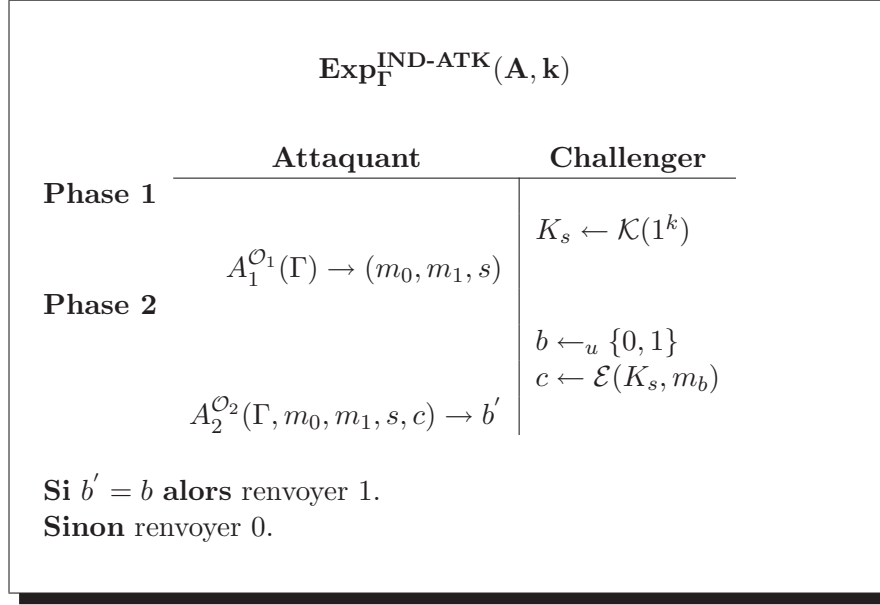


FIG. 6.1 – Expérience en indistingabilité avec un attaquant A de type ATK contre Γ

définit classiquement *l'insécurité d'un schéma*.

Définition 6.8 Nous appelons *insécurité d'un schéma cryptographique* Γ , la quantité

$$\text{InSec}_{\Gamma}^{\text{IND-ATK}}(k) = \max_{A \in \mathcal{A}} \{ \text{Adv}_{\Gamma}^{\text{IND-ATK}}(A, k) \},$$

où $\mathcal{A}$ est l'ensemble des attaquants polynomiaux.

Le modèle ainsi présenté se transpose aisément dans le monde stéganographique. Dans ce nouveau contexte, nous voulons distinguer les supports de couverture des stégo média. Dans ce cadre plus contraint, l'attaquant fournit non plus deux messages mais un message au challenger. En effet, il ne peut pas fournir au challenger le support de couverture dans lequel va être inséré ou non le message car une simple comparaison avec le challenge lui indique si le support a été modifié ou non.

Nous décrivons maintenant le modèle de sécurité adapté au monde stéganographique. L'attaquant A est représenté par un couple (A_1, A_2) d'algorithmes probabilistes et polynomiaux. Chaque algorithme A_i peut faire appel à un oracle $\mathcal{O}_i$ qui lui est propre. Au début de l'expérience, le challenger génère à l'aide de $\mathcal{K}$ une clé secrète K_s . Lors d'une première phase, appelée *phase d'apprentissage*, il effectue autant de requêtes qu'il le souhaite à l'oracle $\mathcal{O}_1$. À la fin de cette étape, il sauvegarde son état interne dans une variable s et génère un message m . Dans la deuxième phase, le challenger choisit aléatoirement un bit b , tire aléatoirement un support de couverture $C \in \mathcal{C}$. Il donne ensuite à l'attaquant C , stéganographié avec m et la clé secrète K_s si $b = 1$, ou C sinon. Durant cette étape, l'attaquant peut effectuer autant de requêtes à l'oracle $\mathcal{O}_2$ qu'il le souhaite. Enfin, à partir de l'état interne s , du message m , du challenge, celui-ci doit essayer de deviner si le challenge appartient à $\mathcal{C}$ ou $\mathcal{S}$. Il fait alors un pari sur b et renvoie b' .

L'oracle $\mathcal{O}_1$ est un oracle d'insertion et/ou d'extraction. $\mathcal{O}_2$ est exclusivement un oracle d'insertion. Les oracles d'insertion, paramétrés par K_s , prennent en entrée un message et un support de couverture et renvoient un stégo médium ou rien suivant la puissance de l'attaquant. Les oracles d'extraction, paramétrés par K_s , prennent en entrée un stégo médium et renvoient le message caché ou un message d'erreur, ou rien suivant la puissance de l'attaquant. Nous distinguons trois types d'attaquant de puissance croissante :

- *L'attaquant PA (Passive Attack)*. Les oracles $\mathcal{O}_1$ et $\mathcal{O}_2$ ne lui renvoient rien. L'attaquant est passif.
- *L'attaquant CMA (Chosen Message Attack)*. $\mathcal{O}_1$ est un oracle d'insertion et $\mathcal{O}_2$ ne renvoie rien. L'attaquant peut faire stéganographier à l'oracle des messages qu'il choisit
- *L'attaquant CHA (Chosen Hidden text Attack)*. $\mathcal{O}_1$ est un oracle d'insertion et d'extraction et $\mathcal{O}_2$ est un oracle exclusivement d'insertion. L'attaquant peut faire extraire à l'oracle des messages de stégo média qu'il choisit.

Soit $ATK \in \{PA, CMA, CHA\}$, alors l'expérience en indistingabilité, décrite précédemment, avec un attaquant de type ATK contre Σ et un paramètre de sécurité 1^k , est notée $\text{Exp}_{\Sigma}^{\text{IND-ATK}}(\mathbf{A}, k)$ et est résumée par la figure 6.2.

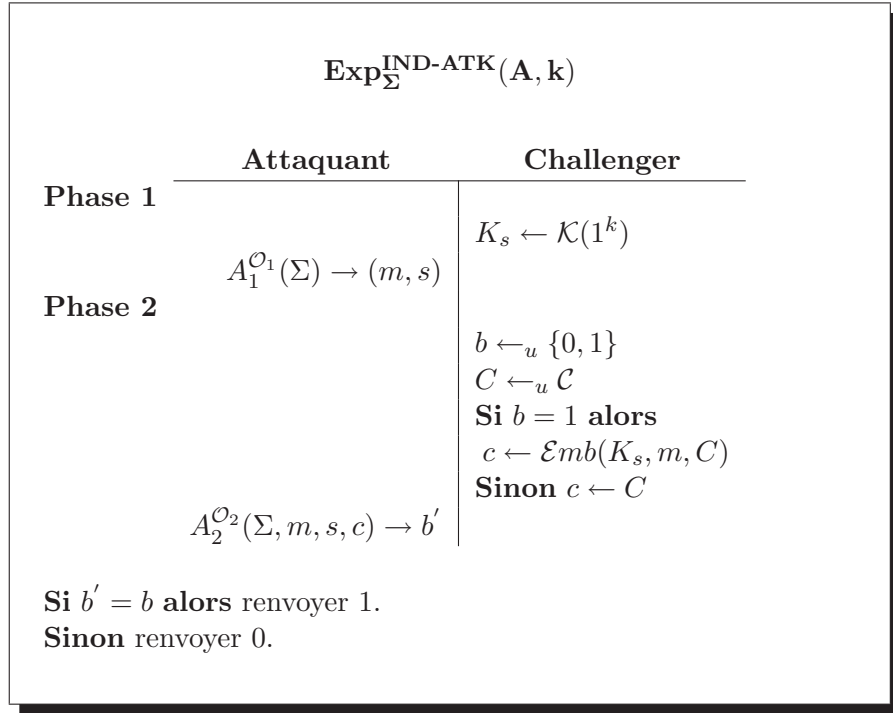


FIG. 6.2 – Expérience en indistingabilité avec un attaquant A de type ATK contre Σ

Définition 6.9 [99] *L'efficacité d'un attaquant A de type IND-ATK contre Σ est défini par son avantage $\text{Adv}_{\Sigma}^{\text{IND-ATK}}(A, k)$ tel que*

$$\text{Adv}_{\Sigma}^{\text{IND-ATK}}(A, k) = 2|\Pr(\text{Exp}_{\Sigma}^{\text{IND-ATK}}(A, k) = 1) - \frac{1}{2}|.$$

De la même façon, nous pouvons définir l'insécurité d'un schéma de stéganographie.

Définition 6.10 [99] *Nous appelons insécurité d'un schéma stéganographique Σ , la quantité*

$$InSec_{\Sigma}^{IND-ATK}(k) = \max_{A \in \mathcal{A}} \{Adv_{\Sigma}^{IND-ATK}(A, k)\},$$

où $\mathcal{A}$ est l'ensemble des attaquants polynomiaux.

L'adaptation au monde de la stéganographie s'est faite en affaiblissant un peu les modèles classiques en cryptographie. En effet, l'attaquant IND-CCA2 ne possède pas d'équivalent stéganographique du fait d'une certaine malléabilité intrinsèque aux schémas de stéganographie. On peut montrer assez facilement qu'un adversaire ayant accès à un oracle d'extraction a un avantage proche de 1. Lorsqu'il reçoit son challenge c , l'adversaire change aléatoirement un bit b_i de c et obtient c' qu'il soumet à l'oracle. Il recommence l'opération q fois. Soit c n'est pas un stégo médium, l'oracle renvoie alors toujours un message d'erreur, soit c est un stégo médium. Dans ce cas, soit b_i n'a pas été utilisé par $\mathcal{E}mb$ et l'oracle lui retourne m , soit b_i a été modifié lors de l'insertion. Deux cas sont alors à étudier : b_i est un bit critique pour l'extraction et l'oracle renvoie un message d'erreur, où b_i n'est pas critique et l'oracle renvoie un message éventuellement différent de m . Dans la majorité des schémas de stéganographie, la proportion p des bits critiques est très faible. Finalement, l'attaquant A renvoie $b' = 0$ si $\mathcal{O}_2$ lui retourne q messages d'erreur et $b' = 1$ sinon. Dans ce contexte, il échoue uniquement lorsque le challenge est un stégo médium et que $\mathcal{O}_2$ lui renvoie q messages d'erreur, *i.e.* lorsque 'il a inversé q fois un bit critique. La probabilité d'un tel évènement est donc

$$\mathcal{Pr}(A \text{ échoue}) = p^q,$$

et son avantage est donc

$$Adv_{\Gamma}^{IND-CHA}(q) = 1 - p^q.$$

Ce modèle n'est donc pas réaliste. En revanche, l'introduction d'un adversaire passif correspond bien à l'adversaire réel qui ne maîtrise ni la clé, ni le médium. Pour modéliser un attaquant réel, nous utilisons l'attaquant IND-PA sous certaines hypothèses et gardons à l'esprit la hiérarchie des modèles de sécurité.

Proposition 6.2

$$IND-CHA \implies IND-CMA \implies IND-PA.$$

Preuve :

Trivial, par construction. ■

Remarque 6.3 :

La notation « $IND-CHA \implies IND-CMA$ » signifie Σ est sûr contre un attaquant IND-CHA implique Σ est sûr contre un attaquant IND-CMA et que la réduction est efficace.

6.2.2 Attaquant spécifique

L'attaquant classique utilisé implicitement dans toutes les stéganalyses effectives de la littérature est un attaquant IND-PA contraint, car il ne choisit plus m . En effet, lors d'une phase d'apprentissage, l'attaquant génère lui-même sa clé secrète, ses messages, ses supports de couverture et essaie d'obtenir de l'information sur les distributions P_C et P_S . Il construit enfin un distingueur pour ces deux distributions. Dans la phase de challenge,

il intercepte les média échangés entre Alice et Bob et doit deviner pour chaque médium intercepté s'il est stégo ou non stégo. Dans ce contexte très précis, l'attaquant n'a accès à aucun oracle d'insertion ou d'extraction avec la clé K_s partagée par Alice et Bob. Pour un attaquant réel, $\mathcal{O}_1$ et $\mathcal{O}_2$ ne renvoient rien. De plus, lors du challenge, puisqu'il intercepte en aveugle les média, il ne peut imposer le message caché ; celui-ci est donc choisi par la challenger. Nous définissons ainsi un attaquant classique qualifié de *spécifique* car il cible un schéma de stéganographie particulier, dans le respect des principes de Kerckhoffs [120]. Cet adversaire est un attaquant IND-SSA (Specific Steganalysis Attack). Le jeu qui lui est associé est résumé dans la figure 6.3.

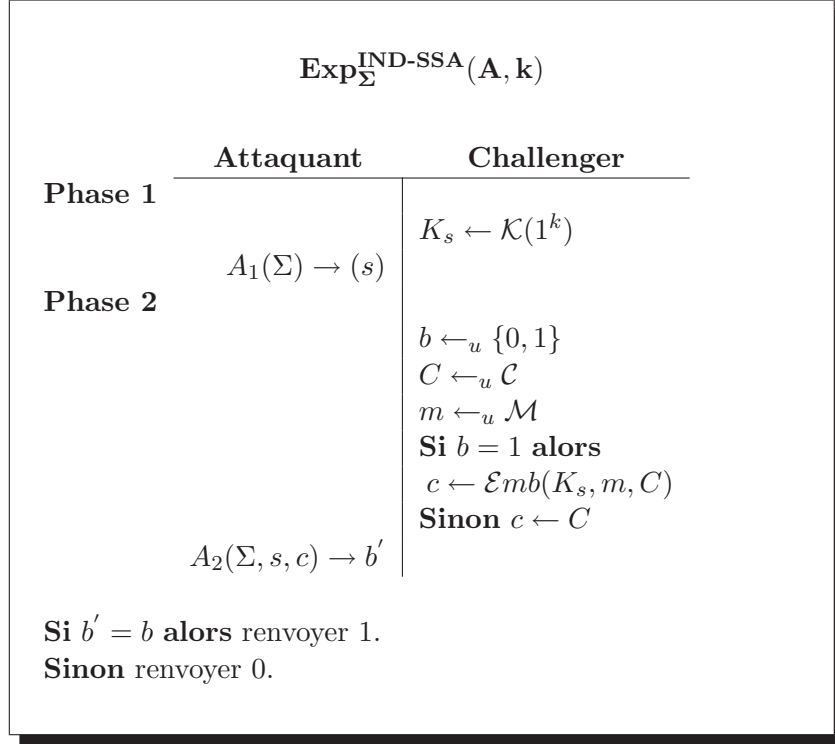


FIG. 6.3 – Expérience en indistingabilité avec un attaquant A de type IND-SSA contre Σ

Dans ce jeu, l'adversaire n'est pas restreint à une stéganalyse précise, il peut utiliser n'importe laquelle d'entre elles. En pratique, nous cherchons à évaluer l'efficacité d'une stéganalyse par discrimination donnée et à relier celle-ci à l'insécurité du schéma attaqué. Pour ce faire, nous introduisons l'attaquant $\text{IND-SSA}(V, D_V)$ qui ne peut utiliser que la stéganalyse par discrimination (V, D_V) pour apprendre de l'information lors de la phase 1 et pour répondre au challenge. Cette attaquant est noté $(A_{(V, D_V)}^1, A_{(V, D_V)}^2)$. Le jeu lui correspondant est résumé dans la figure 6.4 page suivante. Par définition du jeu, nous en déduisons la proposition suivante.

Proposition 6.3 *Soient Σ un schéma de stéganographie et (V, D_V) une stéganalyse par discrimination contre Σ . Alors Σ est ε -sûr contre un attaquant A de type $\text{IND-SSA}(V, D_V)$ si et seulement si*

$$\text{Adv}_{\Sigma}^{\text{IND-SSA}(V, D_V)}(A, k) \leq \varepsilon.$$

$\text{Exp}_{\Sigma}^{\text{IND-SSA}(V, D_V)}(A, k)$		
	Attaquant	Challenger
Phase 1		$K_s \leftarrow \mathcal{K}(1^k)$
	$A_{(V, D_V)}^1(\Sigma) \rightarrow (s)$	
Phase 2		$b \leftarrow_u \{0, 1\}$ $C \leftarrow_u \mathcal{C}$ $m \leftarrow_u \mathcal{M}$ Si $b = 1$ alors $c \leftarrow \text{Emb}(K_s, m, C)$ Sinon $c \leftarrow C$
	$A_{(V, D_V)}^2(\Sigma, s, c) \rightarrow b' = D_V(V(c))$	
	Si $b' = b$ alors renvoyer 1. Sinon renvoyer 0.	

FIG. 6.4 – Expérience en indistingabilité avec un attaquant A de type $\text{IND-SSA}(V, D_V)$ contre Σ

Preuve :

Cela revient à montrer que l'avantage de l'attaquant dans le jeu précédent est exactement $D(P_{(D_V \circ V)(\mathcal{C})}, P_{(D_V \circ V)(\mathcal{S})})$.

$$\begin{aligned}
 \text{Adv}_{\Sigma}^{\text{IND-SSA}(V, D_V)}(A, k) &= 2|\mathcal{P}_r(\text{Exp}_{\Sigma}^{\text{IND-SSA}(V, D_V)}(A, k) = 1) - \frac{1}{2}|, \\
 &= 2|\mathcal{P}_r(b' = 1|b = 1)\mathcal{P}_r(b = 1) + \mathcal{P}_r(b' = 0|b = 0)\mathcal{P}_r(b = 0) - \frac{1}{2}|, \\
 &= |\mathcal{P}_r(b' = 1|b = 1) + \mathcal{P}_r(b' = 0|b = 0) - 1|, \\
 &= |P_{(D_V \circ V)(\mathcal{S})}(1) - P_{(D_V \circ V)(\mathcal{C})}(1)|.
 \end{aligned}$$

De même,

$$|\mathcal{P}_r(\text{Exp}_{\Sigma}^{\text{IND-SSA}(V, D_V)}(A, k) = 1) - \frac{1}{2}| = |P_{(D_V \circ V)(\mathcal{S})}(0) - P_{(D_V \circ V)(\mathcal{C})}(0)|.$$

D'où

$$\begin{aligned}
 \text{Adv}_{\Sigma}^{\text{IND-SSA}(V, D_V)}(A, k) &= \frac{1}{2}(|P_{(D_V \circ V)(\mathcal{S})}(1) - P_{(D_V \circ V)(\mathcal{C})}(1)| \\
 &\quad + |P_{(D_V \circ V)(\mathcal{S})}(0) - P_{(D_V \circ V)(\mathcal{C})}(0)|). \blacksquare
 \end{aligned}$$

De plus, par construction nous avons

Proposition 6.4

$$\text{IND-CHA} \implies \text{IND-CMA} \implies \text{IND-PA} \implies \text{IND-SSA} \implies \text{IND-SSA}(V, D_V), \quad \forall (V, D_V).$$

6.2.3 Attaquant universel

En stéganographie, on considère souvent un attaquant encore plus faible et ne respectant pas les principes de Kerckhoffs [120]. Bien évidemment, cet attaquant n'est jamais pris en compte lors de la phase de conception car beaucoup trop faible. Néanmoins, cet adversaire simule des attaques par discrimination qui ne dépendent pas de l'algorithme et qui mettent en évidence des faiblesses intrinsèques à une classe de schémas stéganographiques. L'objectif est de mesurer le caractère générique d'une attaque. Pour ce faire, l'attaquant dispose de tous les schémas de stéganographie Σ_i , sauf un, Σ_j . Durant la phase d'apprentissage, il essaie d'obtenir de l'information sur les distributions P_C et $P_{S_{i \neq j}}$, en espérant que P_{S_i} ne soit pas trop différente d'au moins un P_{S_j} . Le challenge est alors réalisé avec Σ_j . L'attaque par discrimination que nous présentons au paragraphe 8.2 possède ce caractère générique. Cela signifie en d'autres termes, qu'en entraînant un distingueur sur des schémas de stéganographie connus, nous sommes potentiellement capables de détecter des schémas inconnus. Cet attaquant IND-USA (Universal Steganalysis Attack) est qualifié d'*universel*. Le jeu lui correspondant est résumé dans la figure 6.5. Dans ce jeu, l'adversaire n'est pas

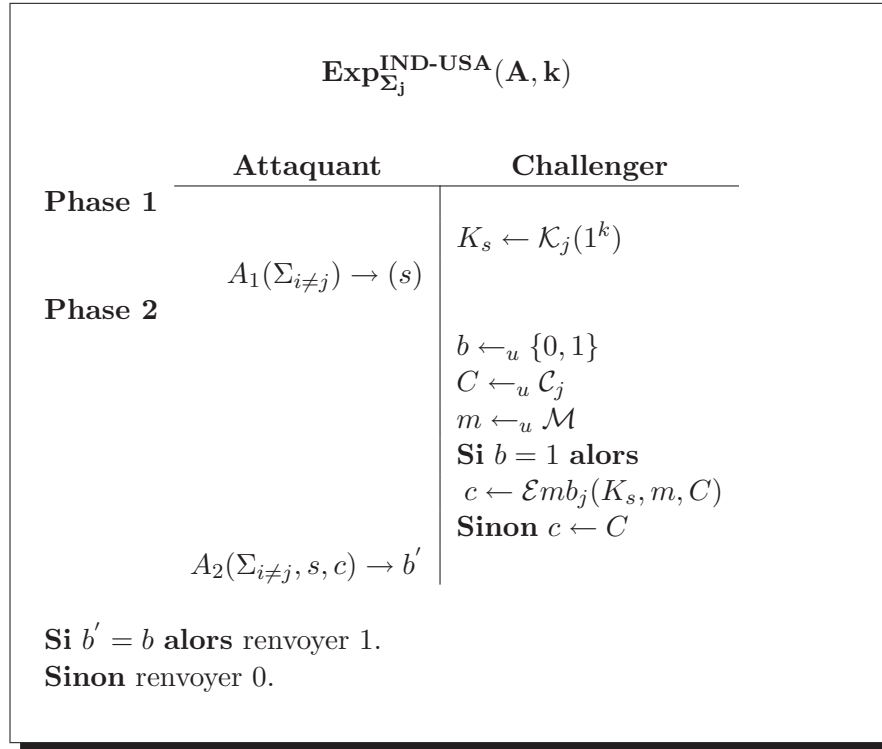


FIG. 6.5 – Expérience en indistingabilité avec un attaquant A de type IND-USA contre Σ_j

restreint à une stéganalyse précise, il peut utiliser n'importe laquelle d'entre elles. En pratique, nous cherchons à évaluer l'efficacité d'une stéganalyse par discrimination donnée et à relier celle-ci à l'insécurité du schéma attaqué. Pour ce faire, nous introduisons l'attaquant $\text{IND-USA}(V, D_V)$ qui ne peut utiliser que la stéganalyse par discrimination (V, D_D) pour apprendre de l'information lors de la phase 1 et pour répondre au challenge. Cet attaquant est noté $(A_{(V, D_V)}^1, A_{(V, D_V)}^2)$. Le jeu lui correspondant est résumé dans la figure 6.6 page suivante. Par définition du jeu, nous en déduisons la proposition suivante.

$\text{Exp}_{\Sigma_j}^{\text{IND-USA}(V, D_V)}(A, k)$		
	Attaquant	Challenger
Phase 1		$K_s \leftarrow \mathcal{K}_j(1^k)$
	$A_{(V, D_V)}^1(\Sigma_{i \neq j}) \rightarrow (s)$	
Phase 2		$b \leftarrow_u \{0, 1\}$ $C \leftarrow_u \mathcal{C}_j$ $m \leftarrow_u \mathcal{M}$ Si $b = 1$ alors $c \leftarrow \text{Emb}_j(K_s, m, C)$ Sinon $c \leftarrow C$
	$A_{(V, D_V)}^2(\Sigma_{i \neq j}, s, c) \rightarrow b' = D_V(V(c))$	
	Si $b' = b$ alors renvoyer 1. Sinon renvoyer 0.	

FIG. 6.6 – Expérience en indistingabilité avec un attaquant A de type $\text{IND-USA}(V, D_V)$ contre Σ_j

Proposition 6.5 Soient Σ un schéma de stéganographie et (V, D_V) une stéganalyse par discrimination contre Σ . Alors Σ est ε -sûr contre un attaquant A de type $\text{IND-USA}(V, D_V)$ si et seulement si

$$\text{Adv}_{\Sigma}^{\text{IND-USA}(V, D_V)}(A, k) \leq \varepsilon.$$

Preuve :

Le calcul est identique à la preuve de la proposition 6.3 page 183.

Définition 6.11 Dans les conditions précédentes, si

$$\text{Adv}_{\Sigma}^{\text{IND-USA}(V, D_V)}(A, k) > 0,$$

alors (V, D_V) est une stéganalyse par discrimination universelle.

De plus, par construction nous avons

Proposition 6.6

$$\text{IND-PA} \implies \text{IND-SSA} \implies \text{IND-USA} \implies \text{IND-USA}(V, D_V), \quad \forall (V, D_V).$$

6.2.4 Évaluation de la sécurité contre une attaque

Au paragraphe précédent, nous avons modélisé les attaquants réels, c'est-à-dire ceux utilisés en pratique dans les articles de stéganalyse et formalisé les modèles de sécurité définis implicitement. Pour ce faire, nous sommes partis des modèles classiques de sécurité définis par C. Cachin [39], S. Katzenbeisser [115] et N. Hopper [99] et nous avons successivement affaibli notre attaquant jusqu'à obtenir les modèles les plus proches des attaquants considérés dans les stéganalyses de la littérature. Cela nous a permis de positionner nos modèles dans la hiérarchie usuelle et de montrer que toute stéganalyse efficace dans nos modèles implique que le schéma de stéganographie n'est pas sûr dans tous les modèles usuels. Dans ce paragraphe, nous relierons la sécurité, par symétrie l'insécurité, d'un schéma de stéganographie aux mesures usuelles de performance des stéganalyses effectives.

Nous avons montré au paragraphe précédent qu'un schéma de stéganalyse par discrimination Σ est ε -sûr contre un attaquant A de type $\text{IND-ATK}(V, D_V)$, $\text{ATK} \in \{\text{SSA}, \text{USA}\}$ si et seulement si

$$\text{Adv}_{\Sigma}^{\text{IND-ATK}(V, D_V)}(A, k) \leq \varepsilon.$$

D'autre part, par définition,

$$\text{InSec}_{\Sigma}^{\text{IND-ATK}}(k) = \max_{A \in \mathcal{A}} \{\text{Adv}_{\Sigma}^{\text{IND-ATK}}(A, k)\},$$

où $\mathcal{A}$ est l'ensemble des attaquants polynomiaux. Nous exprimons ensuite l'avantage $\text{Adv}_{\Sigma}^{\text{IND-ATK}}(A, k)$ en fonction des critères de performance pour une stéganalyse effective. L'adversaire A reçoit le challenge c et doit faire une hypothèse sur c parmi H_1 , « c est un stégo médium » et H_2 , « c est un support de couverture ». Nous rappelons que $\mathcal{C}$ est l'ensemble des supports de couverture et $\mathcal{S}$ l'ensemble des stégo média. Nous sommes dans un cas classique de test à deux hypothèses. Le lecteur intéressé par ce pan du domaine des statistiques peut se référer à [164]. Les définitions classiques suivantes en sont issues. Pour évaluer la « réussite » de l'attaquant, nous différencions deux configurations :

1. c est un support de couverture,
2. c est un stégo médium.

Dans la configuration 1, nous calculons tout d'abord $\Pr(A \text{ répond } H_1 \mid c \in \mathcal{C}) = \mathcal{P}_{fp}$, la probabilité de faux positifs, aussi appelée probabilité de fausse alarme ou encore erreur de 1^{ère} espèce. Nous en déduisons alors $\Pr(A \text{ répond } H_2 \mid c \in \mathcal{C}) = \mathcal{P}_{vn}$, la probabilité de vrais négatifs, aussi appelée spécificité. Bien évidemment, nous avons

$$\mathcal{P}_{fp} + \mathcal{P}_{vn} = 1.$$

Dans la configuration 2, nous calculons tout d'abord $\Pr(A \text{ répond } H_2 \mid c \in \mathcal{S}) = \mathcal{P}_{fn}$, la probabilité de faux négatifs, aussi appelée probabilité de non détection ou encore erreur de 2^{ème} espèce. Nous en déduisons alors $\Pr(A \text{ répond } H_1 \mid c \in \mathcal{S}) = \mathcal{P}_{vp}$, la probabilité de vrais positifs, aussi appelée sensibilité. Bien évidemment, nous avons

$$\mathcal{P}_{fn} + \mathcal{P}_{vp} = 1.$$

En règle générale, ces probabilités sont représentées dans une *matrice de confusion* (cf. tableau 6.1 page suivante).

		Médium	
		$c \in \mathcal{S}$	$c \in \mathcal{C}$
Détecteur	stégo	$\mathcal{P}_{vp}$	$\mathcal{P}_{fp}$
	non stégo	$\mathcal{P}_{fn}$	$\mathcal{P}_{vn}$

TAB. 6.1 – Matrice de confusion d'un détecteur stéganographique

De plus, si le détecteur reçoit n_s challenges c , média stéganographiés et n_c challenges c , supports de couverture, alors la *probabilité de succès*, $\mathcal{P}_{suc}$, du détecteur est donnée par

$$\mathcal{P}_{suc} = \frac{1}{n_c + n_s} (n_s \mathcal{P}_{vp} + n_c \mathcal{P}_{vn}).$$

Traditionnellement, le détecteur est évalué avec autant de challenges de $\mathcal{C}$ que de $\mathcal{S}$, *i.e.* $n_c = n_s$. Dans ce cas particulier, la probabilité de succès est donc la moyenne de la sensibilité et de la spécificité. Dans le cadre de nos stéganalyses, nous nous plaçons dans ce cas particulier.

Les performances des détecteurs dépendent d'un seuil qui impacte les probabilités de détection, de faux positifs et de faux négatifs. Traditionnellement, on trace les courbes ROC (*Receiver Operating Characteristic*) représentant la probabilité de vrais positifs en fonction de la probabilité de faux positifs. Elles mettent ainsi en évidence le compromis à faire lors du choix du seuil. Généralement, on se fixe une probabilité maximum de faux positifs, puis on choisit le seuil en conséquence. Intuitivement, plus la courbe ROC est proche du coin supérieur gauche et meilleur est le détecteur. En stéganographie, on limite au maximum les faux positifs car ils sont plus coûteux que les faux négatifs. En effet, une fois qu'un stégo médium est détecté, on essaie d'extraire l'information cachée. D'autre part, le compromis illustré par la figure 4.13 page 141, montre que les performances des détecteurs stéganographiques dépendent de la quantité d'information dissimulée. Pour prendre en compte ce paramètre, les stéganalystes fixent le taux stéganographique et évaluent la courbe ROC pour ce taux. Pour un détecteur donné, nous obtenons ainsi une famille de courbes ROC, comme l'illustre la figure 6.7 page ci-contre. Cette démarche est équivalente à attaquer avec la même stéganalyse les schémas de stéganographie Σ_r , correspondant au schéma Σ dissimulant au taux stéganographique constant r .

Soit une stéganalyse par discrimination (V, D_V) , dont le détecteur possède une probabilité de faux positifs α et de faux négatifs β dans le modèle IND-ATK(V, D_V). $P_{(D_V \circ V)(\mathcal{C})}$ et $P_{(D_V \circ V)(\mathcal{S})}$ sont des distributions uniformes de paramètres respectifs $(1 - \beta)$ et $(1 - \alpha)$. D'après la preuve de la proposition 6.3 page 183,

$$Adv_{\Sigma}^{\text{IND-ATK}}(A, k) = |1 - (\alpha + \beta)|.$$

On en déduit alors la proposition suivante.

Proposition 6.7 *Soient Σ un schéma de stéganographie et une stéganalyse par discrimination (V, D_V) , dont le détecteur possède une probabilité de faux positifs α et de faux négatifs β dans le modèle IND-ATK(V, D_V) $\in \{SSA, USA\}$. Alors*

$$|1 - (\alpha + \beta)| \leq InSec_{\Sigma}^{\text{IND-ATK}}(k).$$

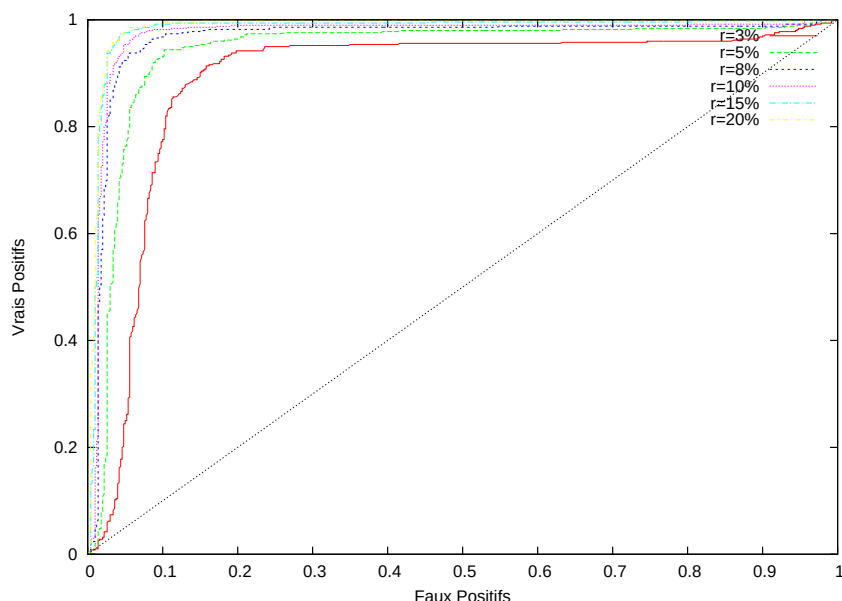


FIG. 6.7 – Famille de courbes ROC d'un détecteur

D'après les propositions 6.4 page 184 et 6.6 page 186, l'étude de détecteurs effectifs nous permet de mettre en évidence une borne sur l'insécurité d'un schéma de stéganographie dans les modèles classiques. Nous avons donc relié les modèles classiques de sécurité avec la sécurité pratique évaluée dans le cadre de stéganalyses effectives. De plus, les performances des stéganalyses effectives nous donnent une borne sur la sécurité des schémas de stéganographie. Toute stéganalyse par discrimination (V, D_V) effective contre un schéma dans le modèle $\text{IND-ATK}(V, D_V)$, implique que ce schéma n'est pas sûr dans les modèles classiques.

6.3 Distingueur de Fisher

Dans cette partie, nous détaillons la conception d'un distingueur à partir d'une attaque par discrimination V donnée. Nous nous sommes focalisés sur des distingueurs linéaires issus de l'analyse discriminante à 2 catégories construits à partir de la fonction de Fisher et de la règle de décision de Mahalanobis-Fisher. Ces distingueurs sont extrêmement simples à mettre en œuvre et se comprennent facilement par des méthodes géométriques. De plus, les distingueurs que nous avons conçus pour nos stéganalyses, présentées aux chapitres 7 et 8, offrent des performances excellentes. Ce paragraphe est essentiellement tiré du chapitre 18 *analyse discriminante* de *Probabilité, Analyse des Données et Statistiques* [164].

6.3.1 L'analyse discriminante

Étant données k catégories distinctes d'observations, l'objectif de l'analyse discriminante est d'affecter à une nouvelle observation l'une des catégories. Dans le cadre de la stéganographie, $k = 2$ et les catégories sont la catégorie des média de couverture et la catégorie des stégo média. Une observation est la donnée d'un médium. L'analyse se décompose en deux étapes. La première consiste à répartir n observations dont on connaît les catégories. La seconde, consiste à affecter une nouvelle observation à une catégorie.

L'affectation est alors équivalente à une hypothèse sur l'appartenance de l'observation à une catégorie. Plus précisément, à chaque observation sont associées p variables explicatives V_i , qui permettent de décrire une observation. Ces variables sont des variables aléatoires et correspondent à une attaque par discrimination de p coordonnées définie au paragraphe 6.2 page 176 pour la stéganographie. Les deux étapes peuvent se résumer de la manière suivante :

- *l'étape de description* : l'objectif est de trouver la combinaison linéaire des variables explicatives qui sépare le mieux les catégories,
- *l'étape de décision* : l'objectif est d'affecter à une nouvelle observation dont on connaît la valeur des variables, l'une des catégories.

Chaque catégorie est représentée par un nuage E_i de $\mathbb{R}^p$, composé n_i individus $(e_i^j)_{j=1\dots n_i}$ de coordonnées $(V_1(e_i^j), \dots, V_p(e_i^j))$, où $V_k(e_i^j)$ est la valeur de la variable explicative V_k pour l'observation e_i^j . Nous considérons sans perte de généralité que chaque e_i est unique. Chaque nuage E_i possède un centre de gravité noté g_i .

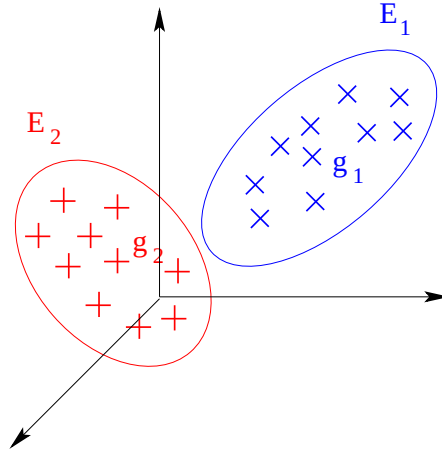


FIG. 6.8 – Représentation des nuages pour $p = 3$ et $k = 2$

Par définition,

$$g_i = \frac{1}{n_i} \sum_{j=1}^{n_i} e_i^j \quad \text{pour } e_i^j \in E_i.$$

Nous notons g le barycentre de tous les individus,

$$g = \frac{1}{n} \sum_{j=1}^k n_j g_j,$$

V_m la matrice de variance de tous les individus et V_m^i la matrice de variance de E_i ,

$$V_m^i = \frac{1}{n_i} \sum_{e_i^j \in E_i} (e_i^j - g_i)(e_i^j - g_i)^t.$$

D'autre part, nous appelons matrice de *variance interclasse*, la matrice B définie par

$$B = \frac{1}{n} \sum_{j=1}^k n_j (g_j - g)(g_j - g)^t.$$

Cette matrice représente la variance des k centres de gravité g_j affectés des poids n_j . De même nous définissons la matrice de *variance intraclasse* W comme la matrice moyenne des V_m^j ,

$$W = \frac{1}{n} \sum_{j=1}^k n_j V_m^j.$$

Les matrices de variance vérifient $V_m = W + B$, ce qui généralise la relation classique définissant la variance totale comme la somme de la moyenne des variances et de la variance des moyennes.

Nous nous intéressons tout d'abord à la première étape de l'analyse discriminante, l'*analyse factorielle discriminante* (AFD). Elle consiste à rechercher de nouvelles variables, ou *variables discriminantes*, qui séparent le mieux en projection les k catégories. Chaque axe correspond à une direction dans $\mathbb{R}^p$. La figure 6.9 illustre clairement que le choix de l'axe de projection est déterminant dans la séparation de la projection des catégories. En effet, l'axe 1 est plus discriminant que l'axe 2.

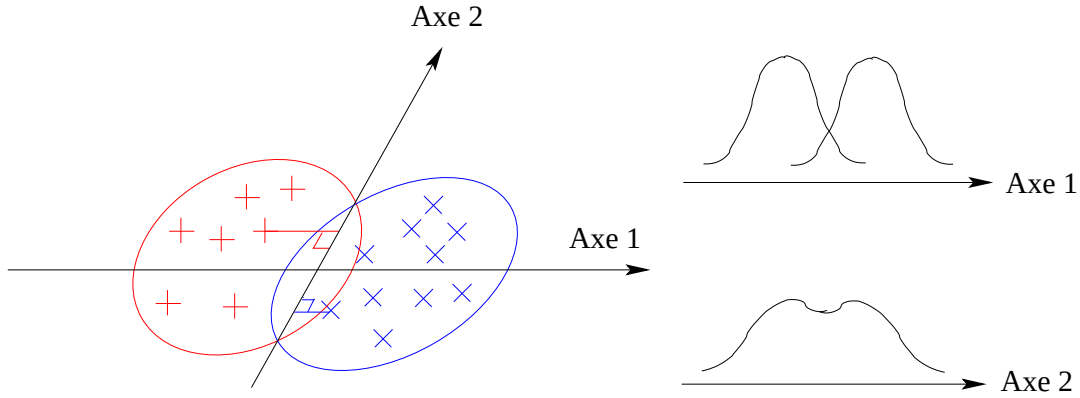


FIG. 6.9 – Pouvoir discriminant des axes de projection

Soit a l'axe discriminant que nous recherchons et $u = Ma$ le *facteur discriminant* associé, où M est une métrique définie sur $\mathbb{R}^p$. En projetant les nuages sur a , les k centres de gravité doivent être aussi éloignés que possible et les autres individus aussi proches de la projection de leur centre de gravité que possible. Ces critères sont équivalents à maximiser l'inertie des g_j projetés sur a . La matrice d'inertie des g_i est donnée par MBM et l'inertie des g_i projetés sur a est $a^t(MBM)a$ en considérant que a est M -normé à 1.

D'autre part, pour que la projection d'une catégorie reste bien groupée autour de son centre, il faut que la variance des catégories projetées soit faible, c'est-à-dire $a^t(MV_m^j M)a$ soit faible pour $j = 1 \dots k$. Cela revient à minimiser la moyenne

$$\sum_{j=1}^k \left(\frac{n}{n_j} \right) a^t(MV_m^j M)a = a^t(MWM)a.$$

Or, $V_m = B + W$ implique

$$a^t(MV_m M)a = a^t(MBM)a + a^t(MWM)a,$$

soit

$$1 - \frac{a^t(MWM)a}{a^t(MV_mM)a} = \frac{a^t(MBM)a}{a^t(MV_mM)a}.$$

Pour répondre aux deux critères nous prenons

$$a = \underset{a}{\operatorname{Argmax}} \left(\frac{a^t(MBM)a}{a^t(MV_mM)a} \right).$$

À l'aide du quotient de Rayleigh, nous pouvons montrer, [164, p. 406], que ce maximum est atteint pour a vecteur propre de $(MV_mM)^{-1}MBM$ associé à sa plus grande valeur propre λ ,

$$M^{-1}V_m^{-1}BMa = \lambda a.$$

Le facteur discriminant $u = Ma$ vérifie donc

$$V_m^{-1}Bu = \lambda u.$$

À chaque valeur propre non nulle, il existe un axe discriminant, le nombre d'axes discriminants est donc au plus $k - 1$, pour des variables explicatives non linéairement liées [164, p. 407].

6.3.2 Discrimination particulière restreinte à deux classes

Dans le cas d'une analyse à deux classes, il n'y a qu'une seule variable discriminante ($k - 1 = 1$). On en déduit que le seul axe discriminant est l'axe passant par les centres de gravité,

$$a = (g_1 - g_2).$$

Deux choix sont possibles pour la métrique M . Traditionnellement, on choisit V_m^{-1} ou W^{-1} . Si $M = W^{-1}$, la métrique est appelée *métrique de Mahalanobis* et

$$u = W^{-1}(g_1 - g_2).$$

$W^{-1}(g_1 - g_2)$ est alors appelé *fonction de Fisher*. Avec deux catégories, le calcul de l'unique vecteur propre se simplifie. Il suffit de remarquer que B peut s'écrire sous la forme

$$B = \frac{n_1 n_2}{n^2} (g_1 - g_2)(g_1 - g_2)^t,$$

et que

$$u = M^{-1}(g_1 - g_2) \tag{6.1}$$

est vecteur propre de $M^{-1}B$. L'axe de facteur discriminant u ainsi obtenu, est l'axe qui sépare le mieux par projection les deux catégories. Nous nous intéressons maintenant au classement d'un nouvel individu e dont on connaît les $V_i(e)$ mais pas la catégorie à laquelle il appartient. Nous allons donc émettre une hypothèse sur son appartenance à l'une des deux classes. Pour cela, nous projetons e sur a et regardons s'il est plus près de g_1 ou g_2 . Si $M = W^{-1}$ ou V_m^{-1} la règle de *Mahalanobis-Fisher* consiste à affecter à e la catégorie 1 si

$$e^t M^{-1}(g_1 - g_2) > \frac{1}{2}(g_1 + g_2)^t M^{-1}(g_1 - g_2), \tag{6.2}$$

et la catégorie 2 sinon. Dans le cas de *variables centrées* (g est pris pour origine du repère) telles que $n_1 = n_2$ alors $g_1 + g_2 = 0$.

En conclusion, à partir d'un ensemble d'apprentissage dont on connaît les valeurs des V_j ainsi que la catégorie pour chaque individu, nous sommes en mesure de calculer g ainsi que le facteur discriminant u . Le facteur discriminant nous donne la direction d'un axe orthogonal à un hyperplan qui sépare le mieux les deux catégories. Ensuite nous comparons $e.u$ à une valeur de seuil T défini par la relation (6.2) page ci-contre, ce qui revient à positionner e par rapport à l'hyperplan d'équation $x^t.u = T$. S'il est « au-dessus », c'est-à-dire si

$$(e - g)^t.u > T, \quad (6.3)$$

alors e est décidé de catégorie 1 sinon il est affecté à la catégorie 2.

Remarque 6.4 :

Dans le reste de l'étude nous manipulons des variables centrées, le seuil T est alors calculé en fonction de l'équation de l'hyperplan affine dans le repère centrée en g . Si les variables ne sont pas centrées, le seuil T' à considérer est tel que $T' = T + g^t.u$.

Lors des étapes 1 des jeux associés aux attaquants IND-SSA et IND-USA, l'adversaire

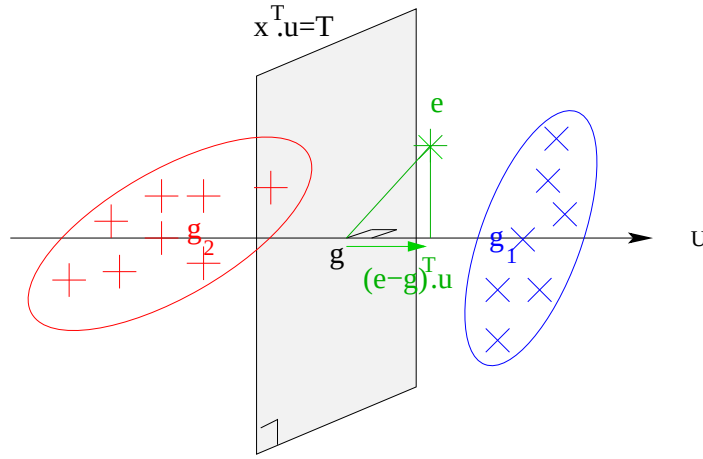


FIG. 6.10 – Étape de décision pour un nouvel individu

construit son ensemble d'apprentissage à partir de son attaque par discrimination (les valeurs V_j). À la fin des étapes 1, il stocke dans son état interne le facteur discriminant u , le barycentre g et le seuil T déterminé à partir des variables centrées et des probabilités d'erreur de détection fixées. Lors des étapes 2, il applique la règle de décision 6.3 en posant $e = c$. Il renvoie b' en fonction de la catégorie supposée.

6.4 Conclusion

Dans ce chapitre, nous nous sommes inspirés des définitions et des modèles de sécurité classiques proposés par C. Cachin [39], S. Katzenbeisser et F. Petitcolas [115] et N. Hopper [99], afin de définir formellement les notions d'attaque par discrimination et de stéganalyse par discrimination. Cette formalisation a pour objectif de représenter le plus fidèlement possible le comportement de l'attaquant réel qui procède en deux étapes distinctes. Dans un premier temps, il doit mettre en évidence des mesures qui permettent de discriminer les stégo média des supports de couverture ; c'est-à-dire concevoir une attaque par discrimination. Dans un second temps, à partir de ces mesures, il doit élaborer une règle

de décision qui associe à tout médium la classe « stégo » ou la classe « non stégo », c'est-à-dire un distingueur. Dans ce contexte, nous définissons la stéganalyse par discrimination comme la donnée d'une attaque par discrimination et d'un distingueur compatible. Pour une attaque donnée, nous nous autorisons alors à utiliser différents distingueurs pour améliorer les performances d'une stéganalyse. Dans la pratique, une stéganalyse donnée peut, par exemple, être définie avec une discrimination linéaire puis améliorée en utilisant des Machines à Support de Vecteurs (SVM), [135]. Nous proposons ensuite deux nouveaux modèles de sécurité et les jeux associés afin de modéliser l'attaquant réel spécifique, d'une part, et l'attaquant réel universel, d'autre part, tels qu'ils sont implicitement définis dans la littérature. Ces modèles sont ensuite reliés aux modèles classiques et, par construction, tout schéma de stéganographie qui n'est pas sûr dans les modèles proposés, ne le sont pas dans non plus dans les modèles classiques. Enfin, nous exprimons l'insécurité d'un schéma de stéganographie dans ces nouveaux modèles, en fonction des performances des stéganalyses effectives décrites dans la littérature. Les attaquants ainsi modélisés sont ceux que nous considérons par la suite. Pour terminer ce chapitre, nous décrivons brièvement l'analyse discriminante de Fisher, à la base des distingueurs que nous avons calibrés pour nos stéganalyses.

Stéganalyse dans le domaine spatial

*« Celui qui est habile en stratégie ne gagne
ni reconnaissance pour sa ruse ni récompense
pour son courage. »*

Sagesse militaire chinoise

Sommaire

7.1	Stéganalyse RS	196
7.2	Relative immunité de MBPIS	198
7.3	Stéganalyse RS locale	201
7.4	Résultats expérimentaux	203
7.4.1	Phase d'apprentissage, stéganalyse MBPIS-RS locale	204
7.4.2	Phase de challenge, stéganalyse MBPIS-RS locale	205
7.4.3	Phase d'apprentissage, stéganalyse LSB-RS classique	206
7.4.4	Phase de challenge, stéganalyse LSB-RS classique	207
7.4.5	Comparaison des détecteurs	208
7.5	Conclusion	209

CE chapitre est consacré à la stéganalyse de l’algorithme de stéganographie *Multi Bit Plane Image Steganography* (MBPIS) proposé par B.C. Nguyen *et al.* à IWDW’06 (International Workshop on Digital Watermarking) [147]. Cet algorithme, détaillé au paragraphe 5.1.2, dissimule le message dans des plans de bits de Codage de Gray Canonique (CGC) d’une image non compressée en excluant ses zones homogènes. Les auteurs espèrent ainsi se prémunir d’une stéganalyse classique proposée par J. Fridrich *et al.*, la stéganalyse RS [77]. Dans ce chapitre, nous montrons clairement que la stratégie de conception adoptée n’est pas efficace. En effet, nous définissons un détecteur dont les performances rendent caduque l’utilisation d’un tel algorithme de stéganographie. Par ailleurs, nous concevons aussi une attaque par discrimination à partir des coefficients de la stéganalyse RS et remettons ainsi en cause l’immunité avancée par les concepteurs de MBPIS. Nous proposons alors d’adapter la stéganalyse RS afin de tirer parti au maximum des contre-mesures introduites par B.C. Nguyen *et al.*. Plus spécifique, notre détecteur est alors bien plus efficace que le détecteur naturel issu de la stéganalyse RS.

Dans le paragraphe 7.1, nous détaillons la stéganalyse RS afin de bien comprendre la stratégie adoptée par les auteurs de MBPIS. Nous expliquons ensuite, au paragraphe 7.2, pourquoi l'immunité annoncée contre la stéganalyse RS semble peu probante. Nous développons au paragraphe 7.3 une adaptation de cette stéganalyse afin de prendre en compte les contre-mesures introduites par les auteurs de MBPIS. Enfin, au paragraphe 7.4, nous évaluons dans un premier temps les performances de notre technique, puis nous montrons expérimentalement que MBPIS n'est pas immune à la stéganalyse RS en calibrant un détecteur efficace défini à partir des coefficients de la stéganalyse RS. Enfin, nous illustrons le gain obtenu par notre adaptation à l'aide d'un tableau comparatif des performances des deux détecteurs. Le résultat de ces travaux sera présenté à IWDW'07 [18].

7.1 Stéganalyse RS

La stéganalyse RS a été introduite par J. Fridrich, M. Goljan et R. Du [77] pour détecter l'utilisation de la stéganographie LSB. Cette technique est dédiée au domaine spatial mais peut s'étendre, d'après les auteurs, à tous types de stéganographie LSB. De plus, cette analyse permet d'effectuer une estimation assez précise de la longueur du message dissimulé. Dans ce paragraphe, nous détaillons cette technique afin de bien comprendre pourquoi l'algorithme de B.C. Nguyen *et al.*, présenté au paragraphe 5.1.2 semble à première vue immune à cette analyse. Nous détaillons la stéganalyse RS pour des images en niveaux de gris, mais celle-ci se généralise tout naturellement, composante par composante, aux images en couleurs.

L'image en niveaux de gris non compressée est tout d'abord divisée en groupes non recouvrants de n pixels consécutifs. En pratique, $n = 4$. Considérons maintenant un tel groupe $G = (x_1, \dots, x_n)$. Nous évaluons l'homogénéité de G en appliquant la fonction de discrimination f définie par

$$f(x_1, \dots, x_n) = \sum_{i=1}^{n-1} |x_{i+1} - x_i|. \quad (7.1)$$

Plus le groupe est homogène, plus les valeurs des pixels voisins sont proches et donc plus $f(x_1, \dots, x_n)$ est faible.

Nous avons aussi besoin d'introduire une fonction inversible F , appelée *fonction d'inversion* et définie sur $[0, 255]^n$. F est définie à partir de

$$F_1 : 0 \leftrightarrow 1, 2 \leftrightarrow 3, \dots, 254 \leftrightarrow 255, \quad (7.2)$$

$$F_{-1} : -1 \leftrightarrow 0, 1 \leftrightarrow 2, \dots, 255 \leftrightarrow 256, \quad (7.3)$$

et F_0 l'identité de $[0, 255]$. Enfin, F paramétrée par M est déterminée de la façon suivante.

$$\begin{aligned} F_M : [0, 255]^n &\longrightarrow [-1, 256]^n \\ x = (x_1, \dots, x_n) &\longrightarrow F_M(x) = (F_{M(1)}(x_1), \dots, F_{M(n)}(x_n)), \end{aligned}$$

où M est appelé *masque*. Ce masque, est un vecteur de n coordonnées à valeurs dans -1, 0 et 1. Par définition,

$$F_{-1}(x) = F_1(x + 1) - 1, \quad \forall x. \quad (7.4)$$

Pour chaque groupe G , nous évaluons $F(G)$ et le classons parmi trois catégories de pixels, R, S et U tels que

$$\begin{aligned} G \in R &\Leftrightarrow f(F(G)) > f(G), \\ G \in S &\Leftrightarrow f(F(G)) < f(G), \\ G \in U &\Leftrightarrow f(F(G)) = f(G). \end{aligned}$$

Nous calculons ensuite la proportion R_M de groupes G dans R pour le masque M , la proportion S_M de groupes G dans S pour M . De la même manière, nous calculons R_{-M} et S_{-M} , où $-M$ est le *masque négatif* tel que $[-M](i) = -M(i)$ pour tout i dans $[1, n]$. L'hypothèse centrale de la stéganalyse RS est la suivante.

$$R_M \cong R_{-M} \text{ et } S_M \cong S_{-M}, \quad (7.5)$$

pour une image naturelle. Cette hypothèse se vérifie en pratique. Le lecteur pourra se référer à l'article originel de J. Fridrich [77]. Les auteurs ont mis en évidence que les coefficients R_M et S_M sont une fonction quadratique de la longueur p du message dissimulé, tandis que les coefficients R_{-M} et S_{-M} peuvent être approximatés par une fonction linéaire de p . La stéganalyse RS consiste alors à obtenir trois points pour les courbes quadratiques et deux pour les linéaires, afin d'effectuer une interpolation et d'en déduire une estimation de p .

Tout d'abord, notons que la valeur des LSB peut être vue comme une variable aléatoire suivant une loi uniforme de paramètre 0.5. L'insertion de type LSB force la valeur du LSB d'un pixel à celle du message. Nous considérons le message comme étant chiffré, la valeur prise par chacun de ses bits est aussi une variable aléatoire suivant une loi uniforme de paramètre 0.5. L'insertion LSB d'un message chiffré de longueur l change donc $l/2$ LSB en moyenne. Nous raisonnons maintenant sur la longueur relative du message, ou taux stéganographique, $p = l/n$, où n est le nombre de LSB disponibles.

D'autre part, les courbes R_M et R_{-M} , ainsi que les courbes S_M et S_{-M} s'intersectent à l'origine, d'après la relation (7.5). L'origine correspond à $p = 0$, c'est-à-dire à une image non stéganographiée. Cette observation découlant de la relation (7.5) peut se justifier de façon heuristique en utilisant la relation (7.4) page ci-contre. Soit un groupe G de pixels, M un masque, alors

$$\begin{aligned} f(F_M(G)) &= \sum_{i=1}^{n-1} |F_{M(i+1)}(x_{i+1}) - F_{M(i)}(x_i)| \\ f(F_{-M}(G)) &= \sum_{i=1}^{n-1} |F_{-M(i+1)}(x_{i+1}) - F_{-M(i)}(x_i)|, \end{aligned}$$

avec

$$f(F_{-M}(G)) = \sum_{i=1}^{n-1} |F_{M(i+1)}(x_{i+1} + 1) - F_{M(i)}(x_i + 1)|. \quad (7.6)$$

Il en résulte qu'appliquer f à $F_{-M}(\cdot)$ est équivalent à appliquer f à $F_M(\cdot)$ pour un média dont les valeurs des pixels sont augmentées de 1. La mesure des coefficients RS du média permet alors de tracer les quatre courbes en $p/2$.

De plus, si nous changeons la valeur de tous les LSB, alors un ratio de $(1 - p/2)$ bits du support de couverture seront inversés. En mesurant les coefficients RS après avoir inversé tous les LSB, nous évaluons les courbes en $(1 - p/2)$. De même, en changeant la valeurs des LSB par des bits aléatoires, 1 LSB sur 2 en moyenne sera inversé. En mesurant les coefficients RS après avoir remplacé tous les LSB par une valeur aléatoire, nous obtenons les points des courbes en $1/2$. Enfin, à partir des mesures en $p/2$, $1/2$ et $(1 - p/2)$, nous pouvons interpoler les courbes des coefficients R_M , R_{-M} , S_M et S_{-M} . En pratique, ces courbes sont à peu près similaires d'une image à l'autre. En particulier, les courbes R_M et S_M se coupent en 0.5. La figure 7.1 est une représentation de ces courbes, aussi appelée *diagramme RS*.

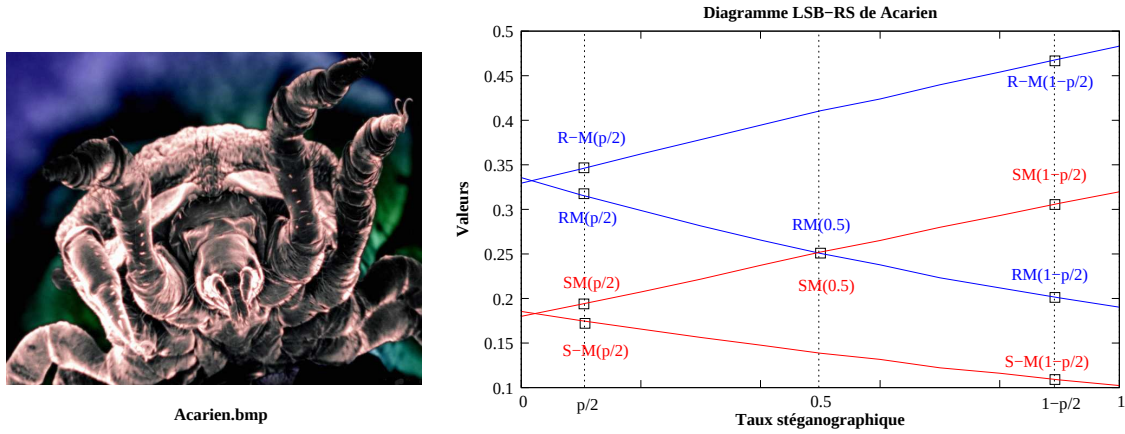


FIG. 7.1 – Diagramme RS de image *acarrien.bmp* de taux stéganographique p avec $M = [0110]$

Au delà de la mise en évidence de mesures qui diffèrent selon que l'image est stéganographiée ou non, la connaissance des lois d'évolution de celles-ci permet une estimation de p . Pour cela, nous effectuons tout d'abord un changement de repère en prenant le $p/2$ comme origine et $(1 - p/2)$ pour 1. Dans ce nouveau repère, le point d'intersection entre R_M et S_M a pour abscisse x vérifiant l'équation quadratique

$$2(d_1 + d_0)x^2 + (d_{-0} - d_{-1} - d_1 - 3d_0)x + d_0 - d_{-0} = 0,$$

où

$$\begin{aligned} d_0 &= R_M(p/2) - S_M(p/2), \\ d_{-0} &= R_{-M}(p/2) - S_{-M}(p/2), \\ d_1 &= R_M(1 - p/2) - S_M(1 - p/2), \\ d_{-1} &= R_{-M}(1 - p/2) - S_{-M}(1 - p/2). \end{aligned}$$

Si x est la plus petite racine positive alors p est donné par

$$p = \frac{x}{x - \frac{1}{2}}.$$

7.2 Relative immunité de MBPIS

La stéganalyse RS repose essentiellement sur la variation de la mesure d'homogénéité définie par la relation (7.1) page 196. Une zone est dite homogène si tous les pixels de

la zone ont tous une valeur très proche. Pour une telle zone, les sous-fenêtres G qui la composent auront donc une mesure d'homogénéité $f(G)$ proche de 0. De la même façon, $f(F_M(G))$ et $f(F_{-M}(G))$ sont proches de 0 pour des masques M bien choisis, d'après la relation (7.6) page 197. La dissimulation d'information dans ces zones « casse » l'homogénéité et nous n'avons plus $f(G)$ proche de 0. Puisque $f(\cdot)$ est positive, ces valeurs ne peuvent qu'augmenter. D'autre part, les actions de $F_M(\cdot)$ et $F_{-M}(\cdot)$ ont tendance à amplifier l'écart entre $f(F_M(G))$, $f(F_{-M}(G))$ et $f(G)$ car ils cassent un peu plus l'homogénéité de G . Au contraire, dans des zones non homogènes, $f(F_M(G))$, $f(F_{-M}(G))$ et $f(G)$ sont en général très différents et la dissimulation d'information dans ces zones a beaucoup moins d'impact sur la variation de ces valeurs. En effet, l'introduction d'un changement sur le LSB d'un pixel x_i dans une fenêtre G peut soit faire augmenter ou baisser $|x_{i+1} - x_i|$ avec probabilités respectives 0.5. Dans des zones non homogènes les LSB peuvent être considérés comme des variables aléatoires binaires qui suivent une loi uniforme de paramètre 0.5. Cet argument reste valable lorsque l'on applique les masques M et $-M$. La stratégie adoptée par les auteurs de MBPIS, spécifié au paragraphe 5.1.2, est donc de dissimuler l'information dans des zones non homogènes. De plus, ceux-ci utilisent le Codage de Gray Canonique (CGC) afin de maîtriser les variations des zones homogènes au cours du processus d'insertion. En effet, l'insertion dans un plan de bits i n'affecte que les zones homogènes de plan de bits i et inférieur. De plus, comme l'insertion s'effectue dans l'ordre décroissant des plans de bits, les zones homogènes sont recalculées à chaque changement de plan. Les auteurs espèrent ainsi n'avoir qu'un impact limité sur la variation des coefficients RS.

Le calcul des zones homogènes au cours du processus d'insertion rend la capacité et la détermination de ces zones inconnues *a priori*. Expérimentalement, la capacité de MBPIS dépasse rarement les 40 %, ce qui ne permet pas de tracer un diagramme RS complet en utilisant MBPIS au lieu de l'insertion LSB pour stéganographier. Nous parlons de diagramme *Algo-RS* lorsque l'algorithme de stéganographie *Algo* est utilisé pour tracer le diagramme RS. De plus, même si il l'on effectue un changement d'échelle pour ramener la capacité sur $[0, 1]$, les diagrammes MBPIS-RS ne ressemblent plus au diagramme LSB-RS comme l'illustre la figure 7.2. Dans ces conditions, MBPIS semble de prime abord immune

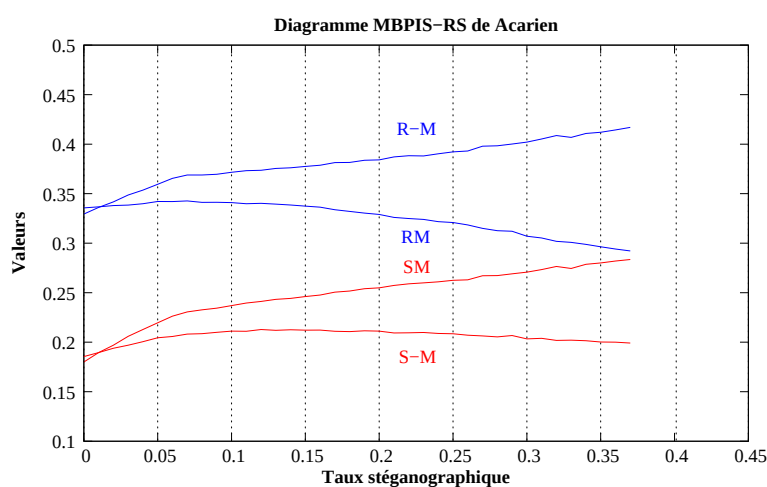


FIG. 7.2 – Diagramme MBPIS-RS de l'image *acarrien.bmp*

contre une application classique de la stéganalyse RS. En effet, le diagramme RS étant incomplet, il est impossible d'effectuer une estimation de la longueur du message par cette méthode. Néanmoins, la figure 7.2 page précédente met clairement en évidence que les coefficients MBPIS-RS varient de manière au moins linéaire en fonction de la quantité d'information insérée. La spécification d'un détecteur performant est alors envisageable.

D'autre part, la variation des coefficients MBPIS-RS permet de douter fortement de l'immunité annoncée par les auteurs de MBPIS. Nous avons donc tracé les diagrammes LSB-RS pour l'image *acarrien.bmp* stéganographiée respectivement à 5, 10, 15 et 20 %. Les diagrammes LSB-RS, représentés sur la figure 7.3, permettent de conclure que l'application directe de la stéganalyse RS en utilisant l'insertion LSB au lieu de MBPIS permet non seulement de détecter MBPIS, mais aussi d'en estimer classiquement la longueur du message. Les écarts à l'origine entre les courbes, R_M et R_{-M} d'une part, S_M et S_{-M} d'autre part, augmentent en fonction de la quantité d'information dissimulée et correspondent aux mesures effectuées en $p/2$.

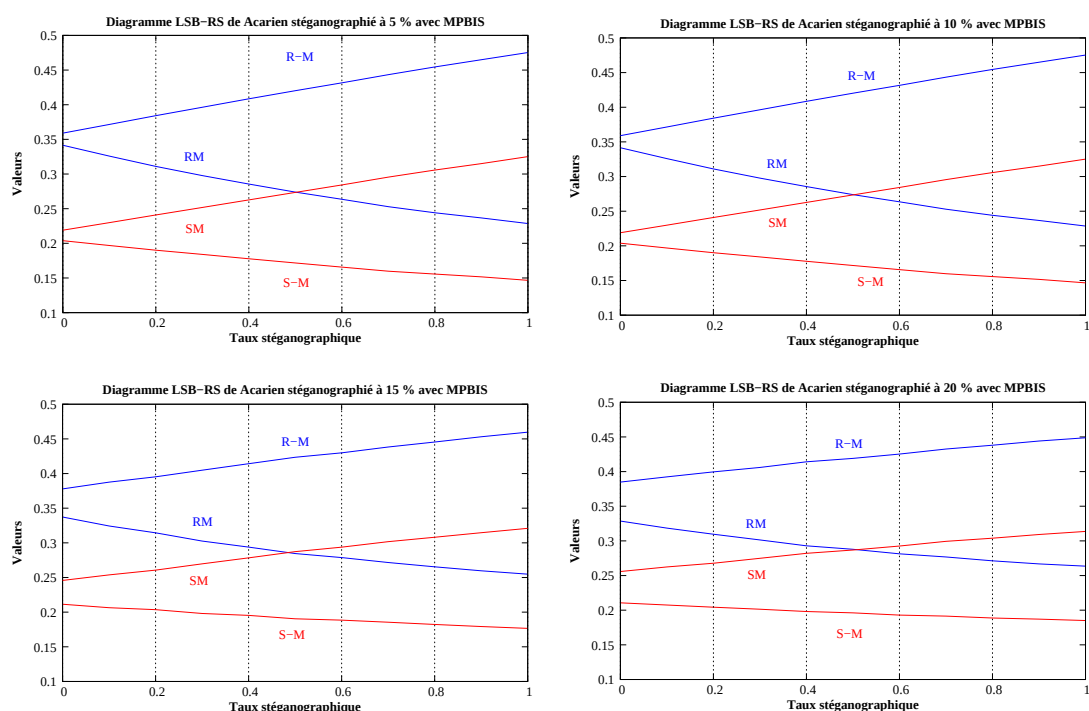


FIG. 7.3 – Diagrammes LSB-RS pour l'image *acarrien.bmp* stéganographiée à 5, 10, 15 et 20 %

L'immunité annoncée par les auteurs de MBPIS n'étant pas avérée, nous nous sommes attachés à élaborer un détecteur efficace. L'estimation de la longueur du message par la stéganalyse RS permet d'en déduire un détecteur trivial. En revanche, la contre-mesure utilisée par MBPIS rend l'estimation moins fiable que pour l'insertion LSB classique. Nous proposons de tirer parti de ces contre-mesures pour concevoir un détecteur plus efficace. La perte de fiabilité dans l'estimation de la longueur est due au fait que les changements s'effectuent dans les zones non homogènes de l'image, celles qui sont le moins impactantes pour la stéganalyse RS classique. Beaucoup de pixels ne sont pas altérés et les changements

les moins impactants sont ainsi « noyés dans la masse ». Pour illustrer notre propos, raisonnons en termes de probabilités. La stéganalyse RS classique calcule la proportion de fenêtres G telles que $G \in R$ ainsi que la proportion des fenêtres $G \in S$, c'est à dire $\mathcal{P}r(G \in R)$ et $\mathcal{P}r(G \in S)$. Regardons maintenant comment évoluent ces probabilités lorsque nous les évaluons sur une image stéganographiée avec MBPIS. Considérons $\mathcal{H}$, l'ensemble des zones homogènes et notons $\mathcal{E}$ l'algorithme d'insertion de MBPIS. Nous calculons alors les variations relatives, ΔR_M , ΔS_M , ΔU_M , des coefficients RS lorsque $\mathcal{E}$ est appliqué.

$$\Delta R_m = \left| \frac{\mathcal{P}r(\mathcal{E}(G) \in R) - \mathcal{P}r(G \in R)}{\mathcal{P}r(G \in R)} \right| = \left| \frac{\mathcal{P}r(\mathcal{E}(G) \in R)}{\mathcal{P}r(G \in R)} - 1 \right|. \quad (7.7)$$

ΔS_M et ΔU_M sont définis de la même façon. De plus, nous avons $\frac{\mathcal{P}r(\mathcal{E}(G) \in R)}{\mathcal{P}r(G \in R)}$

$$\begin{aligned} &= \frac{\mathcal{P}r(G \in \mathcal{H}) \mathcal{P}r(\mathcal{E}(G) \in R | G \in \mathcal{H}) + \mathcal{P}r(G \notin \mathcal{H}) \mathcal{P}r(\mathcal{E}(G) \in R | G \notin \mathcal{H})}{\mathcal{P}r(G \in R)} , \\ &= \frac{\mathcal{P}r(G \in \mathcal{H}) \mathcal{P}r(G \in R | G \in \mathcal{H}) + \mathcal{P}r(G \notin \mathcal{H}) \mathcal{P}r(\mathcal{E}(G) \in R | G \notin \mathcal{H})}{\mathcal{P}r(G \in \mathcal{H}) \mathcal{P}r(G \in R | G \in \mathcal{H}) + \mathcal{P}r(G \notin \mathcal{H}) \mathcal{P}r(G \in R | G \notin \mathcal{H})} , \end{aligned}$$

car MBPIS laissent les zones homogènes invariantes, *i.e.* $\forall G \in \mathcal{H}, \mathcal{E}(G) = G$.

$$\frac{\mathcal{P}r(\mathcal{E}(G) \in R)}{\mathcal{P}r(G \in R)} = \frac{1 + \frac{\mathcal{P}r(G \notin \mathcal{H}) \mathcal{P}r(\mathcal{E}(G) \in R | G \notin \mathcal{H})}{\mathcal{P}r(G \in \mathcal{H}) \mathcal{P}r(G \in R | G \in \mathcal{H})}}{1 + \frac{\mathcal{P}r(G \notin \mathcal{H}) \mathcal{P}r(G \in R | G \notin \mathcal{H})}{\mathcal{P}r(G \in \mathcal{H}) \mathcal{P}r(G \in R | G \in \mathcal{H})}} = \frac{1 + \varepsilon}{1 + \varepsilon'} ,$$

avec ε et ε' négligeables sous l'hypothèse $\mathcal{P}r(G \in \mathcal{H}) \gg \mathcal{P}r(G \notin \mathcal{H})$. L'hypothèse est néanmoins valable pour la majorité des images naturelles dans la mesure où les zones non homogènes correspondent aux contours et sont donc beaucoup moins nombreuses que les zones homogènes. De plus, $\mathcal{P}r(G \in R | G \in \mathcal{H})$ et $\mathcal{P}r(G \notin R | G \in \mathcal{H})$ restent constantes lors de l'insertion et ne dépendent donc pas de $\mathcal{E}$. Nous concluons que ΔR_M est proche de zéro. Le raisonnement identique peut être effectué pour ΔS_M et ΔU_M . Les contre-mesures introduites par MBPIS rendent les coefficients RS quasi-constants lors de la stéganalyse RS classique. Nous proposons alors de se focaliser uniquement sur les zones non homogènes utilisées par MBPIS, en adaptant la stéganalyse RS en une stéganalyse RS locale, ce qui est équivalent à mesurer les variations relatives $\Delta' R_M$ et $\Delta' S_M$ définies par

$$\Delta' R_M = \frac{\mathcal{P}r(\mathcal{E}(G) \in R | G \notin \mathcal{H})}{\mathcal{P}r(G \in R | G \notin \mathcal{H})} , \quad (7.8)$$

$$\Delta' S_M = \frac{\mathcal{P}r(\mathcal{E}(G) \in S | G \notin \mathcal{H})}{\mathcal{P}r(G \in S | G \notin \mathcal{H})} , \quad (7.9)$$

qui ne sont plus négligeables et donc plus faciles à détecter que les variations des coefficients RS classiques.

7.3 Stéganalyse RS locale

L'application directe de la stéganalyse RS semble difficile dans la mesure où les zones homogènes ne sont pas utilisées par MBPIS. De plus, elles évoluent au cours du processus et dépendent explicitement du message inséré. De plus, l'insertion s'effectue dans l'ordre

décroissant des plans de bits ; les coefficients LSB ne sont donc pas toujours altérés. Dans ce paragraphe, nous présentons une adaptation de la stéganalyse MBPIS-RS qui se focalise uniquement sur les zones utiles à MBPIS. Les diagrammes MBPIS-RS locaux ne permettent pas d'estimer la longueur du message mais néanmoins de concevoir un détecteur totalement adapté aux critères de conception de MBPIS. Nous décrivons notre stéganalyse sur des images en niveaux de gris codés sur 8 bits et détaillons ensuite la généralisation naturelle aux images en couleurs.

Tout d'abord, conformément à l'étape 2 de l'algorithme d'extraction de MBPIS, paragraphe 5.8 page 150, nous calculons toutes les zones non homogènes de taille $m \times n$. Ces zones sont représentées par des groupes $m \times n$ pixels. Notons $\hat{G} = \begin{pmatrix} x_{11} & \dots & x_{1n} \\ \vdots & & \vdots \\ x_{m1} & \dots & x_{mn} \end{pmatrix}$ un tel groupe. Nous mesurons l'homogénéité de $\hat{G}$ en appliquant la fonction de discrimination $\hat{f}$ définie par

$$\begin{aligned} \hat{f}(\hat{G}) &= \hat{f} \left(\begin{pmatrix} x_{11} & \dots & x_{1n} \\ \vdots & & \vdots \\ x_{m1} & \dots & x_{mn} \end{pmatrix} \right) \\ &= \sum_{i=1}^m \sum_{j=1}^{n-1} |x_{i,j+1} - x_{i,j}| + \sum_{j=1}^n \sum_{i=1}^{m-1} |x_{i+1,j} - x_{i,j}|. \end{aligned}$$

Nous définissons aussi $\hat{F}$, la fonction d'inversion par

$$\begin{aligned} \hat{F} : \mathcal{M}_{m \times n}([0, 255]) &\longrightarrow \mathcal{M}_{m \times n}([-1, 256]) \\ \hat{G} = \begin{pmatrix} x_{11} & \dots & x_{1n} \\ \vdots & & \vdots \\ x_{m1} & \dots & x_{mn} \end{pmatrix} &\longrightarrow \hat{F}(\hat{G}), \end{aligned}$$

où

$$\hat{F}(\hat{G}) = \begin{pmatrix} F_{M(1,1)}(x_{11}) & \dots & F_{M(1,n)}(x_{1n}) \\ \vdots & & \vdots \\ F_{M(m,1)}(x_{m1}) & \dots & F_{M(m,n)}(x_{mn}) \end{pmatrix},$$

avec M est une matrice de taille $m \times n$ à valeurs dans $\{-1, 0, 1\}$ et F_i définie par les relations (7.2) page 196 et (7.3) page 196. Pour chaque groupe $\hat{G}$, nous calculons $\hat{F}(\hat{G})$ et le classons parmi trois catégories de groupes de pixels, R , S et U tels que

$$\begin{aligned} \hat{G} \in R &\Leftrightarrow \hat{f}(\hat{F}(\hat{G})) > \hat{f}(\hat{G}), \\ \hat{G} \in S &\Leftrightarrow \hat{f}(\hat{F}(\hat{G})) < \hat{f}(\hat{G}), \\ \hat{G} \in U &\Leftrightarrow \hat{f}(\hat{F}(\hat{G})) = \hat{f}(\hat{G}). \end{aligned}$$

Nous évaluons ensuite R_M , la proportion de groupes R pour le masque M , S_M et la proportion de groupes S pour M . De la même manière, nous calculons R_{-M} et S_{-M} , où $-M$ est le masque négatif, tel que $[-M](i, j) = -M(i, j)$ pour tout i, j dans $[1, m] \times [1, n]$. Nous utilisons le masque M défini par $M(i, j) = 1$ si et seulement si $(i + j)$ est pair et $M(i, j) = 0$ sinon. Comme nous ne sommes pas capables de construire le diagramme RS complet, mais seulement une petite partie, nous ne pouvons pas interpoler les courbes

RS et donc estimer le taux stéganographique de cette façon. Nous avons seulement une vision partielle du diagramme RS. La technique mise au point dans [77] implique que les différences entre R_M et R_{-M} d'une part, S_M et S_{-M} d'autre part, augmentent localement avec le taux stéganographique, comme l'illustre la figure 7.1 page 198. Pour tirer profit de cette observation, nous introduisons les différences relatives

$$\begin{aligned}\mathcal{Q}_R &= \frac{R_M - R_{-M}}{R_M} = 1 - \frac{R_{-M}}{R_M}, \\ \mathcal{Q}_S &= \frac{S_M - S_{-M}}{S_M} = 1 - \frac{S_{-M}}{S_M}.\end{aligned}$$

Finalement, à l'image I nous associons un vecteur statistique $V(I)$ de six coordonnées, tel que

$$I \longrightarrow V(I) = (R_M, S_M, R_{-M}, S_{-M}, \mathcal{Q}_R, \mathcal{Q}_S). \quad (7.10)$$

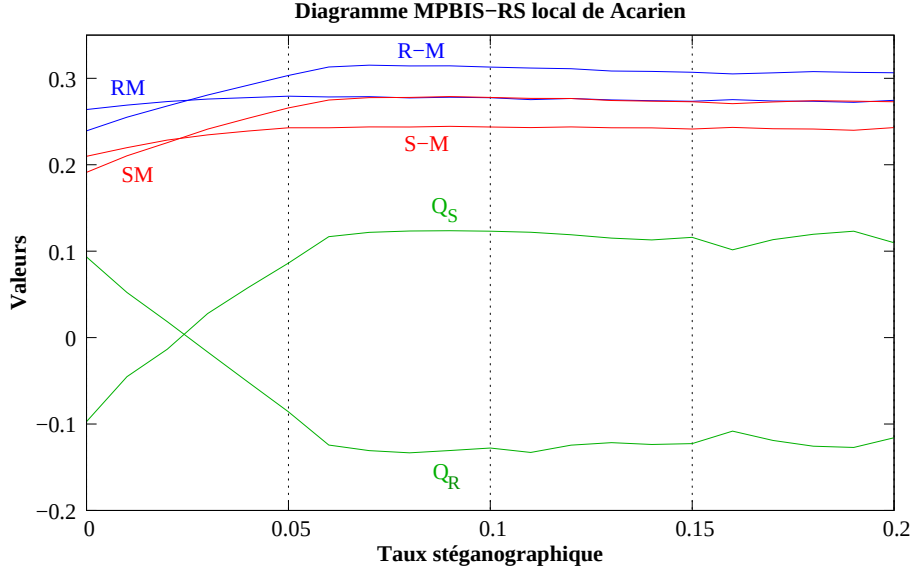
Ce vecteur statistique est le point central de notre stéganalyse. Comme nous ne prenons pas en compte les zones homogènes, ces mesures ne sont plus négligeables et peuvent alors être exploitées pour spécifier un détecteur hautement discriminant. Pour les images non stéganographiées nous avons

$$R_M \cong R_{-M}, S_M \cong S_{-M} \text{ and } \mathcal{Q}_R \cong \mathcal{Q}_S \cong 0.$$

Au contraire, $R_{-M}, S_M, |\mathcal{Q}_R|, |\mathcal{Q}_S|$ augmentent et R_M, S_{-M} diminuent avec le taux stéganographique. Nous mesurons cette déviation statistique en utilisant un discriminant de Fisher, détaillé au paragraphe 6.3. De plus, comme le plan de bits i_{max} est le plus altéré par MBPIS, nous n'évaluons le vecteur $V(I)$ que sur les zones non homogènes de ce plan de bits. D'autre part, nous pouvons généraliser cette technique aux images en couleurs en prenant la moyenne des coefficients RS sur les trois composantes RVB.

7.4 Résultats expérimentaux

Dans ce paragraphe, nous présentons les performances de notre détecteur. Pour ce faire, nous avons choisi les mêmes paramètres que ceux proposés par les auteurs de MBPIS. Nous avons testé notre stéganalyse en prenant des zones non homogènes de dimensions 2×2 , un plan de bit maximal d'ordre $i_{max} = 4$ et un seuil $t = 0$. Pour illustrer le choix de notre vecteur statistique, nous avons tracé le diagramme MBPIS-RS local de l'image *acarrien.bmp* en la stéganographiant avec des messages aléatoires pour un taux stéganographique de 0 à 20 %. La figure 7.4 page suivante représente ce diagramme. Comme nous l'avons détaillé au chapitre 6, la conception d'un détecteur stéganographique s'effectue en deux temps. Lors d'une première étape, nous entraînons le détecteur sur un ensemble maîtrisé constitué de supports de couverture et de stégo média. Lors de cette étape, nous avons choisi d'entraîner un discriminant de Fisher, détaillé au paragraphe 6.3. Enfin, nous évaluons les performances du détecteur en lui soumettant des challenges qui consistent à discriminer des supports de couverture et des images stéganographiées avec MBPIS. Son efficacité est alors représentée par des courbes *Receiver Operating Characteristic* (ROC). À titre de comparaison, nous avons calibré un détecteur en utilisant la stéganalyse RS classique, c'est-à-dire en effectuant la mesure des coefficients RS sur l'ensemble des zones homogènes et non homogènes. Nous nous sommes placés dans les mêmes conditions en utilisant exactement les mêmes ensembles d'images.

FIG. 7.4 – Diagramme MBPIS-RS local de *acarrien.bmp*

7.4.1 Phase d'apprentissage, stéganalyse MBPIS-RS locale

Pour entraîner notre détecteur, nous avons constitué un ensemble aléatoire $\mathcal{C}$ de média de couverture et un ensemble aléatoire $\mathcal{S}$ d'images stéganographiées avec MBPIS. Pour des raisons de simplification des calculs, et sans perte de généralité, ces deux ensembles possèdent le même cardinal.

Nous calculons tout d'abord $\mathcal{V}_c = \{V(I) | I \in \mathcal{C}\}$ comme défini par la relation (7.10) page précédente et $\mathcal{V}_s = \{V(I) | I \in \mathcal{S}\}$ qui sont des sous-ensembles de $\mathbb{R}^6$. Nous notons g_c , respectivement g_s , le barycentre de $\mathcal{V}_c$, respectivement $\mathcal{V}_s$, et g celui de g_c , g_s . Le point g est pris comme origine du repère. Nous calculons V_c , V_s , les matrices intraclasse W , interclasse B et des variances V_m , comme définies au paragraphe 6.3.

L'axe discriminant (g_c, g_s) , est défini par le vecteur discriminant

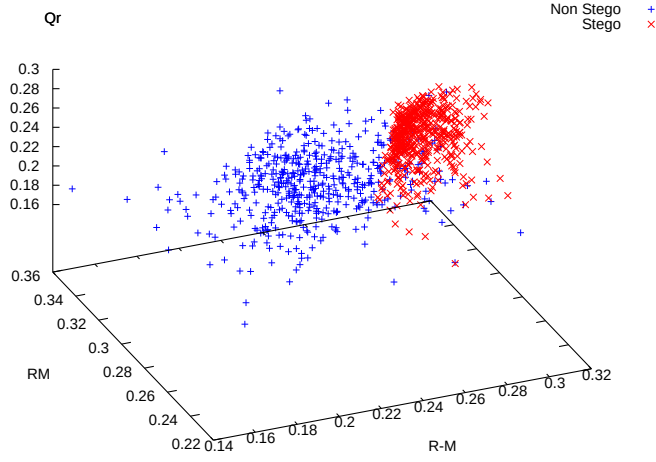
$$u = W^{-1}(g_c - g_s).$$

Le vecteur u est orthogonal à l'hyperplan qui discrimine le mieux $\mathcal{C}$ et $\mathcal{S}$. Enfin, le seuil T positionne ce plan dans l'espace affine. Une image I , représentée par un point p est considérée comme appartenant à $\mathcal{C}$, si $d^2(p, g_c) > d^2(p, g_s)$, où d est la distance basée sur la métrique W^{-1} , ou de façon équivalente

$$p \cdot u = p W^{-1}(g_c - g_s) > T,$$

Durant la phase d'apprentissage, nous avons choisi 500 images au format BMP dans la base *www.worldprint.com* et 500 autres dans la même base, que nous avons stéganographiées avec MBPIS et un taux stéganographique variant de 3 à 25 %. La figure 7.5 représente la projection spatiale de l'ensemble d'apprentissage dans le repère (R_M, R_{-M}, Q_R) .

Enfin, nous avons calibré notre discriminant de Fisher afin d'obtenir les meilleurs taux de détection sur notre ensemble d'apprentissage. Nous obtenons ainsi le vecteur discriminant

FIG. 7.5 – Projection spatiale de $\mathcal{C}$ et $\mathcal{S}$ sur (R_M, R_{-M}, Q_R)

et le barycentre associé

$$u = \begin{pmatrix} -8.327397E + 01 \\ -6.080035E + 02 \\ +1.224144E + 02 \\ +4.573029E + 02 \\ -5.263161E + 01 \\ +2.452215E + 01 \end{pmatrix} \quad g = \begin{pmatrix} +2.910226E-01 \\ +2.525373E-01 \\ +2.976101E-01 \\ +2.450667E-01 \\ -2.384120E-02 \\ +1.886609E-02 \end{pmatrix}.$$

7.4.2 Phase de challenge, stéganalyse MBPIS-RS locale

Nous avons choisi aléatoirement 500 images au format BMP et 500 autres stéganographiées avec MBPIS pour un taux stéganographique variant de 3 à 25 %. Nous les avons ensuite soumis à notre classificateur. Les performances du détecteur sont représentées par les courbes ROC de la figure 7.6. Il apparaît que notre détecteur est capable de détecter l'usage de MBPIS avec une probabilité de détection supérieure à 0.62 et une probabilité de fausse alarme inférieure à 0.1 dans le pire cas, *i.e.* avec un taux stéganographique de 0.03. Les taux de détection sont résumés dans le tableau 7.1.

taux stégo.	$\mathcal{P}_{fp}$	$\mathcal{P}_{fn}$	$\mathcal{P}_{suc}$	Seuil
0.03	0.1	0.622	0.639	-1.01333
0.03	0.15	0.356	0.747	-0.02872
0.05	0.048	0.232	0.86	-2.41151
0.05	0.1	0.066	0.917	-1.01040
0.08	0.01	0.27856	0.85586	-5.48520
0.08	0.05	0.00802	0.97097	-2.40874
0.1	0.01	0.14629	0.92192	-5.48557
0.1	0.02	0.03808	0.97097	-4.30133

TAB. 7.1 – Taux de détection de MBPIS, stéganalyse MBPIS-RS locale, variables centrées

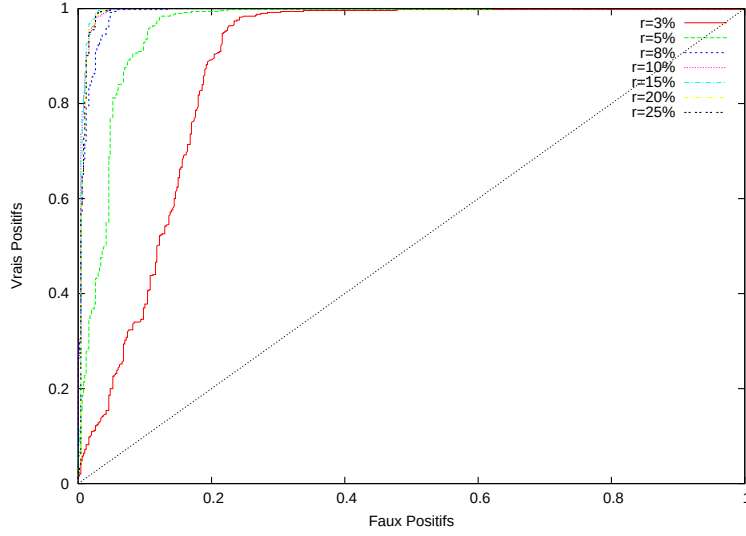


FIG. 7.6 – Courbes ROC du détecteur MBPIS, stéganalyse RS locale

7.4.3 Phase d'apprentissage, stéganalyse LSB-RS classique

Afin d'évaluer l'impact des contre-mesures introduites par les auteurs de MBPIS, nous avons calibré un détecteur à partir des coefficients RS classiques, mesurés directement sur l'images I à analyser. Dans le cas de la stéganalyse RS classique, cela revient à calculer $R_M(p/2)$, $R_{-M}(p/2)$, $S_M(p/2)$, $S_{-M}(p/2)$. Notre détecteur est défini en associant à chaque image I le vecteur statistique à 4 coordonnées

$$I \longrightarrow V(I) = (R_M, S_M, R_{-M}, S_{-M}).$$

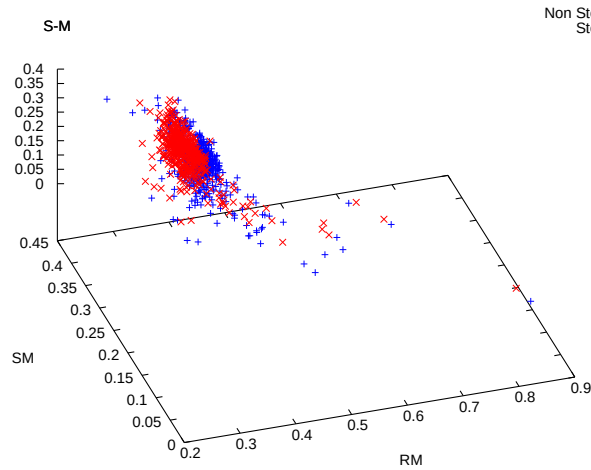
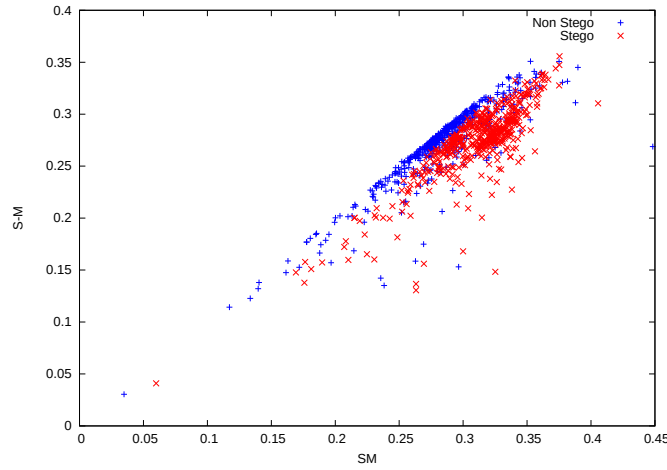
Remarque 7.1 :

Afin d'éviter toute confusion, il est important d'avoir à l'esprit que les coefficients R_M , R_{-M} , S_M , S_{-M} sont calculés sur toute l'image à l'aide d'une fenêtre de 4 pixels, tandis que les coefficients R_M , R_{-M} , S_M , S_{-M} définis par la relation (7.10) page 203 sont calculés uniquement sur les zones non homogènes à l'aide d'une fenêtre 2×2 dans le cadre de la stéganalyse MBPIS-RS locale.

Durant la phase d'apprentissage, nous avons repris les mêmes ensembles $\mathcal{C}$ et $\mathcal{S}$ utilisés pour calibrer notre détecteur dans le contexte de la stéganalyse MBPIS-RS locale. La figure 7.7 représente la projection spatiale de l'ensemble d'apprentissage dans le repère (S_M, S_{-M}, R_M) .

Une projection dans le plan (S_M, S_{-M}) , figure 7.8 page ci-contre, met plus clairement en évidence la séparation des deux populations sur l'ensemble d'apprentissage.

Enfin, nous avons calibré notre discriminant de Fisher afin d'obtenir les meilleurs taux de détection sur notre ensemble d'apprentissage. Nous obtenons ainsi le facteur discriminant

FIG. 7.7 – Projection spatiale de $\mathcal{C}$ et $\mathcal{S}$ sur (S_M, S_{-M}, R_M) FIG. 7.8 – Projection spatiale de $\mathcal{C}$ et $\mathcal{S}$ sur (S_M, S_{-M})

et le barycentre associé

$$u = \begin{pmatrix} -1.359554E + 02 \\ -3.060732E + 02 \\ +1.410827E + 02 \\ +3.137939E + 02 \end{pmatrix} \quad g = \begin{pmatrix} +3.687970E - 01 \\ +2.972389E - 01 \\ +4.053200E - 01 \\ +2.728809E - 01 \end{pmatrix}.$$

7.4.4 Phase de challenge, stéganalyse LSB-RS classique

Nous avons soumis au détecteur ainsi calibré les mêmes images que celles qui nous ont servi à évaluer les performances de notre détecteur dans le cadre de la stéganalyse MBPIS-RS locale. Les performances du détecteur sont représentées par les courbes ROC de la figure 7.9. Aux vues des résultats, il apparaît très clairement que la stéganalyse LSB-RS classique fournit des mesures (les coefficients RS) qui permettent de distinguer

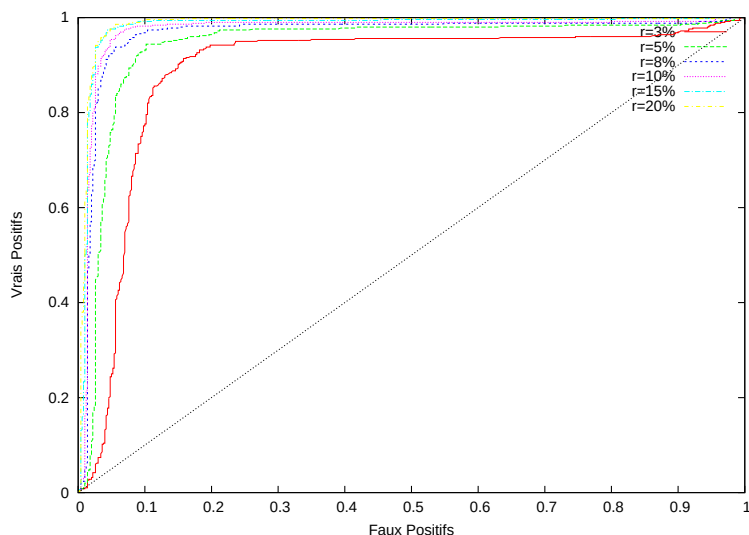


FIG. 7.9 – Courbes ROC du détecteur MBPIS, stéganalyse RS classique

efficacement les images stéganographiées avec MBPIS des autres. L'immunité annoncée par les auteurs ne semble pas avérée.

taux stégo.	$\mathcal{P}_{fp}$	$\mathcal{P}_{fn}$	$\mathcal{P}_{suc}$	Seuil
0.03	0.1	0.224	0.838	+0.63615
0.03	0.15	0.098	0.876	+1.1204
0.05	0.05	0.236	0.857	-0.36829
0.05	0.1	0.068	0.916	+0.63412
0.08	0.01	0.954	0.518	-4.24993
0.08	0.05	0.076	0.937	-0.37045
0.1	0.01	0.83968	0.57558	-4.24606
0.1	0.02	0.22846	0.87588	-2.16538

TAB. 7.2 – Taux de détection de MBPIS, stéganalyse LSB-RS classique, variables centrées

7.4.5 Comparaison des détecteurs

Nous avons calibré d'une part, un détecteur à partir des coefficients MBPIS-RS locaux, mesurés en ne prenant en compte que les zones non homogènes avec une fenêtre carrée et d'autre part, un détecteur à partir des coefficients LSB-RS classiques. Les simulations montrent clairement que se focaliser sur les zones non homogènes du support et adapter ainsi la stéganalyse RS classique donne de bien meilleurs résultats que l'application directe de celle-ci. Néanmoins lorsque le taux stéganographique est très faible ($< 3\%$), la stéganalyse LSB-RS classique fournit un détecteur sensiblement meilleur. Pour les taux supérieurs à 3%, le détecteur MBPIS-RS local est non seulement meilleur mais converge aussi beaucoup plus vite vers 1 comme l'illustre la figure 7.10 page suivante.

Comme dans beaucoup de situations de discrimination, le coût le plus important est emporté par les faux positifs ; la détection initiant souvent des traitements *a posteriori*. Dans le cas de la stéganographie, on va chercher à extraire l'information dissimulée. Quand

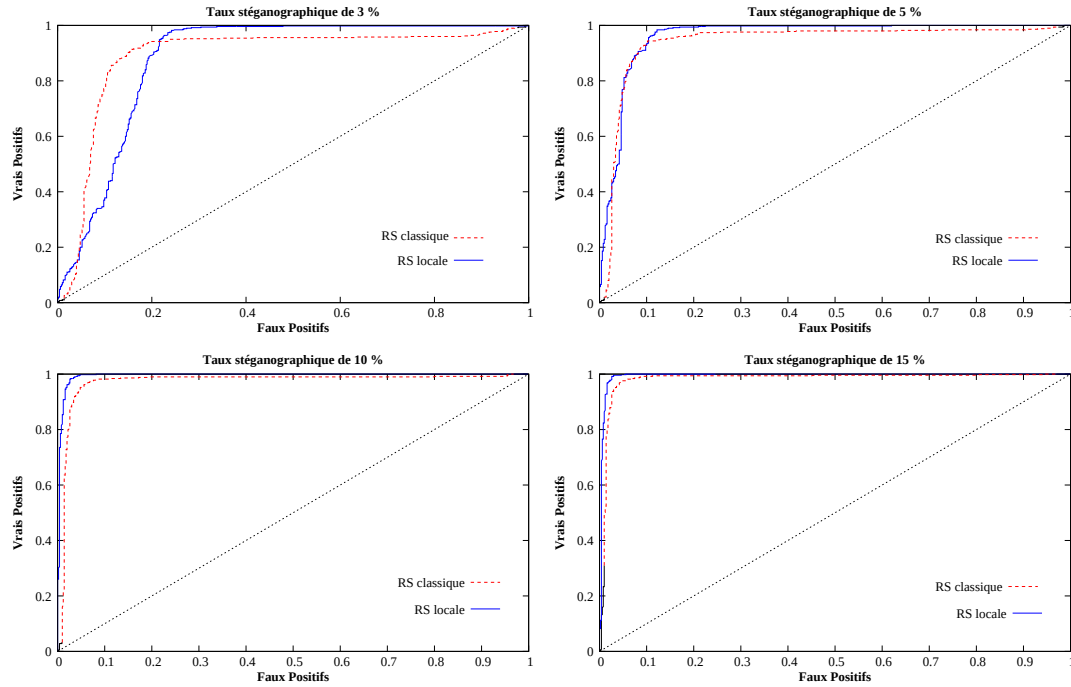


FIG. 7.10 – Comparaison des courbes ROC pour des taux de 3%, 5%, 10% et 15%

les algorithmes sont bien conçus, cette opération est équivalente à une recherche exhaustive sur la clé. On comprend maintenant beaucoup mieux pourquoi il est important de calibrer les détecteurs de manière à obtenir le minimum de faux positifs. Le tableau 7.3 page suivante permet de comparer les détecteurs pour des taux de faux positifs petits. Celui-ci met clairement en évidence l'avantage certain de la stéganalyse MBPIS-RS locale lorsque les taux de faux positifs sont faibles.

7.5 Conclusion

Nous avons présenté une technique pour détecter efficacement l'utilisation de l'algorithme de stéganographie *Multi Bit Plane Image Steganography*. Les auteurs affirment que leur algorithme est robuste à la stéganalyse RS de J. Fridrich *et al.* [77]. Les critères de conception de cet algorithme ne permettent pas, en effet, d'appliquer directement la stéganalyse RS avec MBPIS, notamment parce que la capacité et les zones de l'image exclues lors de l'insertion ne sont pas connues à l'avance. Néanmoins, une application directe de la stéganalyse RS avec l'insertion par LSB sur des images stéganographiées par MBPIS permet d'estimer le taux stéganographique. En revanche, cette estimation est moins fiable étant donné que les auteurs ont introduit des contre-mesures qui diminuent l'efficacité de la stéganalyse RS classique. Nous avons proposé une adaptation de cette stéganalyse en se focalisant exclusivement sur les zones utilisées par MBPIS. En considérant uniquement les zones potentiellement altérées par MBPIS, nous améliorons notablement les statistiques, auparavant biaisées par celles des zones exclues. Une première leçon peut alors être tirée de cette expérience. Il semble assez évident qu'exclure d'un schéma de stéganographie détecté par une stéganalyse donnée, des zones favorables à cette stéganalyse, ne permet pas d'obtenir la furtivité attendue. En effet, à l'instar de MBPIS, cela revient à concentrer

Taux Stégo.	$\mathcal{P}_{fp}$	MBPIS-RS locale		LSB-RS classique	
		$\mathcal{P}_{fn}$	$\mathcal{P}_{suc}$	$\mathcal{P}_{fn}$	$\mathcal{P}_{suc}$
0.03	0.01	0.928	0.531	0.99	0.5
0.03	0.05	0.8	0.575	0.75	0.6
0.03	0.1	0.622	0.639	0.224	0.838
0.05	0.01	0.772	0.609	0.98	0.505
0.05	0.05	0.232	0.86	0.236	0.857
0.05	0.1	0.066	0.917	0.068	0.916
0.08	0.01	0.27856	0.85586	0.954	0.518
0.08	0.05	0.00802	0.97097	0.076	0.937
0.08	0.1	0.002	0.94895	0.032	0.934
0.1	0.01	0.14629	0.92192	0.83968	0.57558
0.1	0.02	0.03808	0.97097	0.22846	0.87588
0.1	0.05	0.002	0.97397	0.04609	0.95195
0.15	0.01	0.13627	0.92693	0.49899	0.74624
0.15	0.02	0.02605	0.97698	0.14085	0.91976
0.15	0.05	0.002	0.97397	0.02414	0.96289
0.2	0.01	0.18437	0.9029	0.38431	0.80341
0.2	0.02	0.03407	0.97297	0.12072	0.92979
0.2	0.05	0.00401	0.97297	0.02213	0.96389

TAB. 7.3 – Comparaison stéganalyse MBPIS-RS locale et LSB-RS classique

des déviations statistiques observées sur un support, en des sous-parties de ce support. La bonne question qu'il faut plutôt se poser lors de la conception d'un algorithme de stéganographie est « *comment gommer ces déviations ?* ». Même s'il est assez naturel de dissimuler de l'information préférentiellement dans des zones non homogènes, il est surtout nécessaire de concentrer ses efforts sur la manière de rendre furtive l'information insérée. Aux vues de nombreuses stéganalyses de la littérature, la stéganographie dans le domaine spatial semble peu robuste. La stéganographie dédiée au format JPEG paraît alors plus adaptée, notamment parce qu'elle sépare intrinsèquement les zones homogènes des autres. En effet, pour des zones homogènes, les coefficients des hautes fréquences sont majoritairement à 0.

Stéganalyse dans le domaine fréquentiel compressé

« *Un problème sans solution est un problème mal posé.* »

Albert Einstein

Sommaire

8.1	Le domaine fréquentiel compressé	212
8.2	Stéganalyse universelle dans le DFC	215
8.2.1	Schéma de stéganalyse universelle	215
8.2.2	Résultats expérimentaux	217
8.3	Stéganalyse spécifique dans le DFC	221
8.3.1	Schéma de stéganalyse spécifique	221
8.3.2	Résultats expérimentaux	232
8.4	Conclusion et perspectives	235

CE chapitre présente une approche nouvelle en matière de stéganalyse adaptée au format JPEG. Partant du constat qu’il est difficile de préserver à la fois les statistiques dans le domaine fréquentiel, dans le domaine spatial et à la fois dans le *Domaine Fréquentiel Compressé* (DFC), nous proposons d’étudier ce dernier pour mettre en évidence des déviations statistiques permettant de discriminer efficacement les supports de couverture des stégo média. Jusqu’ici, les concepteurs et les stéganalystes ne se sont intéressés qu’aux domaines fréquentiel et spatial. Les uns pour maintenir une cohérence des statistiques dans ces deux domaines simultanément, les autres pour y découvrir des distributions de probabilité transformées après dissimulation. L’approche peut paraître quelque peu contre-intuitive au premier abord. En effet, le DFC est constitué d’une suite de bits représentant les coefficients DCT quantifiés après codage RLE et Huffman. En première approximation, nous pouvons penser que cette suite de bits présente des propriétés proches de l’aléa et est donc totalement inutile au stéganalyste. Néanmoins, l’implémentation du standard JPEG présente deux facteurs d’introduction d’un biais statistique dans le DFC. Tout d’abord, la norme JPEG propose d’utiliser des tables de Huffman pré-calculées et

performantes en moyenne pour gagner du temps mais aussi faire l'économie de l'envoi du dictionnaire dans l'entête du fichier JPEG. L'emploi de telles tables rend donc la compression de Huffman sous-optimale dans ce contexte précis. De plus, parmi les coefficients RLE, le coefficient V n'est pas compressé et son poids de Hamming n'est pas aléatoire comme nous l'expliquons par la suite. Il en résulte alors une déviation de l'entropie binaire de la suite de bits considérée. D'autre part, nous avons pu montrer que le critère d'avalanche de l'étape de compression sans perte du format JPEG est proche de 0.5. Forts de cette déviation et de cette propriété, nous avons conçu deux schémas de stéganalyse dédiés à la stéganographie JPEG : une stéganalyse universelle et un schéma de stéganalyse spécifique. Ces stéganalyses ont été validées expérimentalement à l'aide des algorithmes de référence, *Outguess*, *F5* et *JPHide and JPSeek*. Cette approche se distingue des stéganalyses classiques car nos schémas offrent tout d'abord des performances très élevées, mais celles-ci sont surtout quasi-indépendantes du taux stéganographique. De plus, nos schémas de stéganalyse détectent l'usage de stéganographie JPEG avec des taux d'erreur acceptables, pour des taux stéganographiques de 10^{-6} à 10^{-2} , à taux de détection constants, alors que les stéganalyses classiques peinent à détecter pour des taux supérieurs à 10^{-2} . En étant capable de détecter une image JPEG contenant un octet d'information dissimulé, nous avons repoussé aux extrêmes les limites du compromis entre l'indétectabilité et la capacité. Les résultats de ces travaux ont fait l'objet de publications dans des conférences internationales [16, 15] mais aussi dans une revue [17].

En guise d'introduction, nous présentons tout d'abord en détails le domaine fréquentiel compressé et mettons en lumière la déviation de l'entropie binaire que nous avons observée. Nous évaluons le critère d'avalanche de l'étape de compression sans perte du format JPEG et expliquons comment tirer profit à la fois de cette déviation et du critère d'avalanche pour concevoir des détecteurs stéganalytiques possédant des taux de détection quasi-indépendants du taux stéganographique, d'une part et pouvant détecter dès qu'un seul octet est dissimulé, d'autre part. Les distingueurs élaborés, sont tous linéaires et issus d'une analyse discriminante de Fisher présentée au paragraphe 6.3. Au paragraphe 8.2, nous concevons un schéma de stéganalyse universelle adapté au domaine fréquentiel compressé. Les coordonnées du vecteur statistique proposé sont mesurées de façon indépendante des algorithmes de stéganographie analysés. Nous expliquons et illustrons ensuite les limites intrinsèques d'un schéma de stéganalyse universelle, notamment sa sensibilité au changement de base d'images. De plus, nous montrons expérimentalement la propriété d'universalité de notre schéma, c'est-à-dire sa capacité à détecter efficacement l'utilisation d'algorithmes de stéganographie inconnus. Au paragraphe 8.3, nous définissons la *Méthode de Stéganographie Multiple* (MSM) et proposons une attaque par discrimination restreinte à trois coordonnées. Couplé à des distingueurs linéaires, cette attaque est redoutablement efficace et peu coûteuse comme l'illustre les résultats expérimentaux du paragraphe 8.3.2. Les simulations ont été effectuées à l'aide des algorithmes *Outguess*, *F5* et *JPHide and JPSeek* pour les deux schémas. En conclusion, nous positionnons nos travaux par rapport à l'état de l'art du domaine de la stéganalyse et mettons en lumière l'importance des résultats obtenus.

8.1 Le domaine fréquentiel compressé

La plupart des techniques de stéganalyse reposent sur l'observation de déviations statistiques dans le domaine spatial ou le domaine fréquentiel. Nous abordons le problème

d'un point de vue nouveau, en étudiant des statistiques dans le DFC. Cette démarche, bien que contre-intuitive de prime abord, présente deux avantages indéniables. Tout d'abord, l'évaluation des statistiques dans ce domaine est très simple puisqu'elle s'effectue directement sur le fichier binaire privé de ses entêtes. De plus, comme nous le mettons en évidence aux paragraphes 8.2.2 et 8.3.2, le DFC offre des possibilités de discrimination jusque là jamais atteintes. Dans ce paragraphe, nous étudions les propriétés du DFC et mettons en évidence les critères favorables à la discrimination qui justifient notre approche.

Nous avons détaillé au paragraphe 5.2 le format JPEG et pouvons le représenter de façon synthétique en deux étapes : une étape avec perte (changement d'espace, échantillonnage, transformation DCT, quantification) et une étape de codage et compression sans perte. Ces transformations travaillent dans trois domaines distincts, le domaine spatial, le domaine fréquentiel et le domaine fréquentiel compressé. Ces deux étapes et leurs domaines respectifs sont représentés sur la figure 8.1. Ce découpage du format JPEG implique une

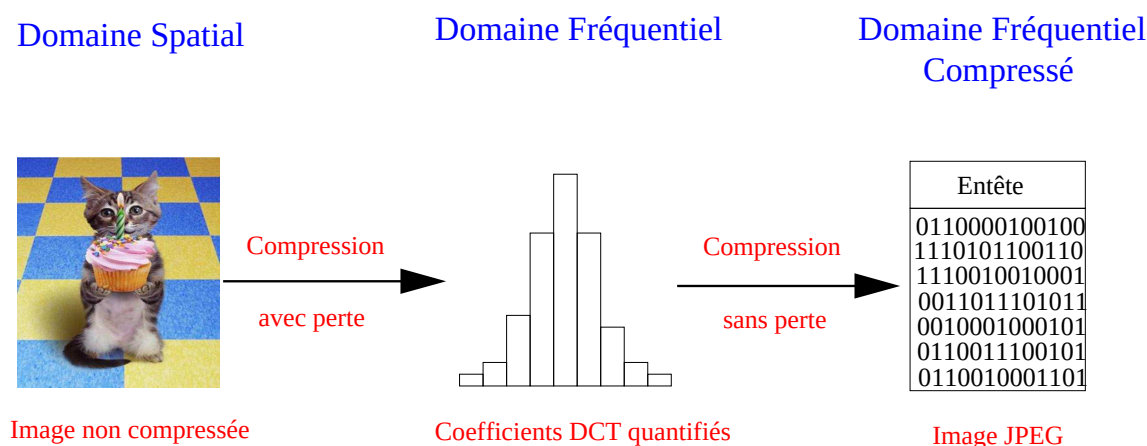


FIG. 8.1 – Étapes du JPEG et domaines associés

forte structure dans chacun des domaines mais surtout entre domaines. Par exemple, les transitions d'un pixel vers son voisin ne sont pas aléatoires et sont notamment fortement corrélées dans les zones homogènes. D'autre part, chaque coefficient DCT dépend de 64 pixels d'un même bloc. De plus, la présence d'une étape de compression avec perte rend extrêmement difficile la prédiction de l'impact dans un domaine donné, d'une modification effectuée dans un autre domaine. Nous pouvons émettre trois hypothèses heuristiques mais intuitives :

- insérer de l'information dans les coefficients DCT implique des modifications à la fois dans le domaine spatial et dans le DFC,
- il est très difficile de préserver à la fois les statistiques dans le domaine spatial, le domaine fréquentiel et le DFC,
- introduire de l'information supplémentaire dans un médium est susceptible de faire varier son entropie.

Soit I une image JPEG à analyser et (b_j) la suite de bits uniquement composée des coefficients DCT, compressés par RLE et Huffman (sans les entêtes du fichier JPEG). Nous avons remarqué une variation de l'entropie binaire des (b_j) lorsque le médium est

stéganographié. L'entropie binaire $H(I)$ est définie par

$$H(I) = -P(I) \log P(I) - (1 - P(I)) \log(1 - P(I)),$$

où $P(I) = \Pr(b_j = 0)$. $H(I)$ est alors une approximation de l'entropie au sens de Shannon. Observer une déviation sur $H(I)$ est équivalent à observer une déviation de P . Lorsque les images sont naturelles (non stéganographiées), P est une variable aléatoire qui suit une loi Gamma tandis que P suit une loi normale lorsque les images sont des stégo média. Plus surprenant, la loi normale suivie par P lorsque les images sont des stégo média est indépendante du taux stéganographique. Comme l'illustre la figure 8.2, cette loi est une loi normale de paramètres $(0.5, \sigma)$. Cette différence de lois de probabilité s'explique par le

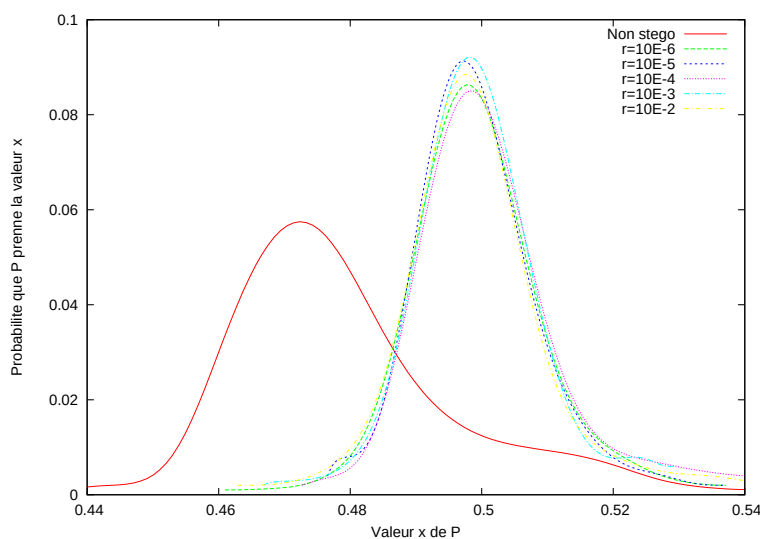


FIG. 8.2 – Lois de probabilité de P pour des supports et des stégo images générées par *JPHide*

critère d'avalanche de l'étape de compression sans perte. Le critère d'avalanche est introduit par Feistel [68] en 1973 comme outil cryptographique. Ce critère mesure la proportion de bits inversés pour un message chiffré lorsqu'un seul bit a été modifié sur dans le message clair. Les bons algorithmes de chiffrement et les bonnes fonctions de hachage possèdent un critère d'avalanche proche de 0.5. Ce critère met en évidence la propriété de diffusion d'un algorithme cryptographique. Notons Q le critère d'avalanche de l'étape de compression sans perte du format JPEG, $P(I)$ la probabilité que b_j vaille 0 avant insertion et $P'(I)$ la même probabilité après insertion. Alors $P'(I)$ s'exprime par

$$P'(I) = P(I)(1 - Q) + (1 - P(I))Q. \quad (8.1)$$

Si Q est proche de 0.5 alors $P'(I) \approx 0.5$. Comme l'illustre la figure 8.3 page suivante, lorsque seulement quelques octets de LSB de coefficients DCT ont été modifiés, presque la moitié des bits, après RLE et Huffman, ont été inversés. Il en résulte que $P'(I)$ se rapproche de 0.5. Ce phénomène est amplifié par les algorithmes de stéganographie eux-mêmes. En effet, afin de préserver les statistiques des coefficients DCT, de l'information supplémentaire est insérée pour corriger les distortions introduites lors de la dissimulation du message.

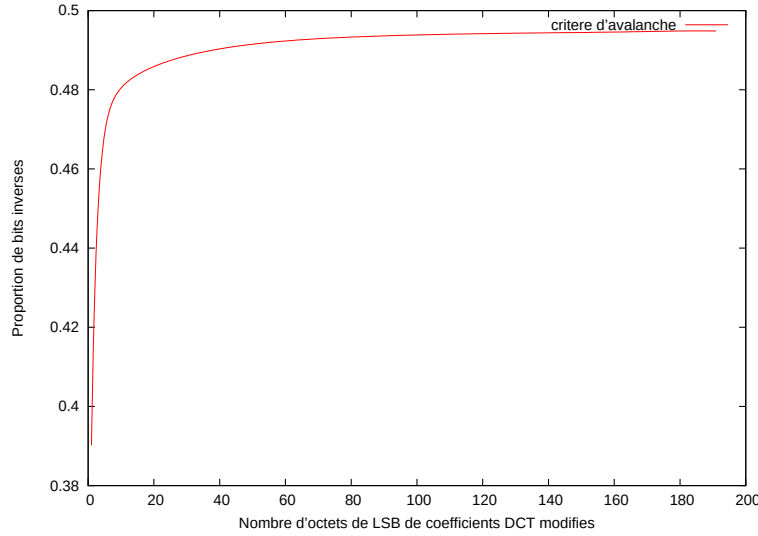


FIG. 8.3 – Critère d'avalanche de l'étape de compression sans perte du JPEG

Il apparaît alors clairement que quelle que soit la quantité d'information que l'on insère dans les LSB des coefficients DCT d'un médium de couverture, $P'(I) \approx 0.5$. Cette remarque est fondamentale car elle rend possible l'existence de détecteurs stéganographiques dont les performances sont quasi-indépendantes du taux stéganographique. Deux caractéristiques essentielles de l'étape de compression sans perte du format JPEG rendent les stéganalyses présentées aux paragraphes 8.2.2 et 8.3.2 efficaces. Tout d'abord, cette étape est bijective, ce qui implique qu'à toute déviation statistique observée dans le DFC correspond une déviation dans le domaine fréquentiel. Néanmoins, étudier ces déviations dans le DFC semble beaucoup plus aisé et sûrement plus discriminant. De telles fonctions peuvent être vues comme des loupes qui révèlent et amplifient des déviations statistiques non apparentes. Deuxièmement, un critère d'avalanche proche de 0.5 implique que quel que soit le changement effectué en entrée, la proportion de changements en sortie est toujours la même. La bijectivité et un critère d'avalanche proche de 0.5 sont deux critères qui définissent une nouvelle classe de fonctions qui doivent rendre possible la mise au point de détecteurs stéganographiques quasi-indépendants du taux stéganographique. L'étape de codage RLE et de compression d'Huffman est l'une d'entre elles.

8.2 Stéganalyse universelle dans le DFC

Nous avons mis en évidence au paragraphe précédent une déviation statistique qui semble apparaître dès lors que des algorithmes de stéganographie changent la valeur des coefficients DCT. D'autre part, cette déviation se mesure sans appel à un oracle d'extraction ni même d'insertion. À partir de cette déviation, nous avons conçu un schéma de stéganalyse universelle dont l'objectif est de tirer au maximum parti de la variation d'entropie binaire constatée.

8.2.1 Schéma de stéganalyse universelle

Soient I une image JPEG à analyser, (b_j) la suite binaire des coefficients DCT quantifiés, codés par RLE et compressés par Huffman, ainsi que $P(I)$ la probabilité que b_j vaille

0. P est donc une variable aléatoire définie sur l'ensemble des images JPEG qui ne suit pas la même loi de probabilité sur l'ensemble des supports de couverture et sur l'ensemble des stégo média. Cette mesure rend compte de la variation de l'entropie binaire, qui n'est autre que l'entropie des motifs de longueur 1.

Or, une telle variation, laisse présager une variation sur l'entropie des motifs de longueur fixée. Soit s la taille des motifs à considérer. Nous découpons la suite (b_j) en blocs B_j de longueur s octets, tels que

$$B_j = b_{j \times 8s+1} \dots b_{(j+1) \times 8s} \in \mathbb{F}_2^{8s}.$$

Nous évaluons ensuite le poids de Hamming $w(B_j) = \sum_{i=1}^{8s} b_{j \times 8s+i}$ de chaque bloc. Le poids de Hamming W est alors une variable aléatoire à valeurs dans $[0, 8s]$ définie sur l'ensemble des blocs de longueur s octets, $\mathbb{F}_2^{8s}$. Pour une image I , nous observons autant de mesures de W que celle-ci contient de blocs B_j . Nous avons donc une estimation expérimentale de loi de probabilité suivie par W . D'autre part, nous avons constaté, comme nous nous y attendions, que cette loi est différente selon que les mesures s'effectuent sur un support de couverture ou sur un stégo médium. Pour discriminer un stégo médium d'un support de couverture, nous mesurons une loi de probabilité et devons déterminer si celle-ci est plus proche d'une loi observée sur les supports de couverture ou d'une loi observée sur les stégo média. Pour ce faire, nous utilisons une pseudo-distance, appelée *distance de Kullbak-Liebler* ou encore *entropie relative* (cf. paragraphe 6 page 174). Nous avons pré-calculé expérimentalement une loi moyenne de référence, $\hat{p}(x)$, sur un ensemble de 1 000 images de couverture et mesurons l'écart à cette distribution par

$$\mathcal{D}(\hat{p}, p) = \sum_{x \in \Omega} \hat{p}(x) \log \frac{\hat{p}(x)}{p(x)}.$$

Comme cette pseudo-distance n'est pas symétrique, nous calculons séparément $\mathcal{D}_1(I) = \mathcal{D}(\hat{p}, p)$ et $\mathcal{D}_2(I) = \mathcal{D}(p, \hat{p})$. La figure 8.4 illustre bien que la distribution mesurée sur l'image non stéganographiée est plus « proche » de la distribution de référence que celle mesurée sur l'image après stéganographie. $\mathcal{D}_1(\cdot)$ et $\mathcal{D}_2(\cdot)$ sont donc deux nouvelles applications coordonnées de notre attaque par discrimination.

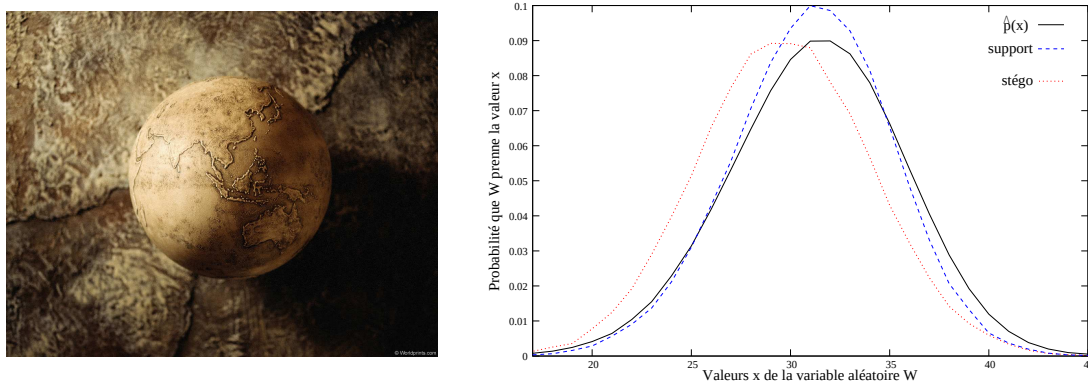


FIG. 8.4 – Média support et lois suivies par W avant et après insertion avec *Outguess*, $s = 8$

Finalement, puisque nous comparons des lois de probabilité, nous calculons aussi les moments statistiques jusqu'à l'ordre k fixé. Soit $\mathcal{M}_i(I) = E((W(I) - E(W(I)))^i)$ le moment d'ordre i de la loi de probabilité suivie par W sur I . Nous définissons alors notre attaque par discrimination par

$$I \longrightarrow V(I) = (\mathcal{M}_1(I), \dots, \mathcal{M}_k(I), \mathcal{D}_1(I), \mathcal{D}_2(I)),$$

paramétrée par (s, k) . En pratique nous prenons $s = 8, 16$ ou 32 et $k = 3$. Chaque composante du vecteur statistique est une variable aléatoire définie sur les images JPEG qui ne suit pas la même loi de probabilité sur les supports et les stégo média comme l'illustre le tableau 8.2.1.

Remarque 8.1 :

$P(I)$ apparaît dans le moment d'ordre 1. En effet, pour tout $n \in \mathbb{N}$, $E(W(I)) = P(I) \times 8s$. Celui-ci possède exactement le même pouvoir discriminant que W et n'apporte donc aucune information supplémentaire.

Satistiques	Support	<i>Outguess</i>	<i>F5</i>	<i>JPHide</i>
Moment d'ordre 1	+124.23	+117.51	+125.35	+125.66
Moment d'ordre 2	+66.92	+83.34	+65.98	+100.90
Moment d'ordre 3	-60.94	+56.79	-12.18	+2.58
Moment d'ordre 4	+13899.60	+20658.80	+13224.48	+29870.12
D_1	+0.297	+0.96	+0.223	+0.169
D_2	+0.387	+0.972	+0.250	+0.150
P	+0.515	+0.541	+0.510	+0.509

TAB. 8.1 – Évolution de V en dissimulant 1 octet dans l'image *monde.jpg*, $s = 32$

Nous avons mis en évidence des statistiques qui présentent une déviation lorsque les images sont stéganographiées par des algorithmes tels *Outguess*, *F5* ou *JPHide* and *JP-Seek*. Ces déviations sont illustrées par les figures 8.5 page 222, 8.6 page 223 et 8.7 page 224. Ces statistiques sont indépendantes de l'algorithme employé mais pour montrer l'universalité de notre attaque, nous devons encore calibrer un distingueur qui détecte efficacement l'utilisation d'algorithmes de stéganographie n'appartenant pas à l'ensemble d'apprentissage.

8.2.2 Résultats expérimentaux

Phase d'apprentissage

Afin de mettre en évidence la propriété d'universalité de notre stéganalyse, nous avons calibré 18 distingueurs différents. Soient $\mathcal{A} = \{\textit{Outguess}, \textit{F5}, \textit{JPHide}\}$, l'ensemble des algorithmes de stéganographie dont nous disposons ainsi que les 6 sous-ensembles $\mathcal{A}_i$ définis par

$$\begin{aligned} \mathcal{A}_1 &= \{\textit{Outguess}\}, & \mathcal{A}_2 &= \{\textit{F5}\}, & \mathcal{A}_3 &= \{\textit{JPHide}\}, \\ \mathcal{A}_4 &= \{\textit{Outguess}, \textit{F5}\}, & \mathcal{A}_5 &= \{\textit{Outguess}, \textit{JPHide}\}, & \mathcal{A}_6 &= \{\textit{F5}, \textit{JPHide}\}. \end{aligned}$$

Pour chaque sous-ensemble $\mathcal{A}_i$, nous avons composé aléatoirement un ensemble d'apprentissage $\mathcal{E}_1^{(i)}$ de 4 000 images, dont 2 000 supports de couverture et 2 000 stégo images, si

$\text{Card}(\mathcal{A}_i)=2$ et de 3 000 images, dont 1 500 supports de couverture et 1 500 stégo images sinon. Les stégo images se répartissent équitablement entre des images stéganographiées avec chacun des algorithmes de $\mathcal{A}_i$, pour des taux stéganographiques de 10^{-6} , 10^{-5} , 10^{-4} , 10^{-3} et 10^{-2} . Les sous-ensembles d'images stéganographiées sont donc de taille 200 images si $\text{Card}(\mathcal{A}_i)=2$ et 300 sinon.

Chaque sous-ensemble $\mathcal{E}_i$ a permis de calibrer 3 détecteurs $D_{3(i-1)+j}$ pour $j = 1 \dots 3$ avec trois valeurs distinctes de s , 8, 16 et 32. Les distingueurs $(D_{3(i-1)+j})_{i=1\dots 6, j=1\dots 3}$ sont issus d'une analyse discriminante de Fisher, présentée au paragraphe 6.3. Chacun d'entre eux est alors équivalent à la donnée d'un facteur discriminant u , vecteur de longueur 7, d'un barycentre associé g si les variables sont centrées et d'un seuil T . Ce seuil est à fixer en fonction du nombre de fausses alarmes maximum souhaité lors de la phase de challenge. Il ne se détermine qu'*a posteriori*. Dans la phase de challenge, nous avons soumis à chacun de nos détecteurs $D_{3(i-1)+j}$, des supports de couverture et des images stéganographiées avec des algorithmes appartenant à $\bar{\mathcal{A}}_i$. Pour chaque ensemble d'apprentissage, nous avons retenu celui des trois détecteurs dont le paramètre s donne les meilleurs résultats lors de la phase de challenge. Les distingueurs D' retenus pour chaque $\mathcal{A}_i$ sont représentés dans le tableau 8.2 page ci-contre.

Les images sont issues d'une base de données constituée d'environ 85 000 images JPEG, téléchargées aléatoirement sur Internet et de la base *www.worldprint.com*. Les paramètres JPEG de ces images sont très variés (taille, facteur de qualité, couleur ou noir et blanc, type d'appareil photo ...) et nous paraissent représentatifs des images JPEG que l'on peut trouver sur Internet. De plus, le paramètre k a été fixé à 5.

Phase de challenge

Pour chaque distingueur D'_i , nous avons composé aléatoirement des challenges de 2 000 images, dont 1 000 supports de couverture et 1 000 stégo images. Pour chaque challenge, les 1 000 stégo images sont générées avec un algorithme de $\bar{\mathcal{A}}_i$, pour des taux stéganographiques de 10^{-6} , 10^{-5} , 10^{-4} , 10^{-3} et 10^{-2} . Les sous-ensembles d'images stéganographiées sont donc de taille 200 images. Nous avons donc soumis à chaque détecteur D'_i , associé à $\mathcal{A}_i$, des supports des couverture et des images stéganographiées avec des algorithmes de $\bar{\mathcal{A}}_i$, c'est-à-dire inconnus lors de la phase d'apprentissage. Les courbes ROC sont représentées sur la figure 8.8 page 225 et les performances résumées dans les tableaux 8.3 page 220, 8.4 page 221 et 8.5 page 230 .

Limites

Nous avons évalué notre schéma de stéganalyse universel à l'aide d'un échantillon d'images représentatif d'Internet. Cela soulève deux questions importantes. Tout d'abord, on peut se demander dans quelles mesures les images de notre base de données sont réellement représentatives des images d'Internet. En effet, même si 85 000 images JPEG semble être un nombre raisonnable pour des simulations sur quelques milliers d'images, elles ne représentent qu'une infime proportion des images disponibles sur le net. D'autre part, si les images pornographiques nous apparaissaient majoritaires il y a encore un ou deux ans de cela, le développement conjoint des appareils numériques et d'Internet ont fait naître un engouement massif pour les sites de partage de photos numériques, notamment

D'_i	s	u	g
D'_1	32	$\begin{pmatrix} -6.075087E - 02 \\ -3.839914E - 04 \\ +4.421784E - 04 \\ +2.525400E - 06 \\ -5.790950E - 08 \\ -3.763350E + 00 \\ -4.005215E + 00 \end{pmatrix}$	$\begin{pmatrix} +9.455674E + 01 \\ +7.323546E + 02 \\ -3.414047E + 03 \\ +1.918682E + 06 \\ -1.540449E + 07 \\ +3.411930E - 01 \\ +1.557549E + 00 \end{pmatrix}$
D'_2	32	$\begin{pmatrix} -5.663508E - 02 \\ -4.412542E - 03 \\ +7.172690E - 05 \\ +3.175758E - 06 \\ +1.933487E - 09 \\ -2.852016E + 00 \\ -3.205219E + 00 \end{pmatrix}$	$\begin{pmatrix} +8.959408E + 01 \\ +7.375370E + 02 \\ -1.361984E + 03 \\ +1.958334E + 06 \\ -1.036354E + 07 \\ +3.878695E - 01 \\ +1.743684E + 00 \end{pmatrix}$
D'_3	32	$\begin{pmatrix} -8.686621E - 02 \\ -3.292457E - 03 \\ -4.498862E - 05 \\ +7.074311E - 07 \\ +5.928603E - 09 \\ +2.678978E + 00 \\ -2.955305E + 00 \end{pmatrix}$	$\begin{pmatrix} +9.245320E + 01 \\ +1.087318E + 03 \\ -5.983621E + 02 \\ +3.125936E + 06 \\ -8.598214E + 06 \\ +5.268383E - 01 \\ +9.335069E - 01 \end{pmatrix}$
D'_4	16	$\begin{pmatrix} +1.780489E - 01 \\ -1.466127E - 02 \\ -3.864907E - 04 \\ +2.750258E - 05 \\ +5.373460E - 07 \\ +1.996971E + 00 \\ -7.300707E - 01 \end{pmatrix}$	$\begin{pmatrix} +5.498874E + 01 \\ +3.027994E + 02 \\ -4.102567E + 03 \\ +2.758759E + 05 \\ -7.027489E + 06 \\ +3.112732E - 01 \\ +4.151011E - 01 \end{pmatrix}$
D'_5	16	$\begin{pmatrix} -1.497300E - 01 \\ -5.416273E - 02 \\ -2.140569E - 03 \\ +7.660233E - 05 \\ +1.459027E - 06 \\ -3.392191E + 00 \\ -3.123299E + 00 \end{pmatrix}$	$\begin{pmatrix} +5.044011E + 01 \\ +2.351162E + 02 \\ -2.454589E + 03 \\ +1.955665E + 05 \\ -4.233435E + 06 \\ +4.030748E - 01 \\ +1.128800E + 00 \end{pmatrix}$
D'_6	16	$\begin{pmatrix} +8.246617E - 02 \\ -2.786796E - 02 \\ -7.513114E - 04 \\ +4.230118E - 05 \\ +7.628112E - 07 \\ -1.217546E + 00 \\ +6.209087E - 02 \end{pmatrix}$	$\begin{pmatrix} +5.254587E + 01 \\ +2.690194E + 02 \\ -3.217135E + 03 \\ +2.373062E + 05 \\ -5.626573E + 06 \\ +3.492606E - 01 \\ +7.899292E - 01 \end{pmatrix}$

TAB. 8.2 – Distingueurs pour la stéganalyse universelle dans le DFC

au format JPEG. Tous ces sites sont pour la plupart protégés contre le téléchargement automatisé d'images. Il en résulte qu'il est maintenant impossible de télécharger des images

Taux Stégo.	$\mathcal{P}_{fp}$	F5 avec D_1 , $s = 32$		Outguess avec D_2 , $s = 32$	
		$\mathcal{P}_{fn}$	$\mathcal{P}_{suc}$	$\mathcal{P}_{fn}$	$\mathcal{P}_{suc}$
10^{-6}	0.005	0.332	0.83125	0.28614	0.85422
	0.01	0.32203	0.83425	0.2343	0.87768
	0.05	0.24227	0.85422	0.14257	0.90714
	0.1	0.18744	0.86021	0.10269	0.89416
10^{-5}	0.005	0.44068	0.77683	0.2652	0.8652
	0.01	0.36391	0.81378	0.22632	0.88168
	0.05	0.25224	0.84823	0.12762	0.90914
	0.1	0.19242	0.85022	0.07976	0.90265
10^{-4}	0.005	0.36527	0.81469	0.29212	0.85122
	0.01	0.3523	0.81868	0.22732	0.88168
	0.05	0.22455	0.86414	0.12064	0.91363
	0.1	0.16667	0.86713	0.07876	0.90315
10^{-3}	0.005	0.335	0.82976	0.22832	0.88318
	0.01	0.32901	0.83175	0.21535	0.88717
	0.05	0.24726	0.85172	0.12164	0.91712
	0.1	0.18943	0.85721	0.07976	0.90564
10^{-2}	0.005	0.33433	0.83067	0.99411	0.50206
	0.01	0.31138	0.83966	0.20259	0.8944
	0.05	0.23553	0.85714	0.11425	0.91622
	0.1	0.18064	0.86164	0.07656	0.90442

TAB. 8.3 – Performances de D_1 et D_2

JPEG de façon réellement aléatoire à partir d'Internet.

D'autre part, les coordonnées du vecteur statistique, définissant une attaque par discrimination d'une stéganalyse universelle, sont mesurées indépendamment des algorithmes de stéganographie. En revanche, elles ne sont pas toutes impactées de la même manière par chaque algorithme. Les lois de probabilité qu'elles suivent sur un ensemble de stégo média dépendent de l'algorithme qui a généré ces stégo média. Ces deux remarques impliquent que les performances d'un détecteur de stéganalyse universelle sont très sensibles au choix de la base de d'images. *A contrario*, les propriétés essentielles, dont l'universalité, doivent être conservées par changement de base d'images. Pour illustrer notre propos, nous avons calibré de nouveaux détecteurs à partir d'images choisies de façon aléatoire, dans un échantillon d'environ 35 000 images JPEG, exclusivement de la base *www.pbase.com*. Les images de cette base sont issues de contributions personnelles de milliers de gens de par le monde. Bien évidemment, les ensembles d'apprentissage contiennent des images de paramètres de taille, de facteur de qualité et de couleur divers. Leur taille est de 2 000 images réparties de façon identique au paragraphe 8.2.2. Les figures 8.9 page 226, 8.10 page 227 et 8.11 page 228 mettent en évidence les lois de probabilité suivies par les coordonnées de V lorsque les images sont issues de *www.pbase.com*.

Les détecteurs ainsi calibrés sont nettement moins performants que ceux construits avec la base de 85 000 images, néanmoins ils permettent de mettre en évidence l'universalité du schéma proposé ainsi que des taux de détection quasi-indépendants du taux

Taux Stégo.	$\mathcal{P}_{fp}$	F5 avec D_3 , $s = 32$		JPHide avec D_4 , $s = 16$	
		$\mathcal{P}_{fn}$	$\mathcal{P}_{suc}$	$\mathcal{P}_{fn}$	$\mathcal{P}_{suc}$
10^{-6}	0.005	0.30209	0.84823	0.95115	0.52172
	0.01	0.3001	0.84673	0.91924	0.5357
	0.05	0.19541	0.88517	0.84347	0.55167
	0.1	0.15753	0.88467	0.77866	0.55916
10^{-5}	0.005	0.29013	0.85172	0.91924	0.53570
	0.01	0.28714	0.85022	0.93121	0.52921
	0.05	0.17846	0.88867	0.83649	0.55267
	0.1	0.13958	0.88767	0.78265	0.55017
10^{-4}	0.005	0.30409	0.84473	0.93619	0.52871
	0.01	0.30409	0.84473	0.91326	0.53819
	0.05	0.21336	0.87519	0.82154	0.55966
	0.1	0.17946	0.87419	0.80160	0.55217
10^{-3}	0.005	0.32104	0.83724	0.97308	0.51023
	0.01	0.31605	0.83724	0.93719	0.52571
	0.05	0.26321	0.84923	0.84646	0.55517
	0.1	0.20439	0.86570	0.80060	0.55467
10^{-2}	0.005	0.34397	0.82676	0.96411	0.51473
	0.01	0.33799	0.82576	0.94616	0.52172
	0.05	0.25424	0.85172	0.87139	0.53620
	0.1	0.19741	0.87019	0.84845	0.53620

TAB. 8.4 – Performances de D_3 et D_4

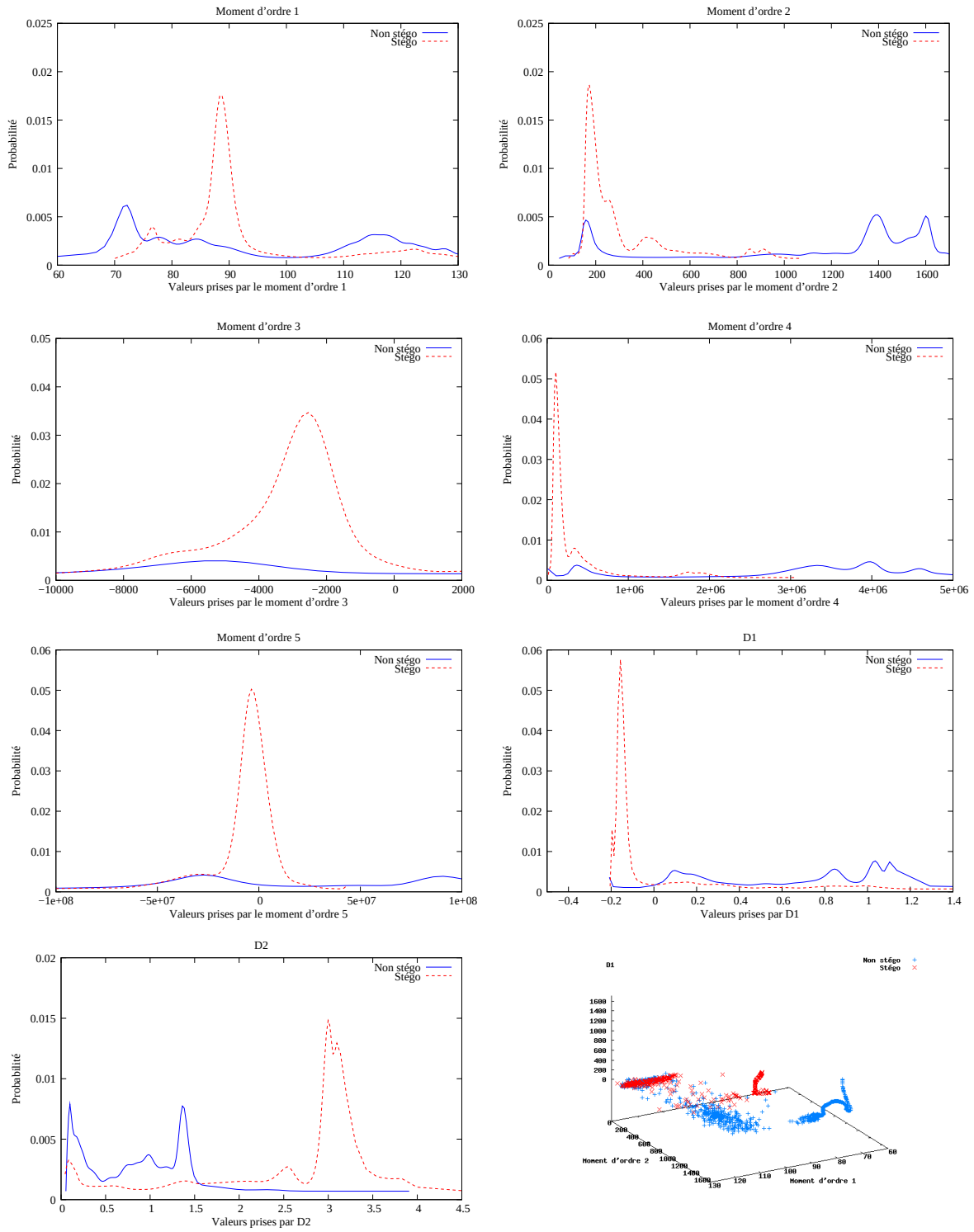
stéganographique qui sont les deux propriétés essentielles de notre schéma de stéganalyse.

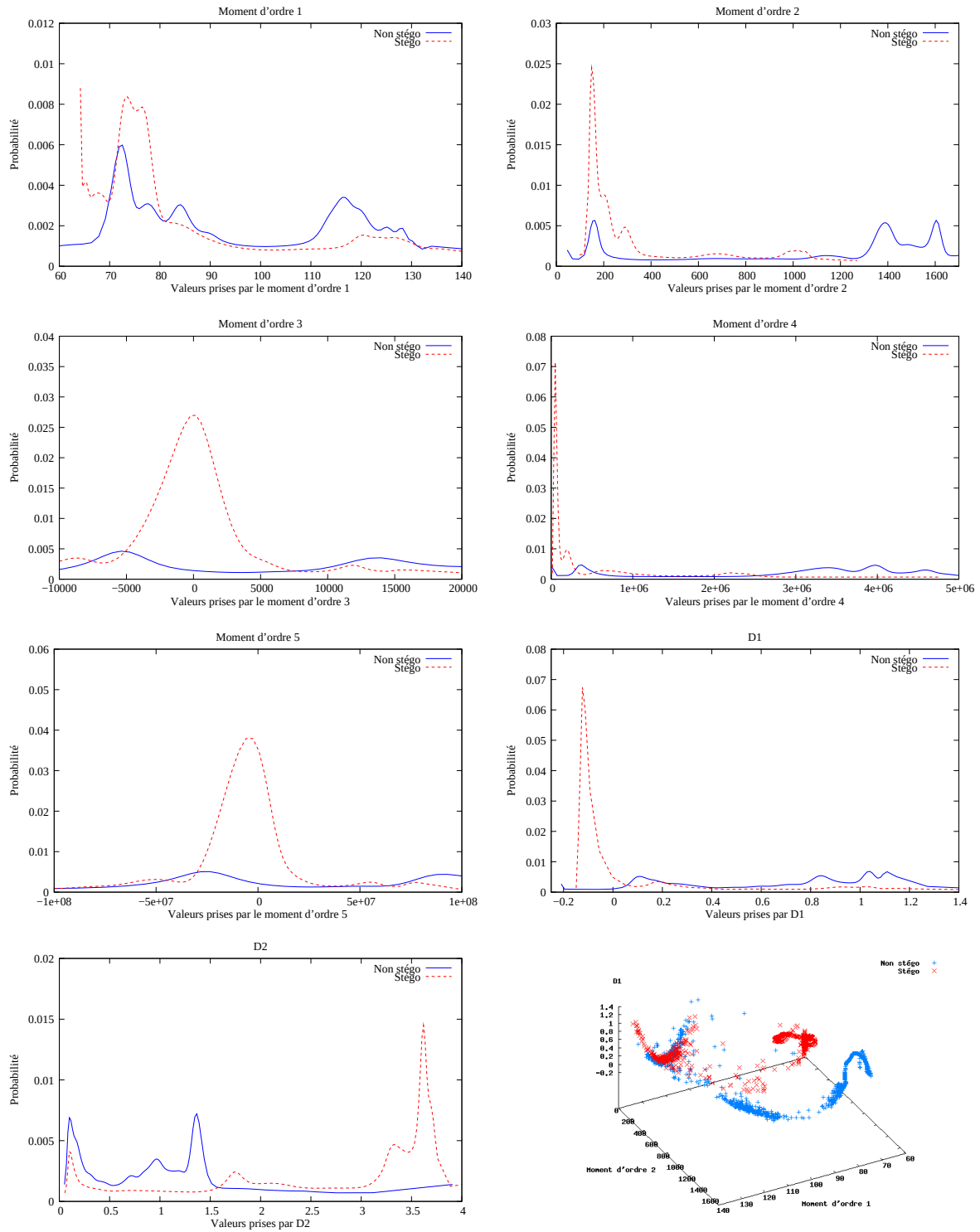
8.3 Stéganalyse spécifique dans le DFC

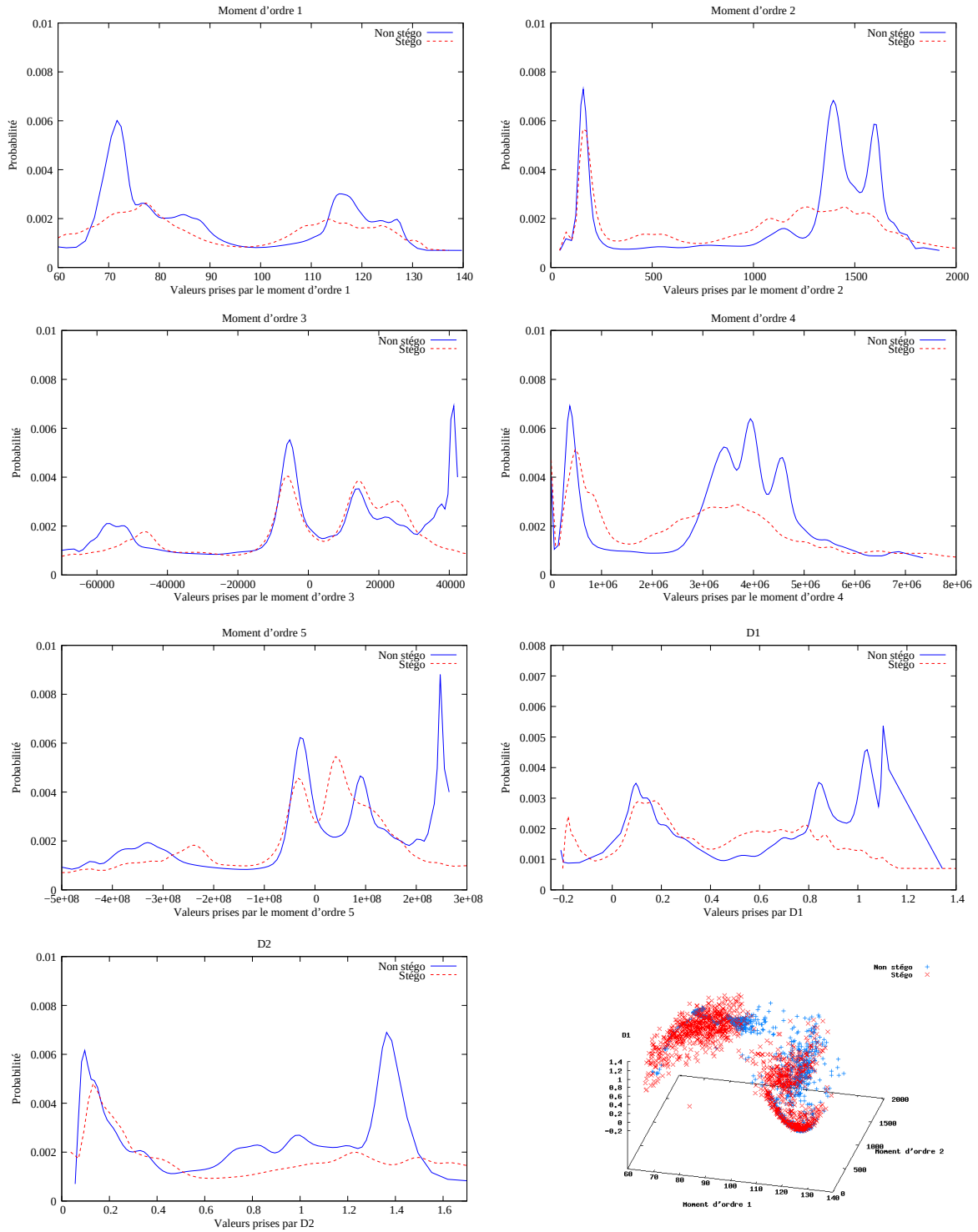
Nous avons mis en évidence au paragraphe précédent des statistiques qui permettent de calibrer des détecteurs dont les performances sont quasi-indépendantes du taux stéganographique. Nous avons montré par ailleurs, que notre schéma de stéganalyse est universelle au sens de la définition 6.11 page 186. Néanmoins, les performances de ces détecteurs sont sensibles au choix de la base d'images. Pour pallier cette sensibilité intrinsèque, nous proposons un schéma de stéganalyse spécifique qui tire profit des variations statistiques observées dans le DFC.

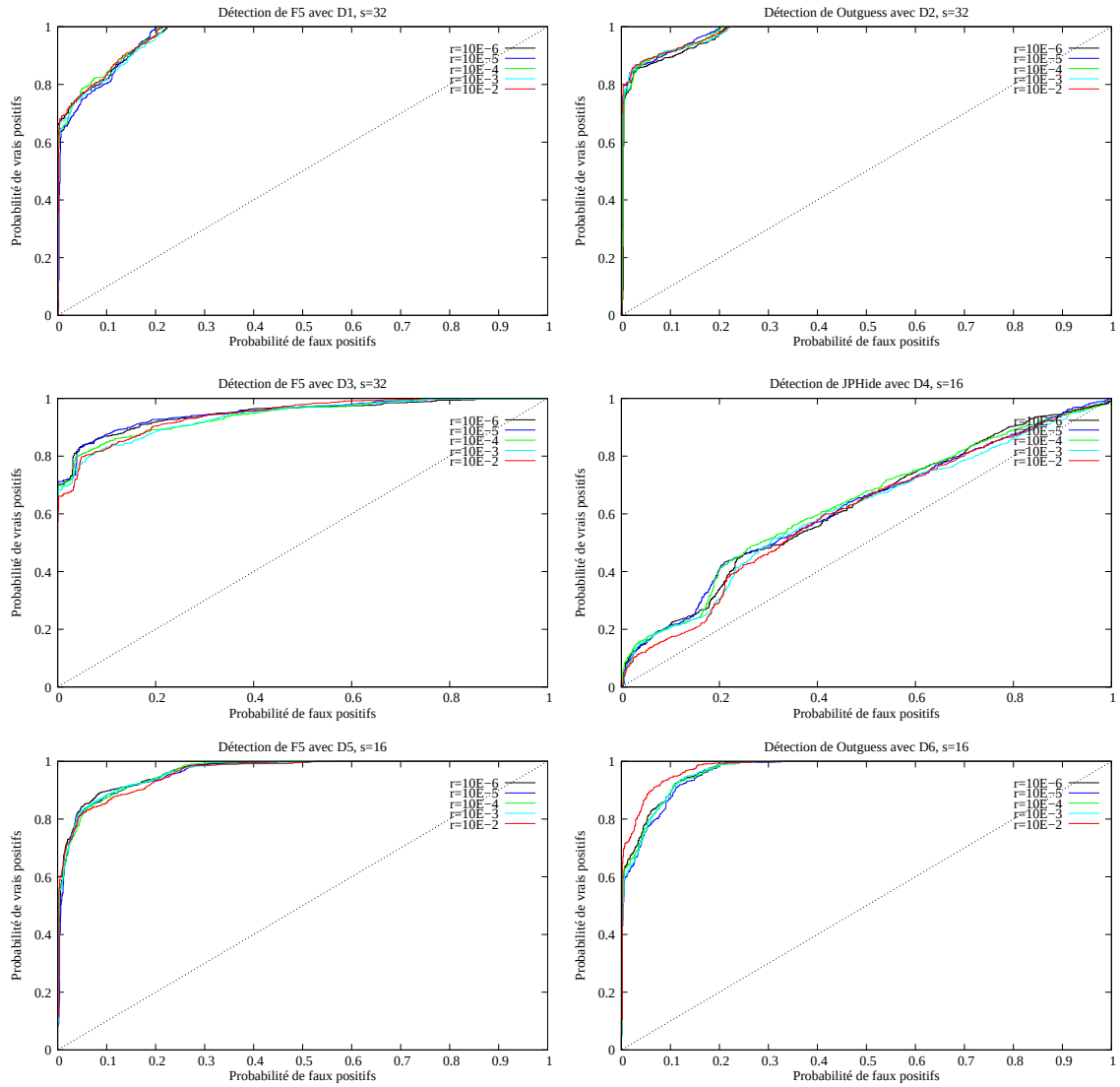
8.3.1 Schéma de stéganalyse spécifique

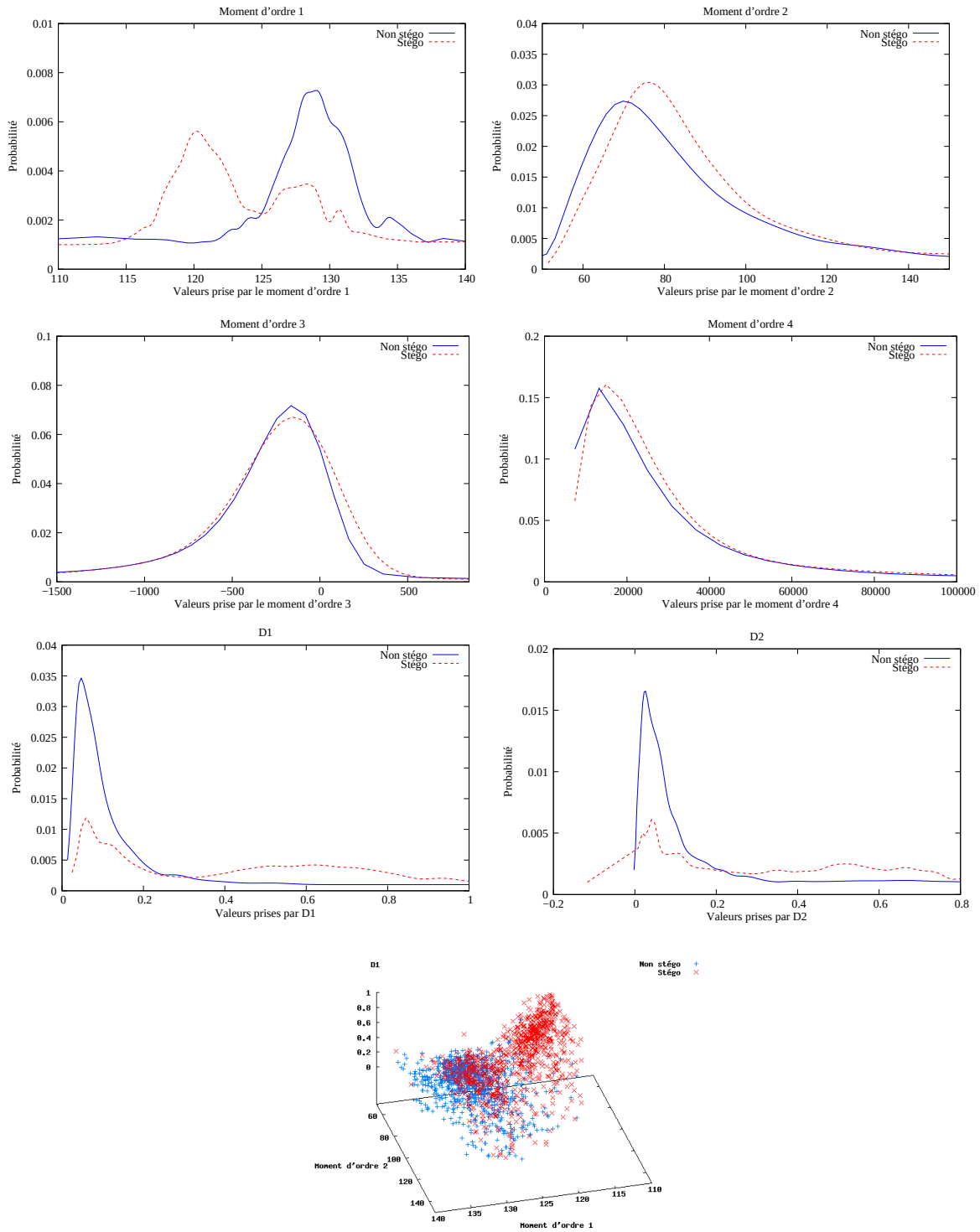
Le point central de la stéganalyse dans le DFC est la variation de l'entropie binaire lors de l'insertion d'information à l'aide d'algorithmes de stéganographie. D'après la relation (8.1) page 214, $P'(I) \approx 0.5$ après application d'un algorithme de stéganographie. De même, si nous dissimulons à nouveau de l'information alors $P''(I)$ après cette seconde insertion vérifie aussi $P''(I) \approx 0.5$. On en déduit que la variation de $P(I)$ après la première insertion est plus importante qu'après les suivantes. Le principe de la Méthode de Stéganographie Multiple (MSM ou MEM pour *Multiple Embedding Method* en anglais) découle de cette remarque. Elle consiste à stéganographier plusieurs fois l'image à analyser

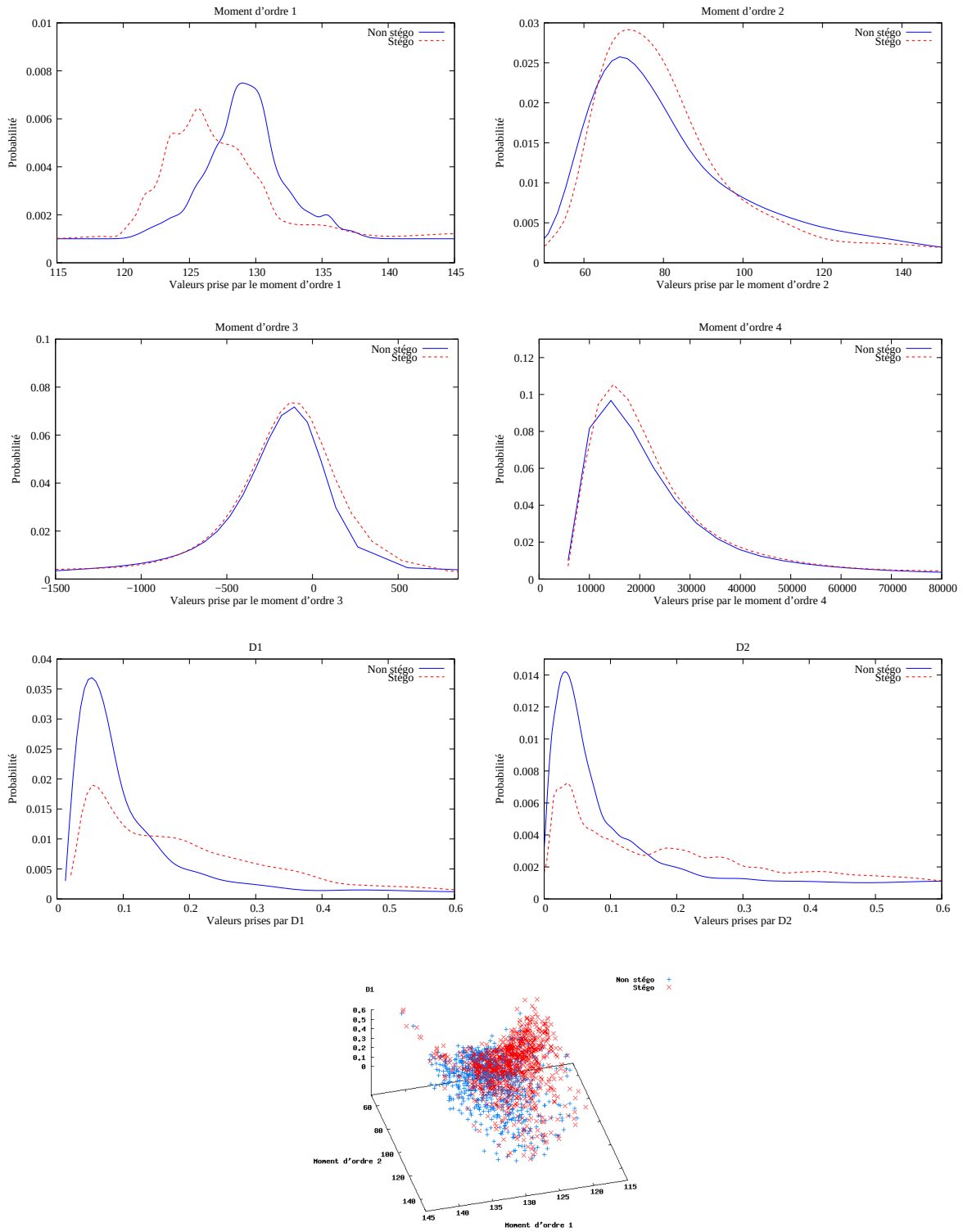
FIG. 8.5 – Distribution des coordonnées de V pour *Outguess*, $s = 32$

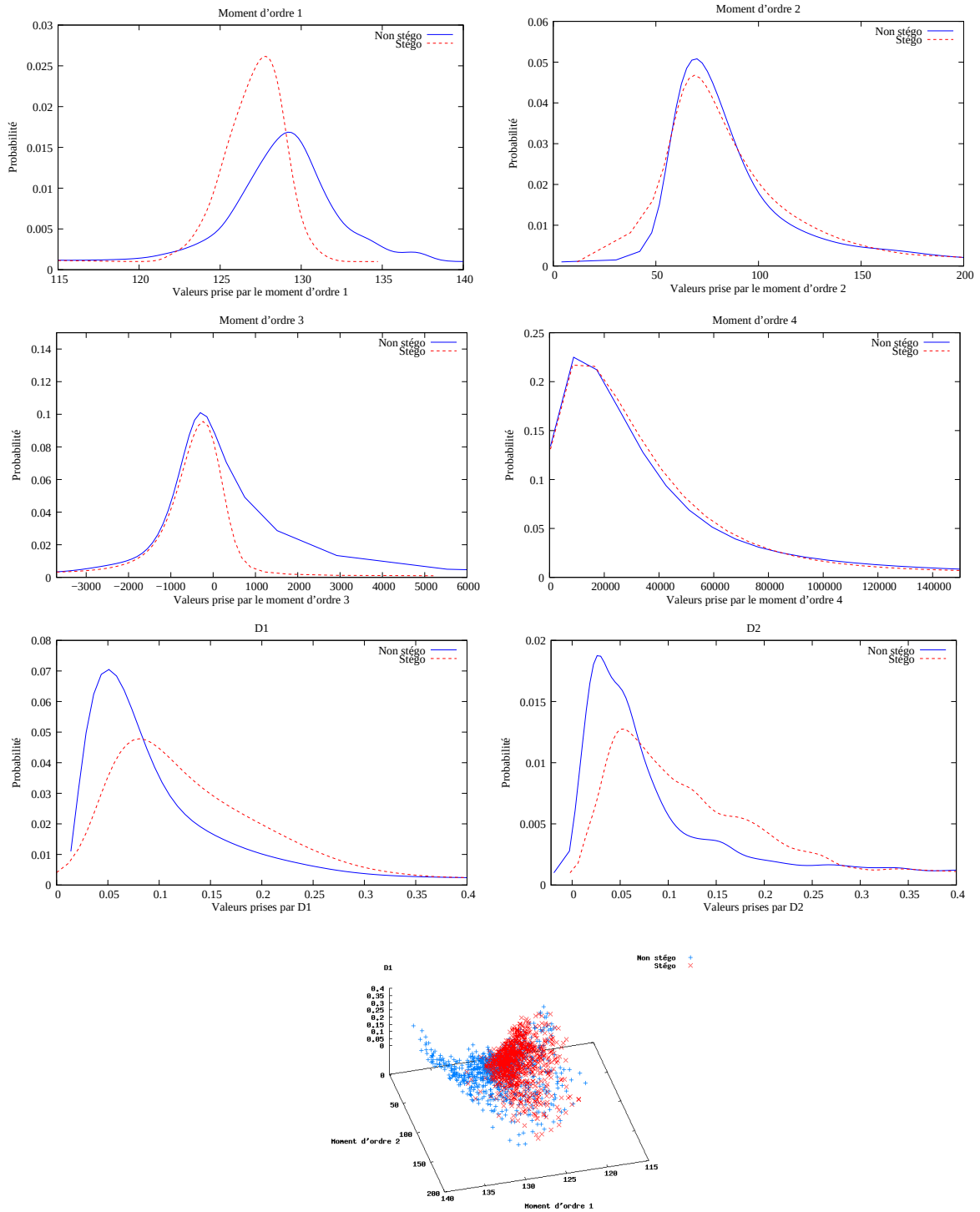
FIG. 8.6 – Distribution des coordonnées de V pour $F5$, $s = 32$

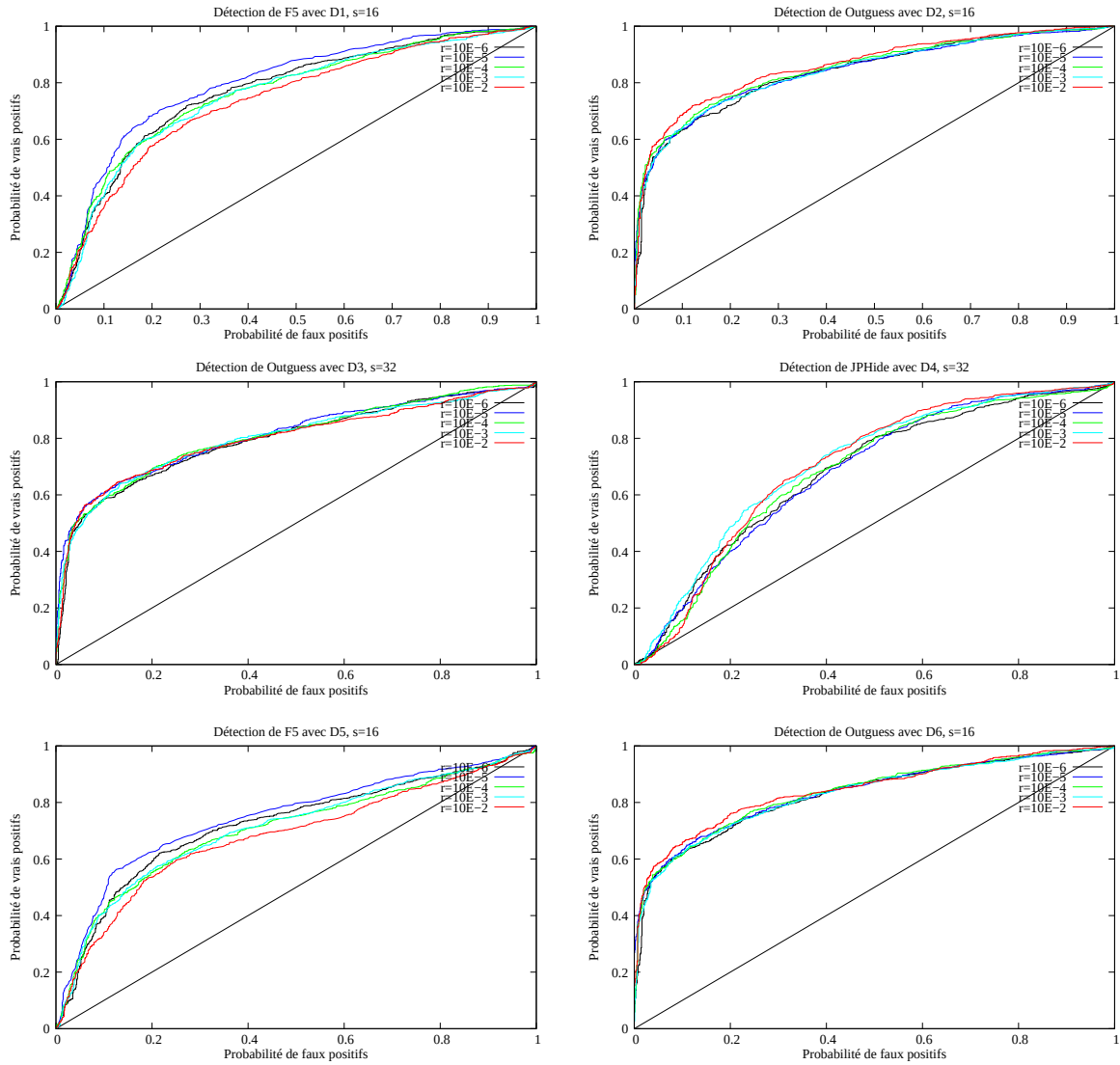
FIG. 8.7 – Distribution des coordonnées de V pour *JPHide*, $s = 32$

FIG. 8.8 – Meilleures courbes ROC pour D_1 , D_2 , D_3 , D_4 , D_5 et D_6

FIG. 8.9 – Distribution des coordonnées de V pour *Outguess*, $s = 32$

FIG. 8.10 – Distribution des coordonnées de V pour $F5$, $s = 32$

FIG. 8.11 – Distribution des coordonnées de V pour $JPHide$, $s = 32$

FIG. 8.12 – Meilleures courbes ROC pour D_1 , D_2 , D_3 , D_4 , D_5 et D_6

Taux Stégo.	$\mathcal{P}_{fp}$	$F5$ avec $D_5, s = 16$		$Outguess$ avec $D_6, s = 16$	
		$\mathcal{P}_{fn}$	$\mathcal{P}_{suc}$	$\mathcal{P}_{fn}$	$\mathcal{P}_{suc}$
10^{-6}	0.005	0.51249	0.82508	0.39382	0.8003
	0.01	0.41159	0.85585	0.39382	0.8003
	0.05	0.15984	0.91405	0.20937	0.86920
	0.1	0.10390	0.89565	0.12363	0.88967
10^{-5}	0.005	0.6341	0.78476	0.42871	0.78283
	0.01	0.46162	0.8389	0.42074	0.78632
	0.05	0.17846	0.90642	0.24925	0.85172
	0.1	0.11665	0.88971	0.16152	0.87369
10^{-4}	0.005	0.46407	0.84312	0.39182	0.8013
	0.01	0.43513	0.85113	0.37288	0.80929
	0.05	0.17465	0.90587	0.20638	0.86920
	0.1	0.12475	0.88652	0.08674	0.90315
10^{-3}	0.005	0.44666	0.84746	0.42074	0.78682
	0.01	0.44666	0.84746	0.40877	0.79231
	0.05	0.17049	0.90854	0.24427	0.85172
	0.1	0.11067	0.88952	0.12463	0.89166
10^{-2}	0.005	0.43956	0.85017	0.31152	0.85699
	0.01	0.40659	0.86054	0.31152	0.85699
	0.05	0.18282	0.90502	0.15394	0.90630
	0.1	0.14585	0.88462	0.07394	0.92110

TAB. 8.5 – Performances de D_5 et D_6

à l'aide de messages et de clés aléatoires et à mesurer les variations de $P(I)$. Si la première variation est « plus importante » que les suivantes, alors l'image est considérée comme non stéganographiée.

Soient Σ un schéma de stéganographie, d'algorithme d'insertion $\mathcal{Emb}(\cdot)$, $(K_i)_{i=1\dots n}$ une suite de clés stéganographiques, $(M_i)_{i=1\dots n}$ une suite de messages aléatoires de longueur fixée l et I l'image JPEG à analyser. Nous définissons tout d'abord la suite $\mathcal{I} = (I_i)_{i=0\dots n}$ par

$$\begin{cases} I_0 &= I, \\ I_i &= \mathcal{Emb}(K_i, M_i, I_{i-1}), \quad \forall i = 1 \dots n. \end{cases}$$

Pour mesurer les variations de $P(I)$, nous calculons la suite $\Delta = (\Delta_i)_{i=0\dots n}$ définie par

$$\begin{cases} \Delta_0 &= 0, \\ \Delta_i &= |P(I_i) - P(I_{i-1})|, \quad \forall i = 1 \dots n. \end{cases}$$

Si I n'a pas été stéganographiée par $\mathcal{Emb}$ alors

$$\begin{cases} \Delta_1 \gg \Delta_i, \quad \forall i > 1, \\ \Delta_i \text{ et } \Delta_j \text{ sont du même ordre de grandeur, } \quad \forall i, j > 1; \end{cases} \quad (8.2)$$

En revanche, si I est stéganographiée alors Δ_i et Δ_j sont du même ordre de grandeur $\forall i, j$. Par ailleurs, nous prenons en compte la variation relative en calculant la suite $Q = (Q_i)_{i=0\dots n}$, par

$$\begin{cases} Q_i = 0, & \forall i = 0 \dots 1, \\ Q_i = \frac{\Delta_{i-1}}{\Delta_i} \text{ si } \Delta_i \neq 0, \infty \text{ sinon,} & \forall i = 2 \dots n. \end{cases}$$

L'équation (8.2) page ci-contre implique

$$Q_2 \gg 1 \quad (8.3)$$

si I est un support de couverture et $Q_2 \approx 1$, sinon.

Nous pouvons alors définir formellement l'attaque par discrimination basée sur la MSM contre un schéma de stéganographie Σ . Nous associons à une image JPEG I , le vecteur statistique

$$I \longrightarrow V(I) = (P(I), \Delta(I), Q(I)),$$

où $\Delta(I) = \Delta_1$ et $Q(I) = Q_2$. Cette attaque est paramétrée par l ; en pratique, nous prenons l de l'ordre de quelques dizaines d'octets. Chaque coordonnée de $V(I)$ ne suit pas la même loi de probabilité sur les supports de couverture et les stégo média générés par Σ , comme l'illustre les figures 8.14 page 236 et 8.15 page 237.

Remarque 8.2 :

le choix du paramètre l est peu impactant dans la mesure où la variation de $P(I)$ est quasi-indépendant du taux stéganographique.

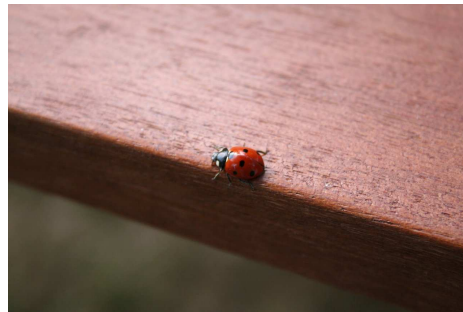


FIG. 8.13 – Image *coccinelle.jpg*

Le tableau 8.3.1 page suivante reprend les statistiques du support de couverture *coccinelle.jpg* (figure 8.13) dans laquelle un octet a été dissimulé avec *Outguess*, *F5* et *JPHide* (paramètre $l = 1$). L'image *coccinelle.jpg* fait environ 107 Ko, un octet dissimulé dans *coccinelle.jpg* correspond donc à un taux stéganographique $\rho = 9.17 \cdot 10^{-6}$. Les relations (8.2) page précédente et (8.3) sont alors vérifiées. Pour lire les statistiques de l'image stéganographiée une fois, il faut décaler les mesures d'une ligne vers le haut et annuler Δ_1 , Q_0 et Q_1 . Dans ce cas, on remarque que les relations (8.2) page précédente et (8.3) ne sont plus vérifiées.

	<i>Outguess</i>			<i>F5</i>		
i	P_i	Δ_i	Q_i	P_i	Δ_i	Q_i
0	0.5037	0	0	0.5037	0	0
1	0.5470	0.04334	0	0.5322	0.02851	0
2	0.5470	$7.147 \cdot 10^{-5}$	606.5	0.5319	$3.730 \cdot 10^{-4}$	76.45
3	0.5470	$2.297 \cdot 10^{-5}$	3.112	0.5318	$1.110 \cdot 10^{-4}$	3.36
4	0.5470	$3.914 \cdot 10^{-5}$	0.586	0.5315	$1.897 \cdot 10^{-4}$	0.585
	<i>JPHide</i>					
	P_i	Δ_i	Q_i			
0	0.5037	0	0			
1	0.5039	$1.797 \cdot 10^{-4}$	0			
2	0.5039	$4.659 \cdot 10^{-5}$	3.856			
3	0.5041	$1.277 \cdot 10^{-4}$	0.3646			
4	0.5040	$1.154 \cdot 10^{-4}$	1.1066			

TAB. 8.6 – Statistiques pour *coccinelle.jpg*, un octet dissimulé par *Outguess*, *F5* et *JPHide*

8.3.2 Résultats expérimentaux

Phase d'apprentissage

Afin d'illustrer notre schéma de stéganalyse, nous avons calibré 21 détecteurs différents. Pour chacun des algorithmes *Outguess*, *F5* et *JPHide*, nous avons entraîné 7 distingueurs de paramètres respectifs $l = 10, 50, 100, 200, 400, 600$ et 800 octets. Pour chacun de ces distingueurs, nous avons composé aléatoirement un ensemble d'apprentissage de 2 000 supports de couvertures. Lors de l'application de la MSM, chaque support de couverture a engendré une image stéganographiée avec l octets. Chaque détecteur a donc été entraîné avec 2 000 supports de couverture et 2 000 stégo images. Les distingueurs sont issus d'une analyse discriminante de Fisher, présentée au paragraphe 6.3. Chacun d'entre eux est alors équivalent à la donnée d'un facteur discriminant u , vecteur de longueur 3, d'un barycentre associé g si les variables sont centrées et d'un seuil T . Ce seuil est à fixer en fonction du nombre de fausses alarmes maximum souhaité lors de la phase de challenge. Il ne se détermine qu'*a posteriori*. Les distingueurs retenus sont représentés dans le tableau 8.7.

	<i>Outguess</i>	<i>F5</i>	<i>JPHide</i>
l	50 octets	10 octets	10 octets
u	$\begin{pmatrix} -3.897063E + 01 \\ +2.299124E + 02 \\ +2.046336E - 04 \end{pmatrix}$	$\begin{pmatrix} -1.779760E + 01 \\ +1.365236E + 02 \\ +3.349051E - 05 \end{pmatrix}$	$\begin{pmatrix} -1.143656E + 01 \\ +1.472741E + 02 \\ -7.731891E - 06 \end{pmatrix}$
g	$\begin{pmatrix} +4.875425E - 01 \\ +9.624832E - 03 \\ +1.292244E + 02 \end{pmatrix}$	$\begin{pmatrix} +4.849074E - 01 \\ +7.688310E - 03 \\ +1.279877E + 02 \end{pmatrix}$	$\begin{pmatrix} +5.069493E - 01 \\ +8.883871E - 03 \\ +7.207079E + 02 \end{pmatrix}$

TAB. 8.7 – Distingueurs pour la stéganalyse spécifique dans le DFC

Les images sont issues d'une base de données constituée d'environ 85 000 images JPEG, téléchargées aléatoirement sur Internet et de la base *www.worldprint.com*. Les paramètres JPEG de ces images sont très variés (taille, facteur de qualité, couleur ou noir et blanc, type d'appareil photo ...) et nous paraissent représentatifs des images JPEG que l'on peut

trouver sur Internet.

Phase de challenge

Pour chaque distingueur, nous avons composé aléatoirement des challenges de 2 000 images, dont 1 000 supports de couverture et 1 000 stégo images. Pour chaque challenge, les 1 000 stégo images sont générées avec l'un des algorithmes *Outguess*, *F5* ou *JPHide*, pour des taux stéganographiques de 10^{-6} , 10^{-5} , 10^{-4} , 10^{-3} et 10^{-2} . Les sous-ensembles d'images stéganographiées sont donc de taille 200 images. Les courbes ROC sont représentées sur la figure 8.16 page 237 et les performances résumées dans les tableaux 8.8 et 8.9 page suivante.

Taux Stégo.	$\mathcal{P}_{fp}$ max.	<i>Outguess</i> avec $l = 50$		<i>JPHide</i> , $l = 10$	
		$\mathcal{P}_{fn}$	$\mathcal{P}_{suc}$	$\mathcal{P}_{fn}$	$\mathcal{P}_{suc}$
10^{-6}	0.005	0.86667	0.56142	0.99301	0.50250
	0.01	0.79697	0.59188	0.48403	0.75125
	0.05	0.22929	0.85431	0.00299	0.96603
	0.1	0.06364	0.91777	0.00299	0.95055
10^{-5}	0.005	0.92908	0.53154	0.92223	0.53746
	0.01	0.92908	0.53154	0.92223	0.53746
	0.05	0.46707	0.74517	0.00299	0.97203
	0.1	0.02634	0.92879	0.00299	0.95504
10^{-4}	0.005	0.94995	0.52347	0.84447	0.57414
	0.01	0.86619	0.56173	0.84447	0.57414
	0.05	0.10827	0.91684	0.01097	0.97753
	0.1	0.03575	0.93878	0.01097	0.97753
10^{-3}	0.005	0.90918	0.54634	0.78963	0.60160
	0.01	0.83591	0.58116	0.78963	0.60160
	0.05	0.22807	0.85970	0.00299	0.96555
	0.1	0.05882	0.92985	0.00299	0.94658
10^{-2}	0.005	0.91147	0.62090	0.99202	0.50275
	0.01	0.87518	0.63403	0.99202	0.50275
	0.05	0.14949	0.90209	0.00299	0.97254
	0.1	0.04790	0.92299	0.00299	0.95956

TAB. 8.8 – Détection de *Outguess* et *F5*

Analyse

Nous avons expliqué au paragraphe précédent que le schéma de stéganalyse universelle présenté était sensible aux changements de base d'images. En revanche, le schéma de stéganalyse spécifique que nous venons de détailler, utilise des variations de mesures effectuées sur des images. Ces variations sont intimement liées à l'algorithme de stéganographie employé. Pour une même image, une même clé et un même message, chaque algorithme induit une variation de ces mesures qui lui est propre. De ce fait, le schéma proposé est peu sensible aux changements de base d'images. Pour s'en assurer, nous avons calibré de nouveaux détecteurs spécifiques à partir d'images choisies de façon aléatoire, dans un

		<i>F5</i> avec $l = 10$	
Taux Stégo.	$\mathcal{P}_{fp}$ max.	$\mathcal{P}_{fn}$	$\mathcal{P}_{suc}$
10^{-6}	0.005	0.98195	0.50604
	0.01	0.91073	0.53773
	0.05	0.51755	0.71730
	0.1	0.51755	0.71730
10^{-5}	0.005	0.92965	0.53104
	0.01	0.92965	0.53104
	0.05	0.69548	0.63806
	0.1	0.69548	0.63806
10^{-4}	0.005	0.98991	0.50479
	0.01	0.94450	0.52446
	0.05	0.82543	0.5789
	0.1	0.18063	0.85830
10^{-3}	0.005	0.90313	0.54610
	0.01	0.90313	0.54610
	0.05	0.90313	0.54610
	0.1	0.90313	0.54610
10^{-2}	0.005	0.93964	0.52648
	0.01	0.93964	0.52648
	0.05	0.84004	0.56329
	0.1	0.31187	0.77862

TAB. 8.9 – Détection de *JPHide*

échantillon d'environ 35 000 images JPEG, exclusivement de la base *www.pbase.com*. Les images de cette base sont issues de contributions personnelles de milliers de gens de par le monde. Bien évidemment, les ensembles d'apprentissage contiennent des images de paramètres de taille, de facteur de qualité et de couleur divers. Chaque ensemble d'apprentissage est composé de 1 000 supports de couverture et 1 000 stégo images stéganographiées à l'aide d'*Outguess*, *F5* ou *JPHide* à des taux stéganographiques de 10^{-6} , 10^{-5} , 10^{-4} , 10^{-3} et 10^{-2} . De plus, nous avons composé aléatoirement des ensembles de 2 000 challenges, dont 1 000 supports de couverture et 1 000 stégo images.

D'autre part, afin de comparer le pouvoir discriminant des statistiques spécifiques et universelles, nous avons pris comme nouvelle attaque par discrimination

$$I \longrightarrow V(I) = (\mathcal{M}_1(I), \dots, \mathcal{M}_4(I), \mathcal{D}_1(I), \mathcal{D}_2(I), \Delta(I), Q(I)).$$

Les figures 8.17 page 238 et 8.18 page 239 mettent en évidence les lois de probabilité suivies par Δ et Q lorsque les images sont issues de *www.pbase.com*. Nous pouvons alors remarquer qu'elles sont similaires à celles des images de la base initiale de 85 000 images.

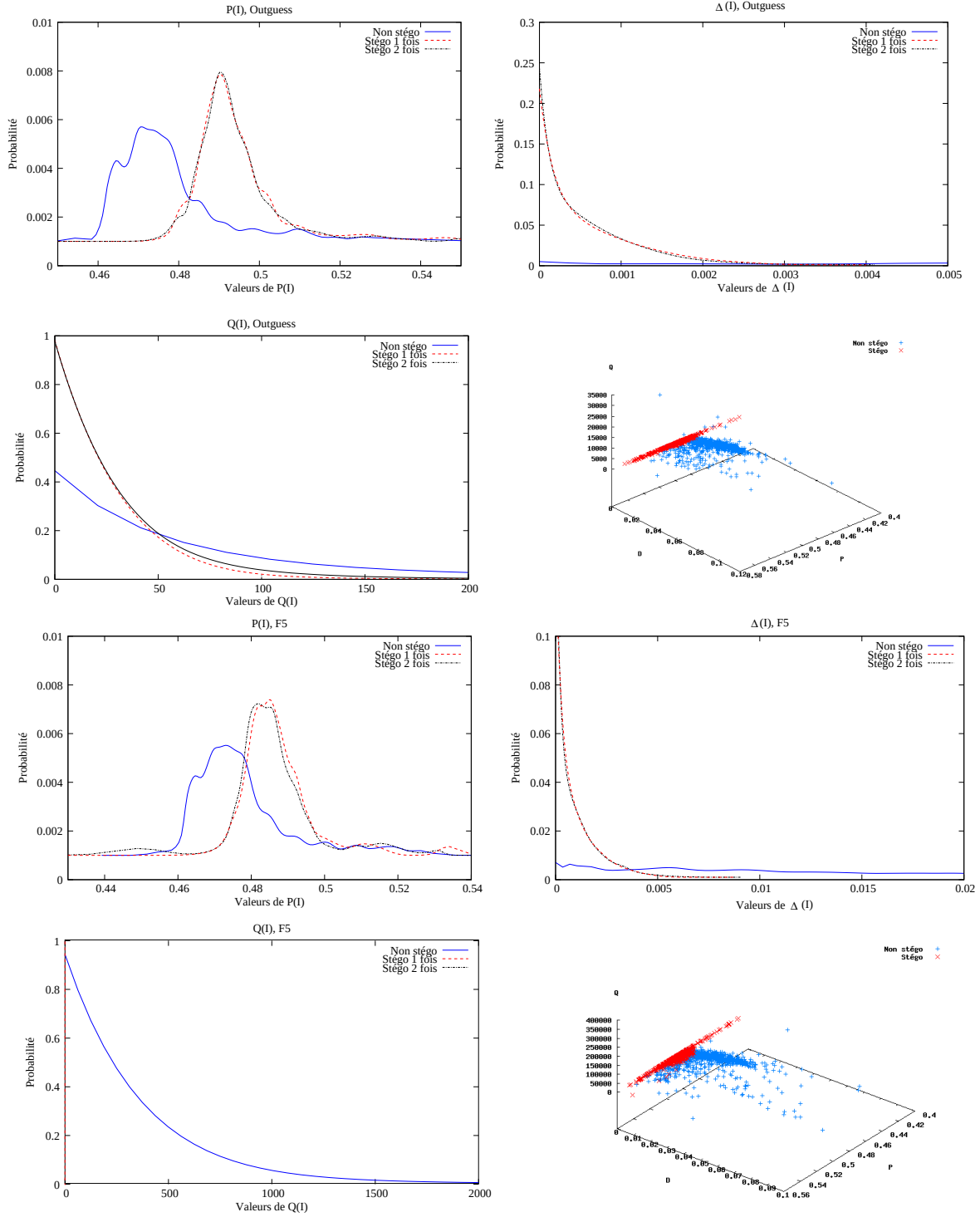
Aux vues des courbes ROC de la figure 8.19 page 239, il apparaît que le schéma de stéganalyse spécifique issue de la Méthode de Stéganographie Multiple, n'est pas sensible aux changements de base d'images, comme l'on pouvait s'y attendre. De plus, l'ajout de coordonnées supplémentaires a permis d'améliorer un peu les performances mais de manière non significative. D'autre part, les taux de détection sont, comme pour la stéganalyse

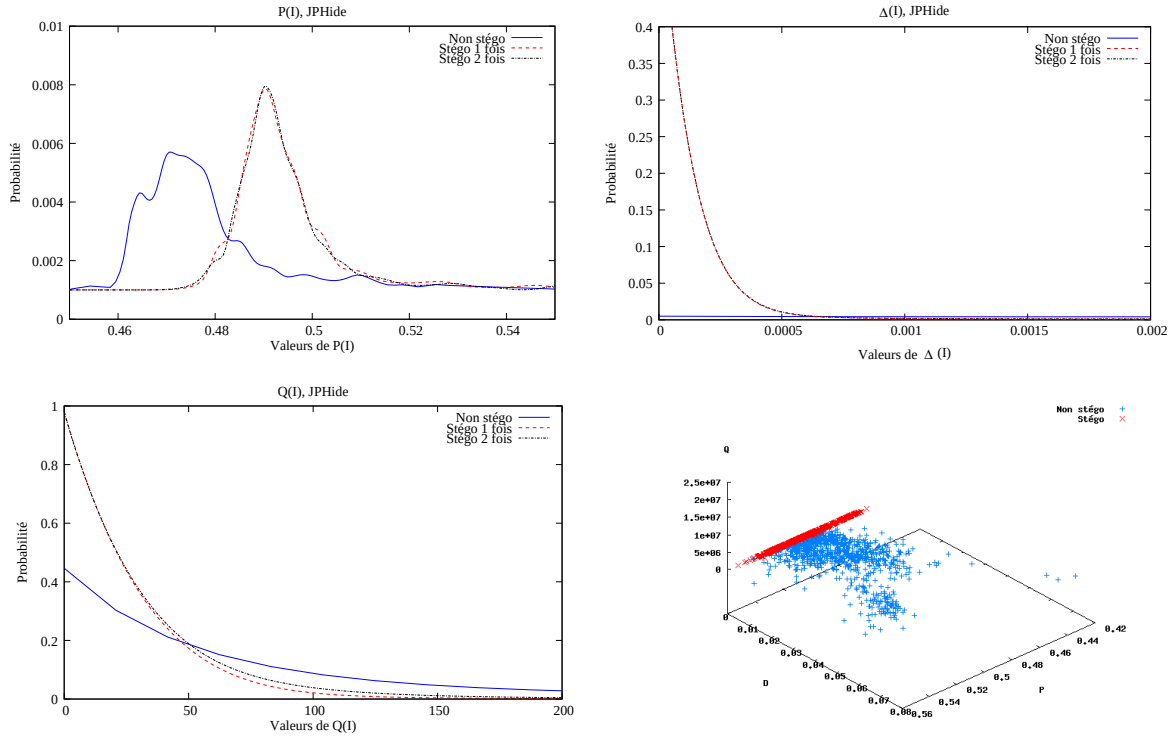
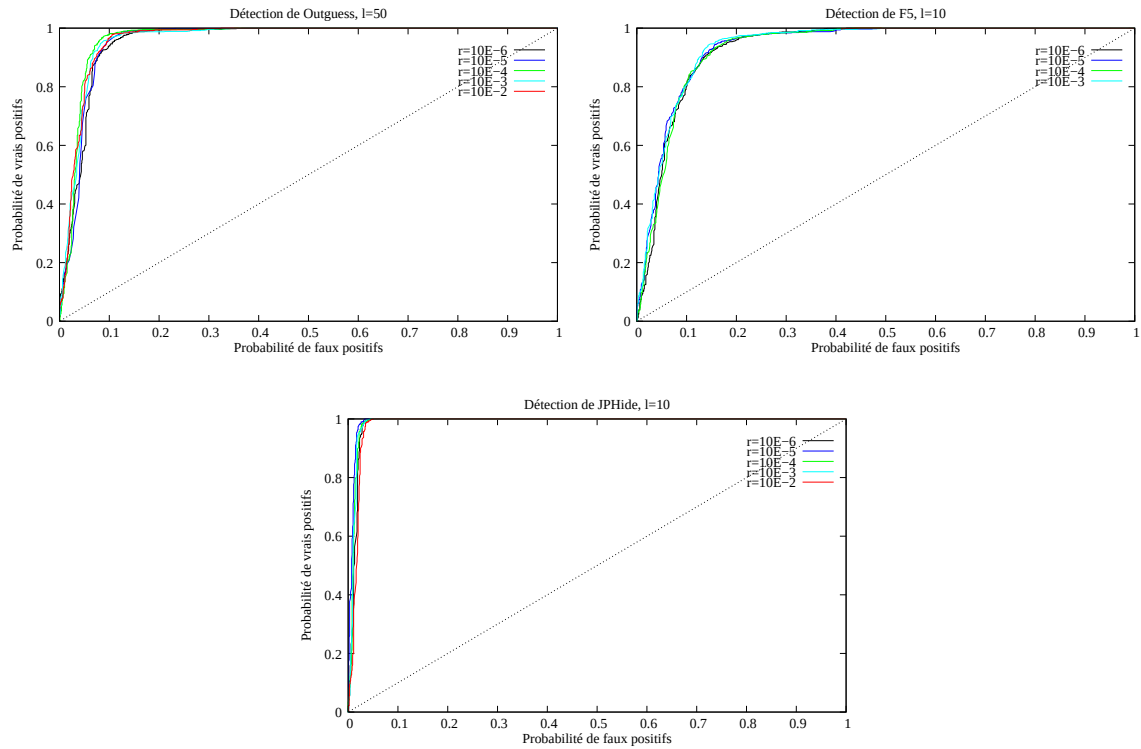
universelle, quasi-indépendants du taux stéganographique. Ce qui apparaît pour le moins contre-intuitif est la loi suivie par $P(I)$ lorsque I parcourt les images JPEG naturelles. En effet, tandis que Huffman (cf. paragraphe 5.2.5) est un algorithme de compression optimal, on s'attend à obtenir une distribution Gaussienne centrée en 0.5. Deux arguments peuvent expliquer cela. Tout d'abord, il faut rappeler que le codage de Huffman est optimal si et seulement si le dictionnaire est calculé en fonction du message. Or, pour économiser de la place, la majorité des logiciels gérant le format JPEG emploient les tables normalisées et ne stockent dans l'entête du fichier que le numéro de la table et non le dictionnaire en entier. Bien que ces tables aient été calculées pour être les meilleures en moyenne, elles ne sont pas optimales. D'autre part, seules les valeurs RL et S du codage RLE (cf. paragraphe 5.2.4) des coefficients AC et la valeur S du codage RLE des coefficients DC sont compressés par Huffman ; les valeurs V de ces coefficients étant considérées comme aléatoires. Ces valeurs V ne le sont clairement pas, les faibles valeurs étant majoritaires après la quantification. Le poids de ces valeurs n'est donc pas aléatoire ce qui permet ainsi d'expliquer la loi suivie par P .

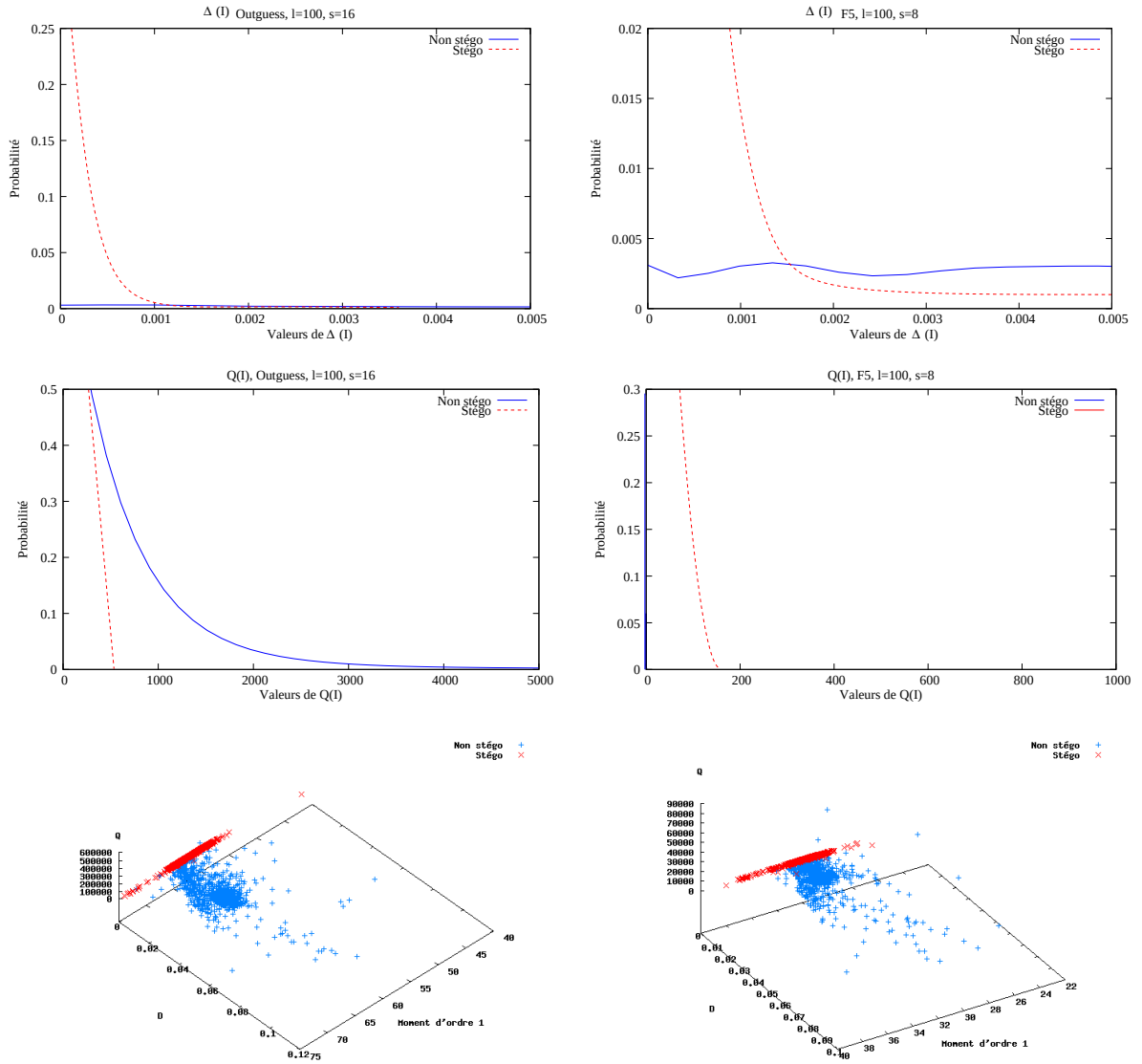
8.4 Conclusion et perspectives

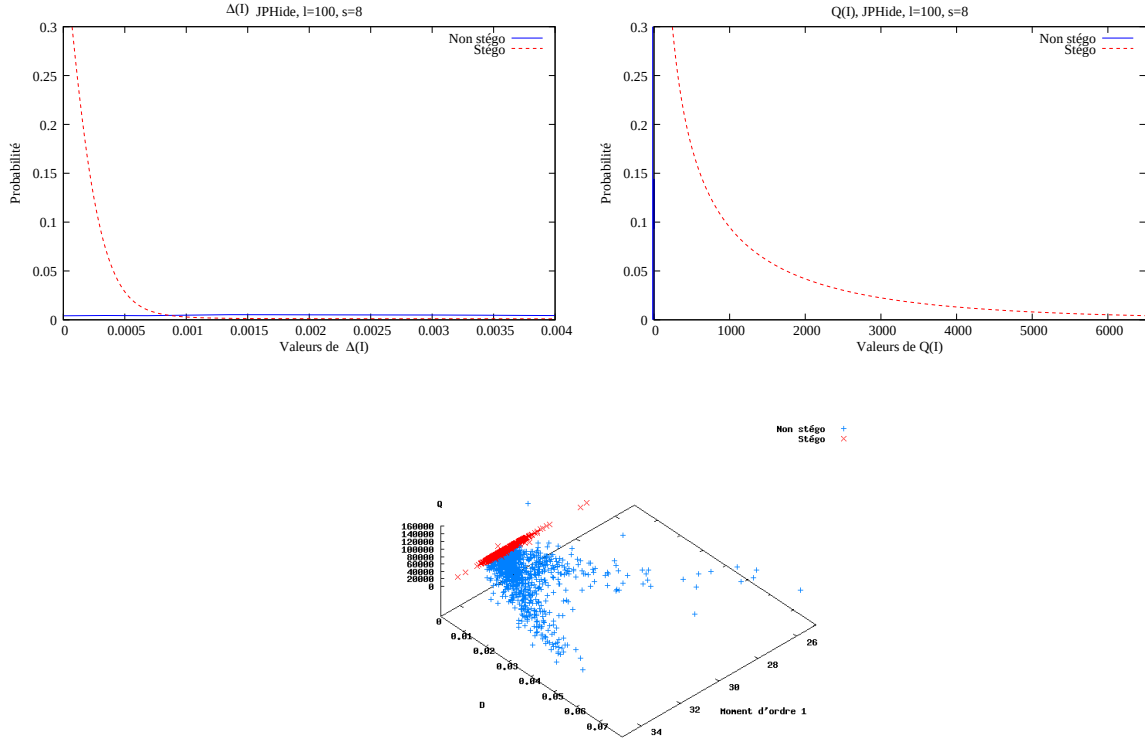
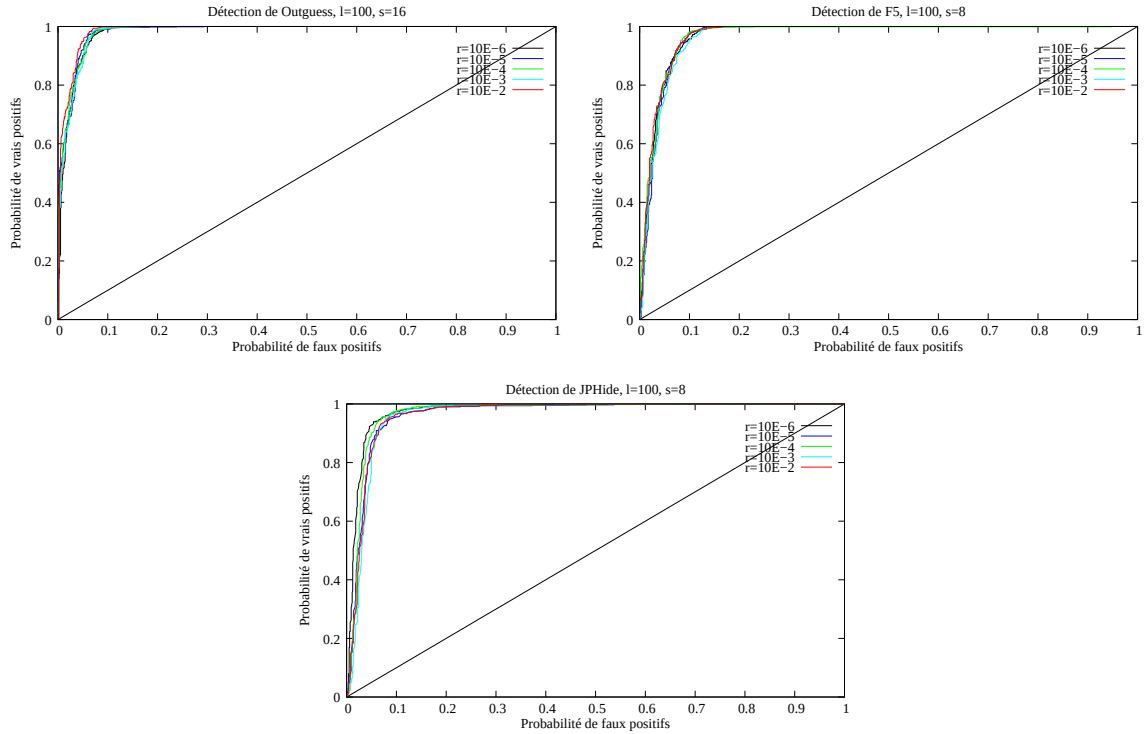
Au cours de ce chapitre, nous avons présenté une approche novatrice bien que contre-intuitive. En effet, la dernière étape du format JPEG étant une compression sans perte de Huffman, nous nous attendons à obtenir des données binaires qui ont des propriétés proches de l'aléa. Bien qu'Huffman soit un algorithme de compression démontré optimal, nous observons une déviation statistique de la répartition des bits à 0 sur l'ensemble des images JPEG naturelles. Cette déviation semble s'expliquer aisément si l'on considère que la plupart des logiciels manipulant les images JPEG préfèrent utiliser les tables de Huffman normalisées plutôt que de devoir recalculer le dictionnaire et le joindre en entête. Dans ce contexte, la compression appliquée à l'image n'est certes pas optimale mais du moins performante. D'autre part, si l'on regarde de plus près le format JPEG, on s'aperçoit que tous les coefficients RLE ne sont pas compressés. De plus, il est assez évident que la répartition des poids de Hamming pour ces valeurs non compressées n'est pas aléatoire. Cette déviation statistique de l'entropie binaire semblerait anodine sans une propriété forte de l'étape de compression sans perte. En effet, celle-ci possède un critère d'avalanche proche de 0.5. Ceci implique que lorsque quelques modifications seulement sont effectuées sur les données d'entrée (les coefficients DCT quantifiés), environ un bit sur deux est inversé en sortie. Or, des algorithmes de stéganographie dédiés au format JPEG n'ont pas d'autre choix que de dissimuler l'information dans les coefficients DCT quantifiés. Cela signifie, en d'autres termes, qu'il existe des détecteurs dont les performances sont quasi-indépendantes du taux stéganographique. Nous avons mis en évidence de tels détecteurs et avons repoussé à l'extrême le compromis entre l'indétectabilité et la capacité.

Nous avons décliné l'approche qui consiste à évaluer la déviation observée en deux schémas de stéganalyse extrêmement efficaces. Nous nous sommes tout d'abord placés dans le cas le plus défavorable et avons défini un schéma de stéganalyse universelle capable de détecter des algorithmes inconnus avec des taux de détection très élevés ($> 90\%$ pour $\mathcal{P}_{fa} < 0.1$) mais surtout constants en fonction du taux stéganographique, à $\mathcal{P}_{fa}$ fixée. Le schéma proposé est néanmoins, comme tout autre schéma de stéganalyse universelle, sensible aux changements de base d'images. Nous avons ensuite conçu un schéma de stéganalyse spécifique, la Méthode de Stéganographie Multiple, adapté au domaine

FIG. 8.14 – Distribution des coordonnées de V pour *Outguess* et *F5*

FIG. 8.15 – Distribution des coordonnées de V pour *JPHide*FIG. 8.16 – Meilleures courbes ROC pour *Outguess*, *F5* et *JPHide*

FIG. 8.17 – Distribution de Δ et Q pour *Outguess* et *F5*

FIG. 8.18 – Distribution de Δ et Q pour *JPHide*FIG. 8.19 – Meilleures courbes ROC pour *Outguess*, *F5* et *JPHide*

fréquentiel compressé. Cette technique offre de bonnes performances pour un coût calculatoire dérisoire. En effet, tandis que les stéganalyses spécifiques à l'état de l'art proposent des vecteurs statistiques de quelques dizaines de coordonnées [137, 83], nous évaluons un vecteur statistique de trois coordonnées. De plus, les complexités en temps et en mémoire de chacune de nos coordonnées sont linéaires en la taille de l'image; elles se réduisent à un comptage du nombre de zéros dans les données binaires du fichier JPEG. Outre des taux de détection élevés, nos stéganalyses possèdent deux propriétés qui leur sont propres. Tout d'abord, quelle que soit la quantité d'information dissimulée, les taux de détection sont constants à $\mathcal{P}_{fa}$ fixée. De ce fait, nous sommes capables, de détecter des stégo média stéganographiés à des taux compris entre 10^{-6} et 10^{-2} , alors que les stéganalyses classiques commencent seulement à détecter difficilement des taux stéganographiques de l'ordre de 10^{-2} . Pour des taux stéganographiques avoisinants les 10^{-6} , la quantité d'information insérée est de l'ordre de un octet. Nos schémas sont pour le moment, les seuls à pouvoir détecter des taux stéganographiques de cet ordre et à posséder des taux de détection quasi-indépendants du taux stéganographique, tant pour des schémas spécifiques qu'universels. De ce fait, c'est ce qui caractérise l'originalité de notre approche, nous n'avons pas pu effectuer des simulations comparatives.

Les schémas proposés ont été validés avec des algorithmes de référence, *Outguess*, *F5* et *JPHide and JPSeek*, socle commun de comparaisons des stéganalyses classiques. Au-delà de ces trois algorithmes, nous conjecturons que la plupart des algorithmes dédiés au format JPEG introduisent la même déviation que celle que nous avons observée. C'est un travail long et fastidieux qui reste néanmoins à faire pour montrer que **le format JPEG n'est intrinsèquement pas adapté** pour faire de la stéganographie. Cette étude aura permis de mettre en lumière une nouvelle classe de fonctions qui semble être prometteuse dans l'élaboration de futurs schémas de stéganalyse. Les fonctions de cette classe, dont l'étape de compression sans perte du format JPEG fait partie, sont tout d'abord bijectives mais possèdent aussi un critère d'avalanche proche de 0.5. Ces deux propriétés nous ont permis de concevoir des détecteurs de performances quasi-indépendantes du taux stéganographique et capable de détecter la dissimulation d'un seul octet. Mettre en évidence de telles fonctions paraît être un axe de recherche fécond pour la stéganalyse; cette étude en est la parfaite illustration. Une autre approche peut consister à essayer d'adapter la fonction RLE+Huffman ou Huffman à d'autres domaines pour bénéficier des mêmes propriétés, à l'image de S. Lyu et H. Farid [136] qui proposent de détecter la stéganographie dans le domaine spatial, en évaluant des déviations statistiques dans le domaine fréquentiel des transformées en ondelettes. Le principe est de se servir de cette fonction comme une « loupe » afin de grossir des déviations statistiques autrement invisibles.

Conclusion et perspectives

*« Il faut toujours se réserver le droit de rire
le lendemain de ses idées de la veille. »*

Napoléon Bonaparte

DANS cette partie, nous avons abordé l'étude des canaux de communication dans un contexte non coopératif sous l'angle de la stéganalyse. La stéganographie peut en effet être assimilée à une action de codage de canal adapté pour des canaux non physiques. De plus, la furtivité recherchée en fait naturellement un canal non coopératif. Tout comme les canaux de communication classiques, l'attaquant passif doit tout d'abord *désencapsuler* l'information de sa redondance avant de pouvoir effectuer une cryptanalyse de l'information chiffrée échangée. Dans le cadre de la stéganographie, cela revient à extraire l'information dissimulée. Cet objectif est le Saint Graal du stéganalyste. De manière plus pragmatique, nous devons d'abord être en mesure de distinguer les supports de couverture des stégo média dans la masse des média avant même imaginer une possible extraction. Les stéganalyses de la littérature se placent dans ce contexte ; c'est donc ce point de vue que nous avons naturellement adopté. Dans ce domaine, notre contribution a été multiple. Nous avons tout d'abord revisité les schémas classiques de sécurité en stéganographie afin de modéliser l'attaquant passif réel et proposons ainsi deux schémas de sécurité correspondants aux stéganalyses spécifique et universelle usuelles. Nous définissons alors formellement les notions d'attaque par discrimination, de distingueur, de stéganalyse et précisons la propriété d'universalité d'un schéma de stéganalyse. Nous relierons ensuite les modèles ainsi construits aux modèles classiques et montrons que des algorithmes qui ne sont pas sûrs dans les modèles que nous proposons, ne sont pas sûrs dans les modèles classiques. Finalement, nous exprimons l'insécurité des schémas de stéganographie en fonction des performances des distingueurs effectifs mis en évidence lors de stéganalyses pratiques. Ce sont ces modèles d'attaquant que nous mettons en œuvre pour évaluer les performances de nos stéganalyses. La centaine d'algorithmes de stéganographie (cf. annexe D) que nous avons recensée sur Internet est majoritairement dédiée aux images non compressées et JPEG ; c'est ce qui a motivé notre intérêt pour la stéganalyse dédiée aux images fixes. Nous avons dans un premier temps adapté une technique classique d'analyse développée par J. Fridrich *et al.* [76, 77], la stéganalyse RS, afin de détecter l'algorithme *Multi Bit Plane Image Steganography* (MBPIS), proposé par B.C. Nguyen *et al.* [147] à IWDW en 2006. Nous avons calibré un distingueur [18] qui détecte efficacement des images non compressées stéganographiées à l'aide de MBPIS pour des taux stéganographiques supérieurs à 3%. De plus, nous avons montré que les coefficients de la stéganalyse RS permettent de

concevoir un détecteur efficace contre MBPIS, ce qui affecte l'immunité de ce schéma contre la stéganalyse RS, annoncée par les auteurs. Dans un second temps, nous nous sommes intéressés à la stéganalyse dédiée au format JPEG. Nous avons développé une nouvelle approche [15, 16, 17] qui consiste à évaluer une déviation de l'entropie binaire dans le domaine fréquentiel compressé. À partir de cette approche, un schéma de stéganalyse universel [16, 17] a été conçu. Ce schéma permet de détecter des algorithmes inconnus avec des taux de détection élevés. D'autre part, nous avons aussi mis au point un schéma spécifique [15] contre les algorithmes *Outguess*, *F5* et *JPHide and JPSeek* dont les performances sont, là encore, très élevées. Ce qui différencie nos travaux des stéganalyses à l'état de l'art, ce sont tout d'abord des taux de détection quasi-indépendants du taux stéganographique mais aussi la possibilité de détecter à des taux où les autres stéganalyses échouent. Nos détecteurs sont capables de détecter un unique octet inséré dans une image fixe repoussant ainsi aux limites, le compromis ancestral entre l'indétectabilité et la capacité.

Une des premières leçons que nous enseigne cette étude, est qu'il est illusoire de croire qu'il suffit de ne pas utiliser des parties d'une image favorables à une technique de stéganalyse donnée pour s'en protéger. En effet, en excluant de telles zones, les statistiques les plus discriminantes sont ainsi « noyées » parmi les zones non utilisées. Néanmoins ce procédé ne gomme absolument pas les déviations observées mais tend plutôt à diminuer leur poids relatif en laissant invariant de nombreuses zones de l'image. Une simple exclusion de ces zones permet d'observer à nouveau ces déviations, comme l'illustre l'adaptation du chapitre 7. D'autre part, aux vues des techniques présentées au chapitre 8, le JPEG ne semble pas être un format de support adapté à la stéganographie. En effet, celui-ci présente trois domaines, spatial, fréquentiel et fréquentiel compressé qui sont fortement structurés mais aussi fortement corrélés. De ce point de vue, il paraît alors très difficile de préserver les statistiques dans les trois domaines simultanément, lors de l'insertion. De plus, les propriétés mises en évidence laissent présager que la plupart des algorithmes de stéganographie adaptés au format JPEG soient sensibles aux schémas de stéganalyse présentés. Ces propriétés, bijectivité et critère d'avalanche proche de 0.5, définissent une classe de fonctions prometteuse du point de vue du stéganalyste. À l'instar de l'étape de compression sans perte du format JPEG, elles doivent permettre d'élaborer des détecteurs dont les performances sont quasi-indépendantes du taux stéganographique et efficaces pour des taux extrêmement faibles. Rechercher de telles fonctions est alors une voie possible pour de futures stéganalyses. En prenant la solution dans l'autre sens, essayer d'adapter la fonction RLE+Huffman à d'autres types de données devrait révéler, de la même manière, des déviations statistiques jusqu'alors invisibles et obtenir ainsi des propriétés similaires pour de détecteurs travaillant avec d'autres supports que les coefficients DCT. La difficulté majeure que doit surmonter un algorithme de stéganographie est la préservation de toutes les statistiques possibles, c'est-à-dire avoir une distribution P_c de média de couverture identique à la distribution P_S des stégo média engendrés par l'algorithme. Or, les messages à dissimuler sont en général chiffrés et sont donc considérés comme aléatoires. Pour obtenir $P_c = P_S$, il faut donc trouver des données réellement aléatoires dans le format du support. Dans le cas des images fixes, les hypothèses faites sur le caractère aléatoire des LSB des pixels ou des coefficients DCT ne sont clairement pas avérées. En effet, les pixels de l'image, ou mêmes les coefficients DCT, sont fortement corrélés par la nature même de l'image. Introduire un message aléatoire dans leur LSB va tendre à les rendre plus aléatoires qu'ils ne le sont initialement. Pour concevoir des schémas de stéganographie pratiques et sûrs, deux approches s'offrent à nous. La plus naturelle, est de rechercher des

supports qui contiennent effectivement de l'aléa, ce qui est loin d'être évident, ou bien de partir de schémas prouvés sûrs et de dégrader leur sécurité, tout en la maîtrisant, afin de les rendre utilisables en pratique.

Conclusion

*« Curiosité n'est que vanité. Le plus souvent,
on veut savoir que pour en parler. »*

Blaise Pascal (Pensées sur la religion)

N OUS avons analysé dans cette thèse des canaux de communication dans un contexte non coopératif sous deux angles distincts ; sous l'angle des codes correcteurs d'erreurs, d'une part, et sous l'angle de la stéganographie d'autre part. L'objectif commun est de proposer des techniques permettant à un adversaire d'avoir accès au message chiffré transmis par un système de communication, en vue d'une cryptanalyse et d'évaluer ainsi les conditions réelles d'un attaquant cryptographique. Dans le cadre des canaux de communication classiques, l'étape précédant la cryptanalyse consiste à enlever les bits de redondance ajoutés au message chiffré et à corriger d'éventuelles erreurs introduites par le canal de transmission. Dans un contexte un peu différent, une étape de stéganographie, consistant à dissimuler le message chiffré dans un support et à transmettre ce dernier, vient s'intercaler entre le chiffrement et le codage de canal. Si l'on considère le support comme un canal de transmission à part entière, nous nous ramenons donc à l'analyse d'un canal non coopératif, dans le même esprit que celle effectuée pour la reconstruction des codes correcteurs d'erreurs. Néanmoins, le travail de l'attaquant est quelque peu différent. En effet, dans ce contexte, celui-ci doit d'abord détecter les média qui contiennent de l'information cachée puis, dans un deuxième temps, extraire cette information.

De le cadre de nos travaux sur la reconstruction des codes correcteurs d'erreurs, nous nous sommes inspirés du formalisme proposé par G.D. Forney [87] et repris par R.J. McEliece [143], afin d'unifier sous une même approche, l'ensemble des algorithmes de reconstruction étudiés. Nous nous sommes notamment intéressés à la reconstruction des codes en blocs linéaires [19] en effectuant l'analyse de l'algorithme de Sicot-Houcke [175] dans le formalisme adopté. Cet algorithme constitue la brique de base pour reconstruire les codes convolutifs. Dans l'esprit des travaux d'É. Filiol [69, 70, 71], nous avons mis en évidence de nouvelles relations qui nous ont permis d'obtenir un algorithme de meilleure complexité théorique [9], d'automatiser complètement le processus de reconstruction mais aussi de démontrer la conjecture d'É. Filiol, [71, p. 170], sur la nature des codeurs renvoyés par son algorithme. Enfin, en concevant un algorithme de reconstruction d'entrelaceur [14], nous avons pu ainsi généraliser ces techniques à la reconstruction des turbo-codes [9]. Nous montrons, dans ce cas précis, qu'au-delà de la reconstruction du code, il est possible de trouver un turbo-codeur équivalent. Les aspects théoriques ont été corroborés par des

simulations dont les paramètres des codeurs ont été extraits de normes des systèmes de communication actuels.

Notre première contribution en stéganalyse a été de définir de nouveaux modèles de sécurité, afin de formaliser les notions de stéganalyse par discrimination spécifique et universelle, ainsi que les adversaires réels mettant en œuvre de tels schémas. Nous nous sommes ensuite focalisés sur l'analyse stéganographique d'images fixes, au format non compressé et JPEG ; les images fixes de ces formats étant les média les plus répandus sur Internet. Nous avons proposé une adaptation de la stéganalyse RS de J. Fridrich *et al.* [77], en une stéganalyse RS locale, afin de détecter l'algorithme *Multi Bit Plane Image Steganography* [147], spécifié par B.C. Nguyen *et al.* à IWDW'06. Nous avons ainsi calibré des détecteurs à partir des coefficients de la stéganalyse RS locale, d'une part, et RS classique, d'autre part. Les simulations montrent clairement [18] que les détecteurs issus de la stéganalyse RS locale sont plus performants et détectent, avec une probabilité proche de 1, l'algorithme MBPIS pour des taux stéganographiques supérieurs à 5%. Par ailleurs, nous avons élaboré une nouvelle approche pour la stéganalyse dédiée au format JPEG. Celle-ci consiste à étudier une déviation statistique de l'entropie binaire des coefficients DCT quantifiés et codés par RLE puis Huffman, dans le domaine fréquentiel compressé. À partir de cette déviation, nous avons spécifié deux schémas de stéganalyse. Le premier est un schéma de stéganalyse universelle [16, 17] qui est capable de détecter des algorithmes de stéganographie potentiellement inconnus. Le second, est un schéma de stéganalyse spécifique [15], testé avec *Outguess*, *F5* et *JPHide and JPSeek*. Ces deux schémas possèdent des performances qui sont quasi-indépendantes de la quantité d'information dissimulée. De ce fait, ils sont capables de détecter une image stéganographiée dès lors qu'un seul octet lui a été inséré, cela, avec des taux de détection proches de 1. Cette propriété étonnante et quelque peu contre-intuitive, a été expliquée théoriquement et validée expérimentalement. Les simulations ont aussi mis en évidence l'invariance de cette caractéristique par changement de la base d'images de tests. De plus, celles-ci ont été effectuées pour des taux stéganographiques variants de 10^{-6} à 10^{-2} , dans un intervalle où les schémas de stéganalyse à l'état de l'art ne fonctionnent pas.

Aux vues des résultats que nous avons obtenus, il apparaît alors que les conditions de l'attaquant réel ne sont pas optimales et semblent même assez éloignées des hypothèses traditionnelles faites sur l'adversaire cryptographique. Tout d'abord, les algorithmes de reconstruction présentent des limites intrinsèques. En effet, les techniques détaillées aux chapitres 2 et 3 ne reconstruisent que les codes et non les décodeurs. Dans ces conditions décoder et retrouver le message chiffré est un problème NP-complet [29]. Pour lever ces indéterminées théoriques, il est nécessaire de faire des hypothèses supplémentaires, le plus souvent « aux vues du terrain ». Les futurs algorithmes de reconstruction devront alors prendre en compte ces hypothèses par l'ajout de nouveaux paramètres. D'autre part, des limites pratiques ont été mises en évidence. Les performances des algorithmes de reconstruction chutent exponentiellement avec l'erreur du canal et la longueur du code. Un schéma de codage de canal composé d'un code correcteur d'erreurs suivi d'un entrelaceur nous apparaît alors bien plus difficile à reconstruire, en pratique, qu'un code correcteur seul. En effet, la présence d'un entrelaceur favorise l'uniformisation de la répartition de l'erreur sur le train binaire. Une telle répartition de l'erreur simule l'action d'un canal binaire symétrique, qui est un canal défavorable pour la reconstruction. De plus, reconstruire un tel schéma est équivalent à reconstruire un code de longueur la profondeur de

l'entrelaceur, qui est en pratique très grande. *A contrario*, les travaux de M. Cluzeau [51] ouvrent une voie prometteuse. En effet, corriger l'erreur au fur et à mesure du processus de reconstruction permet de tolérer des niveaux de bruit beaucoup plus importants. Une autre voie, encore inexploitée à ce jour, pour faire diminuer artificiellement le taux d'erreur binaire, est de prendre en compte l'information souple en sortie de la démodulation. Moyennant une métrique adaptée, il peut être possible de sélectionner les lignes de la matrice d'interception les moins bruitées pour la recherche des mots de poids faible.

Dans l'hypothèse où un schéma de stéganographie est couplé au chiffrement du message, l'adversaire semble démuni. En effet, même si nous avons conçu des détecteurs très performants, l'extraction en aveugle de l'information dissimulée est équivalente à la recherche exhaustive sur la clé stéganographique qui a servi à choisir aléatoirement les positions contenant le message. L'adversaire ne dispose que du support, il doit donc retrouver la clé sans même connaître la sortie du générateur pseudo-aléatoire. Néanmoins, la sécurité d'un schéma de stéganographie s'évalue en fonction de sa détectabilité. Les propriétés que nous avons mises en évidence laissent penser que la majorité des algorithmes de stéganographie dédiés au format JPEG sont sensibles à la variation d'entropie binaire que nous avons observée. Le format JPEG ne paraît pas être un format adapté pour la stéganographie, notamment parce qu'il possède une structure très forte et agit dans trois domaines corrélés entre eux. Les taux de détection élevés et indépendants, en pratique, du taux stéganographique, nous poussent à essayer d'adapter l'étape de compression sans perte du format JPEG afin de concevoir des détecteurs adaptés à d'autres types de média. De même, il semble judicieux de coupler nos détecteurs avec ceux issus des stéganalyses classiques en espérant une certaine orthogonalisation de leur pouvoir de discrimination. D'autre part, une certaine contradiction nous a marqués lors de cette étude. La théorie nous indique clairement qu'un schéma de stéganographie est indétectable si et seulement si la distribution P_S des stégo média engendrés par le schéma est identique à la distribution P_C des supports de couverture (cf. chapitre 6). Or, tous les algorithmes de stéganographie que nous avons rencontrés dissimulent un message dont les propriétés statistiques sont proches de celles de l'aléa, dans des données qui ne sont visiblement pas aléatoires (LSB des pixels ou des coefficients DCT). Le véritable challenge du concepteur est peut-être de trouver des données aléatoires dans des supports « naturels » et donc fortement structurés.

Annexe A

Encadrement de $\prod_{i=0}^{n-1} (1 - 2^{i-M})$

Pour prouver la proposition 2.7 page 71, nous allons démontrer la proposition suivante.

Proposition A.1 *Quels que soient $M \geq n \geq 1$,*

$$P(n) : \quad 1 - \sum_{i=0}^{n-1} 2^{i-M} \leq \prod_{i=0}^{n-1} (1 - 2^{i-M}) \leq 1 - \sum_{i=0}^{n-1} \left(2^{i-M} - \sum_{j=i+1}^{n-1} 2^{i-M} 2^{j-M} \right).$$

Preuve : la démonstration se fera par récurrence. Notons

$$\begin{aligned} u_n &= \prod_{i=0}^{n-1} (1 - 2^{i-M}), \\ v_n &= 1 - \sum_{i=0}^{n-1} 2^{i-M}, \\ w_n &= 1 - \sum_{i=0}^{n-1} \left(2^{i-M} - \sum_{j=0}^{n-1} 2^{i-M} 2^{j-M} \right), \\ &= 1 - \sum_{i=0}^{n-1} 2^{i-M} \left(1 - \sum_{j=i+1}^{n-1} 2^{j-M} \right). \end{aligned}$$

Nous avons $v_1 = u_1 = w_1 = 1 - 2^{-M}$, (P_1) est vérifiée.

Supposons (P_n) vraie. Tout d'abord, évaluons

$$\begin{aligned}
 w_{n+1} - w_n &= 1 - \sum_{i=0}^n 2^{i-M} \left(1 - \sum_{j=i+1}^n 2^{j-M} \right) - w_n, \\
 &= - \sum_{i=0}^{n-1} 2^{i-M} \left(1 - \sum_{j=i+1}^{n-1} 2^{j-M} - 2^{n-M} \right) + 1 - 2^{n-M} - w_n, \\
 &= 2^{n-M} \sum_{i=0}^{n-1} 2^{i-M} - 2^{n-M} + 1 - \sum_{i=0}^{n-1} 2^{i-M} \left(1 - \sum_{j=i+1}^{n-1} 2^{j-M} \right) - w_n, \\
 &= 2^{n-M} \left(\sum_{i=0}^{n-1} 2^{i-M} - 1 \right). \blacksquare
 \end{aligned}$$

Nous pouvons maintenant majorer u_{n+1} ,

$$\begin{aligned}
 u_{n+1} &= (1 - 2^{n-M})u_n \leq (1 - 2^{n-M})w_n \\
 &\leq w_n - 2^{n-M} \left(1 - \sum_{i=0}^{n-1} 2^{i-M} \left(1 - \sum_{j=i+1}^{n-1} 2^{j-M} \right) \right), \\
 &\leq w_n - 2^{n-M} + 2^{n-M} \sum_{i=0}^{n-1} 2^{i-M} \left(1 - \sum_{j=i+1}^{n-1} 2^{j-M} \right), \\
 &\leq w_n + 2^{n-M} \left(\sum_{i=0}^{n-1} 2^{i-M} - 1 \right) - 2^{n-M} \sum_{i=0}^{n-1} 2^{i-M} \sum_{j=i+1}^{n-1} 2^{j-M}, \\
 &\leq w_{n+1} - 2^{n-M} \sum_{i=0}^{n-1} 2^{i-M} \sum_{j=i+1}^{n-1} 2^{j-M}, \\
 &\leq w_{n+1}. \blacksquare
 \end{aligned}$$

Une minoration de u_{n+1} est obtenue de la même façon,

$$\begin{aligned}
 u_{n+1} &= (1 - 2^{n-M})u_n \geq (1 - 2^{n-M})v_n \\
 &\geq (1 - 2^{n-M}) \left(1 - \sum_{i=0}^{n-1} 2^{i-M} \right), \\
 &\geq 1 - \left(\sum_{i=0}^{n-1} 2^{i-M} + 2^{n-M} \right) + 2^{n-M} \sum_{i=0}^{n-1} 2^{i-M}, \\
 &\geq v_{n+1} + 2^{n-M} \sum_{i=0}^{n-1} 2^{i-M}, \\
 &\geq v_{n+1}. \blacksquare
 \end{aligned}$$

Finalement, (P_{n+1}) est vraie, ce qui implique que (P_n) le soit pour tout $n \geq 1$. $\blacksquare$

Nous allons maintenant en déduire un encadrement de $1 - \prod_{i=0}^{n-1} (1 - 2^{i-M})$.

$$\begin{aligned}
1 - u_n &\leq \sum_{i=0}^{n-1} 2^{i-M} = 2^{-M}(2^n - 1) \text{ et} \\
1 - u_n &\geq \sum_{i=0}^{n-1} \left(2^{i-M} - \sum_{j=i+1}^{n-1} 2^{i-M} 2^{j-M} \right), \\
&\geq 2^{-M} \sum_{i=0}^{n-1} 2^i - 2^{-2M} \sum_{i=0}^{n-1} 2^i \sum_{j=i+1}^{n-1} 2^j, \\
&\geq 2^{-M}(2^n - 1) - 2^{-2M} \sum_{i=0}^{n-1} 2^i \sum_{j=0}^{n-1} 2^j, \\
&\geq 2^{-M}(2^n - 1) - (2^{-M}(2^n - 1))^2.
\end{aligned}$$

Nous obtenons donc

$$2^{-M}(2^n - 1)(1 - 2^{-M}(2^n - 1)) \leq 1 - \prod_{i=0}^{n-1} (1 - 2^{i-M}) \leq 2^{-M}(2^n - 1).$$

En posant $M = m$ et $n = n_e$ nous en déduisons aisément une minoration nécessaire à la preuve de la proposition 2.7 page 71. ■

Quelques standards et normes

Dans cette annexe, nous passons succinctement en revue les différents codes correcteurs et longueurs d'entrelaceurs utilisés dans les normes et standards de télécommunication les plus répandus. Ceux-ci nous ont permis de valider expérimentalement nos algorithmes de reconstruction avec des codes correcteurs d'erreurs utilisés dans des systèmes de communication existants. Nous les avons regroupés selon quatre grandes familles,

- téléphonie mobile terrestre,
- transmissions satellites,
- réseaux sans fils,
- Télévision Numérique Terrestre.

Dans le contexte de notre étude, nous nous limitons à la description des codeurs linéaires en blocs, de codeurs convolutifs, des turbo-codeurs parallèles ainsi qu'aux entrelaceurs en blocs. Les codeurs convolutifs sont donnés sous forme polynomiale et octale. D'autre part, les codes en blocs qui apparaissent dans ces normes sont majoritairement des *codes cycliques*. Nous en rappelons ici la définition et quelques propriétés.

Définition B.1 *Un code linéaire $\mathcal{C} \subset \mathbb{F}_2^n$ est dit cyclique quand il est stable par l'automorphisme de décalage cyclique*

$$\begin{aligned} T : \mathbb{F}_2^n &\longrightarrow \mathbb{F}_2^n \\ (x(1), \dots, x(n)) &\longrightarrow (x(2), \dots, x(n), x(1)). \end{aligned}$$

i.e. $\forall x \in \mathcal{C}, T(x) \in \mathcal{C}$.

Traditionnellement, les codes cycliques sont représentés dans l'algèbre $\mathbb{F}_2[D]/(D^n - 1)$ par l'application

$$\begin{aligned} \mathbb{F}_2^n &\longrightarrow \mathbb{F}_2[D]/(D^n - 1) \\ (x(1), \dots, x(n)) &\longrightarrow x(n)D^{n-1} + \dots + x(2)D + x(1). \end{aligned}$$

L'endomorphisme de décalage cyclique est alors associé à la multiplication par D dans $\mathbb{F}_2[D]/(D^n - 1)$. Un code cyclique est alors entièrement défini par la donnée d'un polynôme g , diviseur unitaire de $D^n - 1$. g est appelé *polynôme générateur* de $\mathcal{C}$ et $\mathcal{C}$ est engendré

par $g, Dg, \dots, D^{n-1-\deg g}g$. Si $g = g_0 + g_1D + \dots + g_rD^r$, $\mathcal{C}$ est de dimension $k = n - r$ et

$$G = \begin{pmatrix} g_0 & g_1 & \dots & g_r & 0 & \dots & 0 \\ 0 & g_0 & \dots & g_{r-1} & g_r & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & g_0 & g_1 & \dots & g_{r-1} & g_r \end{pmatrix},$$

est une de ses matrices génératrices. Dans la suite, les codes cycliques seront donnés par leur polynôme générateur.

B.1 Téléphonie mobile terrestre

Dans le cadre de la téléphonie mobile terrestre, nous nous sommes concentrés sur les standards européens 2G et 3G (GSM et UMTS) ainsi que sur leurs homologues américains (IS95 et CDMA 2000).

B.1.1 Le GSM

Pour plus de détails, le lecteur pourra se référer à [191]. Les différentes combinaisons proposées sont les suivantes :

- Code cyclique + code convolutif + entrelaceur,
- code convolutif + entrelaceur,
- code de Reed-Solomon + entrelaceurs.

Les codeurs cycliques sont sous forme systématique et sont générés à partir des polynômes suivants :

$$\begin{aligned} g_{c_0} &= 1 + D + D^3, \\ g_{c_1} &= 1 + D^2 + D^3 + D^5 + D^6, \\ g_{c_2} &= 1 + D^2 + D^3 + D^4 + D^8, \\ g_{c_3} &= 1 + D^2 + D^3 + D^5 + D^{13} + D^{14}, \\ g_{FireCode} &= (1 + D^{23})(1 + D^3 + D^{17}). \end{aligned}$$

Les codeurs convolutifs sont construits à partir des huit polynômes générateurs suivants :

$$\begin{aligned} g_{0conv} &= 1 + D^3 + D^4 = (23), \\ g_{1conv} &= 1 + D + D^3 + D^4 = (33), \\ g_{2conv} &= 1 + D^2 + D^4 = (25), \\ g_{3conv} &= 1 + D + D^2 + D^3 + D^4 = (37), \\ g_{4conv} &= 1 + D^2 + D^3 + D^5 + D^6 = (133), \\ g_{5conv} &= 1 + D + D^4 + D^6 = (145), \\ g_{6conv} &= 1 + D + D^2 + D^3 + D^4 + D^6 = (175), \\ g_{7conv} &= 1 + D + D^2 + D^3 + D^6 = (171). \end{aligned}$$

Six codeurs convolutifs non systématiques, non récurrents sont spécifiés.

$$G_{0conv} = \begin{bmatrix} g_{0conv} \\ g_{1conv} \end{bmatrix}, \quad G_{1conv} = \begin{bmatrix} g_{4conv} \\ g_{7conv} \end{bmatrix}, \quad G_{2conv} = \begin{bmatrix} g_{1conv} \\ g_{2conv} \\ g_{3conv} \end{bmatrix}, \quad G_{3conv} = \begin{bmatrix} g_{4conv} \\ g_{5conv} \\ g_{6conv} \end{bmatrix},$$

$$G_{4conv} = \begin{bmatrix} g_{4conv} \\ g_{7conv} \\ g_{5conv} \end{bmatrix}, \quad G_{5conv} = \begin{bmatrix} g_{1conv} \\ g_{2conv} \\ g_{3conv} \\ g_{1conv} \\ g_{2conv} \\ g_{3conv} \end{bmatrix}.$$

De même, sept codeurs récurrents systématiques sont donnés.

$$G_{6conv} = \begin{bmatrix} 1 \\ \frac{g_{1conv}}{g_{0conv}} \end{bmatrix}, \quad G_{7conv} = \begin{bmatrix} \frac{g_{1conv}}{g_{3conv}} \\ \frac{g_{2conv}}{g_{3conv}} \\ \frac{g_{3conv}}{1} \end{bmatrix}, \quad G_{8conv} = \begin{bmatrix} 1 \\ \frac{g_{5conv}}{g_{4conv}} \\ \frac{g_{6conv}}{g_{4conv}} \end{bmatrix}, \quad G_{9conv} = \begin{bmatrix} \frac{g_{1conv}}{g_{3conv}} \\ \frac{g_{2conv}}{g_{3conv}} \\ g_{3conv} \\ 1 \\ 1 \end{bmatrix},$$

$$G_{10conv} = \begin{bmatrix} \frac{g_{4conv}}{g_{6conv}} \\ \frac{g_{5conv}}{g_{6conv}} \\ g_{6conv} \\ 1 \\ 1 \end{bmatrix}, \quad G_{11conv} = \begin{bmatrix} \frac{g_{1conv}}{g_{3conv}} \\ \frac{g_{2conv}}{g_{3conv}} \\ \frac{g_{3conv}}{g_{3conv}} \\ 1 \\ 1 \end{bmatrix}, \quad G_{12conv} = \begin{bmatrix} \frac{g_{4conv}}{g_{6conv}} \\ \frac{g_{5conv}}{g_{6conv}} \\ \frac{g_{6conv}}{g_{6conv}} \\ 1 \\ 1 \end{bmatrix}.$$

Les entrelaceurs blocs sont de longueur $l_e \in \{228, 456, 1368, 1392\}$.

B.1.2 L'UMTS

Les paramètres des codeurs sont issus des normes [183, 184, 185, 186]. Les différentes combinaisons proposées sont les suivantes :

- Code cyclique + code convolutif + 2 entrelaceurs,
- code cyclique + turbo-code + 2 entrelaceurs.

Les codeurs cycliques sont sous forme systématique et engendrés à partir des polynômes suivants :

$$\begin{aligned} g_{c_0} &= 1 + D + D^3 + D^4 + D^7 + D^8, \\ g_{c_1} &= 1 + D + D^2 + D^3 + D^{11} + D^{12}, \\ g_{c_2} &= 1 + D^5 + D^{12} + D^{16}, \\ g_{c_3} &= 1 + D + D^5 + D^6 + D^{23} + D^{24}. \end{aligned}$$

Deux codeurs convolutifs non récurrents, non systématiques sont spécifiés.

$$G_{0conv} = \begin{bmatrix} 1 + D^2 + D^3 + D^4 + D^8 \\ 1 + D + D^2 + D^3 + D^4 + D^7 + D^8 \end{bmatrix} = \begin{bmatrix} 561 \\ 753 \end{bmatrix},$$

$$G_{1conv} = \begin{bmatrix} 1 + D^2 + D^3 + D^5 + D^6 + D^7 + D^8 \\ 1 + D + D^3 + D^4 + D^7 + D^8 \\ 1 + D + D^2 + D^5 + D^8 \end{bmatrix} = \begin{bmatrix} 557 \\ 663 \\ 711 \end{bmatrix}.$$

Le turbo-code est parallèle et ses codeurs convolutifs internes sont identiques et donnés par

$$G_{2conv} = \begin{bmatrix} 1 \\ \frac{g_{1conv}}{g_{0conv}} \end{bmatrix},$$

avec

$$\begin{aligned} g_{0conv} &= 1 + D + D^3 = 13, \\ g_{1conv} &= 1 + D^2 + D^3 = 15. \end{aligned}$$

La partie systématique du deuxième codeur est supprimée. Son entrelaceur interne est un entrelaceur par blocs dont la taille l_i est variable et telle que $40 \leq l_i \leq 5114$.

Les entrelaceurs sont des entrelaceurs par blocs dont la taille l_e est variable et peut aller jusqu'à 153 600 pour le premier et 19 200 pour le second.

B.1.3 IS95

Les paramètres sont issus de [127, 187]. Le schéma est constitué d'une paire de codeurs convolutifs et entrelaceurs. Les codeurs convolutifs sont identiques à ceux de l'UMTS.

$$\begin{aligned} G_{0conv} &= \begin{bmatrix} 1 + D^2 + D^3 + D^4 + D^8 \\ 1 + D + D^3 + D^4 + D^7 + D^8 \end{bmatrix} = \begin{bmatrix} 561 \\ 753 \end{bmatrix}, \\ G_{1conv} &= \begin{bmatrix} 1 + D^2 + D^3 + D^5 + D^6 + D^7 + D^8 \\ 1 + D + D^3 + D^4 + D^7 + D^8 \\ 1 + D + D^2 + D^5 + D^8 \end{bmatrix} = \begin{bmatrix} 557 \\ 663 \\ 711 \end{bmatrix}. \end{aligned}$$

La longueur des entrelaceurs l_e est prise dans $\{128, 384, 576\}$.

B.1.4 CDMA 2000

Les paramètres des codeurs et entrelaceurs sont issus des spécifications [188, 189, 190]. Deux types d'association sont possibles.

- Code cyclique + code convolutif + entrelaceur,
- code cyclique + turbo-code + entrelaceur.

Les codeurs cycliques sont sous forme systématique et générés avec les polynômes suivants :

$$\begin{aligned} g_{c0} &= 1 + D + D^2 + D^6, \\ g_{c1} &= 1 + D + D^2 + D^5 + D^6, \\ g_{c2} &= 1 + D + D^3 + D^4 + D^7 + D^8, \\ g_{c3} &= 1 + D + D^3 + D^4 + D^6 + D^7 + D^8 + D^9 + D^{10}, \\ g_{c4} &= 1 + D + D^4 + D^8 + D^9 + D^{10} + D^{11} + D^{12}, \\ g_{c5} &= 1 + D + D^2 + D^5 + D^6 + D^{11} + D^{14} + D^{15} + D^{16}. \end{aligned}$$

Les codeurs convolutifs sont non récurrents et non systématiques.

$$\begin{aligned}
 G_{0conv} &= \begin{bmatrix} 1 + D^2 + D^3 + D^4 + D^8 \\ 1 + D + D^2 + D^3 + D^4 + D^7 + D^8 \end{bmatrix} = \begin{bmatrix} 561 \\ 753 \end{bmatrix}, \\
 G_{1conv} &= \begin{bmatrix} 1 + D^2 + D^3 + D^5 + D^6 + D^7 + D^8 \\ 1 + D + D^3 + D^4 + D^7 + D^8 \\ 1 + D + D^2 + D^5 + D^8 \end{bmatrix} = \begin{bmatrix} 557 \\ 663 \\ 711 \end{bmatrix}, \\
 G_{2conv} &= \begin{bmatrix} 1 + D + D^2 + D^3 + D^4 + D^6 + D^8 \\ 1 + D + D^3 + D^4 + D^5 + D^8 \\ 1 + D^2 + D^5 + D^7 + D^8 \\ 1 + D^3 + D^4 + D^5 + D^7 + D^8 \end{bmatrix} = \begin{bmatrix} 765 \\ 671 \\ 513 \\ 473 \end{bmatrix}.
 \end{aligned}$$

Le turbo-code est parallèle et ses codeurs convolutifs internes sont identiques et donnés par

$$G_{3conv} = \begin{bmatrix} 1 \\ g_{1conv} \\ g_{0conv} \\ g_{2conv} \\ g_{0conv} \end{bmatrix},$$

avec

$$\begin{aligned}
 g_{0conv} &= 1 + D + D^3 = 13, \\
 g_{1conv} &= 1 + D^2 + D^3 = 15, \\
 g_{2conv} &= 1 + D + D^2 + D^3 = 17.
 \end{aligned}$$

La partie systématique du deuxième codeur est supprimée. Son entrelaceur interne est un entrelaceur par blocs dont la taille l_i est variable et telle que $378 \leq l_i \leq 20\,730$.

La taille l_e de l'entrelaceur est variable ; $384 \leq l_e \leq 36\,864$.

B.2 Transmissions satellites

B.2.1 INMARSAT

Les paramètres des standards INMARSAT (INMARSAT-A, INMARSAT-B, INMARSAT-C, INMARSAT-M, INMARSAT-aero) ont été repris de [150]. Le schéma de codage proposé est composé d'un code cyclique, d'un code convolutif et d'un entrelaceur (pour les versions C, M et aero).

$$G_{conv} = \begin{bmatrix} 1 + D + D^2 + D^3 + D^6 \\ 1 + D^2 + D^3 + D^5 + D^6 \end{bmatrix} = \begin{bmatrix} 171 \\ 133 \end{bmatrix}.$$

L'entrelaceur possède une taille $l_e = 10\,368$.

B.2.2 INTELSAT

Les paramètres du schéma de codage sont issus des standards publiés par l'Intelsat Earth Station Standards [106, 107]. Les deux principaux schémas sont les suivants :

- Codeur convolutif + entrelaceur,
- Reed-Salomon + entrelaceur en hélice + codeur convolutif.

Le codeur convolutif est défini par

$$G_{conv} = \begin{bmatrix} 1 + D + D^2 + D^3 + D^6 \\ 1 + D^2 + D^3 + D^5 + D^6 \end{bmatrix} = \begin{bmatrix} 171 \\ 133 \end{bmatrix}.$$

B.2.3 GLOBALSTAR

GLOBALSTAR est une constellation de satellite qui s'appuie essentiellement sur les standards IS95 [187, 165]. Le schéma de codage est uniquement constitué d'un code convolutif et d'un entrelaceur bloc.

$$G_{0conv} = \begin{bmatrix} 1 + D^2 + D^3 + D^4 + D^8 \\ 1 + D + D^2 + D^3 + D^4 + D^7 + D^8 \end{bmatrix} = \begin{bmatrix} 561 \\ 753 \end{bmatrix},$$

L'entrelaceur possède une taille $l_e \in \{96, 192, 384\}$.

B.2.4 DVB-S

Les schémas de codage proposés dans le standards sont les mêmes que pour le DVB-T et sont répertoriés dans [63, 61]. Cinq schémas de codage sont proposés.

- Code Reed-Solomon + entrelaceur convolutif + code convolutif,
- turbo-code + entrelaceur,
- code convolutif + entrelaceur,
- turbo-code + entrelaceur + code convolutif,
- code Reed-Solomon + entrelaceur convolutif + code convolutif + entrelaceur.

Le seul codeur convolutif utilisé est non systématique et non récursif.

$$G_{conv} = \begin{bmatrix} 1 + D + D^2 + D^3 + D^6 \\ 1 + D^2 + D^3 + D^5 + D^6 \end{bmatrix} = \begin{bmatrix} 171 \\ 133 \end{bmatrix}.$$

Les entrelaceurs blocs possèdent une taille variable et sont générés à partir de séquences de longueur 1023.

Le turbo-code est parallèle de rendement $r = \frac{1}{3}$. Il est composé de deux codes convolutifs de rendement $r = \frac{1}{2}$, de longueur de contrainte $K = 4$ et dont la partie systématique du deuxième codeur a été supprimée. Son entrelaceur interne est un entrelaceur blocs de taille l_i variable telle que $96 \leq l_i \leq 1504$.

B.3 Réseaux sans fils

B.3.1 La norme 802.11a et HYPERLAN-2

Ces deux standards sont très similaires. Ils proposent un seul schéma de codage composé d'un code convolutif et d'un entrelaceur bloc. Ces paramètres sont détaillés dans [65, 104, 146].

$$G_{conv} = \begin{bmatrix} 1 + D + D^2 + D^3 + D^6 \\ 1 + D^2 + D^3 + D^5 + D^6 \end{bmatrix} = \begin{bmatrix} 171 \\ 133 \end{bmatrix}.$$

Les entrelaceurs blocs possèdent une taille $l_e \in \{48, 96, 192, 288\}$.

B.4 Télévision Numérique Terrestre

La TNT est basée sur le standard DVB-T [60, 62]. Cinq schémas de codage sont proposés.

- Code Reed-Solomon + entrelaceur convolutif + code convolutif,
- turbo-code + entrelaceur,
- code convolutif + entrelaceur,
- turbo-code + entrelaceur + code convolutif,
- code Reed-Solomon + entrelaceur convolutif + code convolutif + entrelaceur.

Le seul codeur convolutif utilisé est non systématique et non récursif.

$$G_{conv} = \begin{bmatrix} 1 + D + D^2 + D^3 + D^6 \\ 1 + D^2 + D^3 + D^5 + D^6 \end{bmatrix} = \begin{bmatrix} 171 \\ 133 \end{bmatrix}.$$

Les entrelaceurs blocs possèdent une taille variable et sont générés à partir de séquences de longueur 1023.

Le turbo-code est parallèle de rendement $r = \frac{1}{4}$. Il est composé de deux codes convolutifs de rendement $r = \frac{2}{3}$, de longueur de contrainte $K = 4$ et dont la partie systématique du deuxième codeur a été supprimée. Son entrelaceur interne est un entrelaceur blocs de taille l_i variable telle que $144 \leq l_i \leq 648$.

Hauteur moyenne d'un trie dynamique

Dans cette annexe, nous démontrons le théorème 4.2 page 126 qui donne l'expression de la hauteur moyenne d'un trie. Pour ce faire, nous avons besoin d'introduire la transformée de Mellin. Cette brève introduction est extraite de [52]. Cette démonstration s'articule en trois parties. Tout d'abord, nous considérons le nombre N des mots infinis du trie comme une variable aléatoire discrète à valeurs dans $\mathbb{N}$, en utilisant une transformée de Poisson. Nous exprimons ensuite la probabilité $p_{k,n}$ qu'un trie soit de hauteur donnée. Dans un deuxième temps, nous considérons à nouveau N fixé et évaluons la hauteur moyenne d'un trie à partir de $p_{k,n}$. Nous l'approximons ensuite par une série double exponentielle. Enfin, la dernière étape nous donne un équivalent asymptotique de cette hauteur en appliquant la transformée de Mellin.

C.1 Transformée de Mellin

Notation : dans la suite, la notation $\langle \alpha, \beta \rangle$ désigne la bande ouverte du plan complexe $\{s \in \mathbb{C} \mid \Re(s) \in]\alpha, \beta[\}$.

Définition C.1 Soit $f :]0, +\infty[\rightarrow \mathbb{R}$ une fonction localement sommable sur $]0, +\infty[$ (c'est-à-dire sur tout compact de $]0, +\infty[$), la transformée de Mellin de f , notée $f^*(s)$, est définie par

$$f^*(s) = \int_0^\infty f(x) x^{s-1} dx.$$

On appelle bande fondamentale la plus grande bande complexe $\langle \alpha, \beta \rangle$ sur laquelle l'intégrale converge.

Il existe une correspondance entre les développements asymptotiques d'une fonction f en 0 (resp. $+\infty$) et les pôles de sa transformée de Mellin f^* dans un demi-plan gauche (resp. droit) par rapport à la bande fondamentale. À chaque terme du développement asymptotique de f de la forme $x^c(\log x)^k$ correspond un pôle d'ordre $k+1$ de sa transformée f^* en $s = -c$.

Nous rappelons qu'une *fonction holomorphe* est une fonction à valeurs dans $\mathbb{C}$, définie et dérivable en tout point d'un sous-ensemble ouvert de $\mathbb{C}$. De même, une *fonction méromorphe* est une fonction holomorphe sur $\mathcal{C}$ sauf en un certain nombre de points singuliers

appelés *pôles*. Soit Φ une fonction méromorphe en $s = s_0$. Cette fonction se développe en série de Laurent au voisinage de s_0 par

$$\Phi(s) = \sum_{k=-\infty}^{+\infty} c_k (s - s_0)^k.$$

La fonction $\Phi(s)$ possède un pôle d'ordre r si $r > 0$ et $c_{-r} \neq 0$ et est holomorphe en s_0 si $c_k = 0$ pour tout $k < 0$. On définit la *partie singulière* de Φ en $s = s_0$ par

$$\sum_{k=-\infty}^{-1} c_k (s - s_0)^k.$$

Définition C.2 Soit Φ fonction méromorphe sur Ω et $\mathcal{S} \subseteq \Omega$ l'ensemble des pôles de Φ dans Ω . La partie singulière de Φ sur Ω est la somme formelle des parties singulières de Φ sur chacun des points de $\mathcal{S}$. De plus, si E est la partie singulière de Φ sur Ω , nous notons

$$\Phi(s) \asymp E \quad (s \in \Omega).$$

Le théorème suivant permet d'expliciter la correspondance entre le développement asymptotique en $+\infty$ de f et sa transformée de Mellin.

Théorème C.1 Soit f continue sur $]0, +\infty[$ de transformée de Mellin $f^*(s)$ définie sur une bande non vide, $\langle \alpha, \beta \rangle$. On suppose que

1. f admet un prolongement méromorphe sur une bande $\langle \gamma, \beta \rangle$ avec $\gamma > \beta$ et est analytique sur $\Re(s) = \gamma$,
2. il existe un entier $r > 1$, un nombre réel $\eta \in]\alpha, \beta[$ et une suite strictement croissante de réels $(T_j)_{j \in \mathbb{N}}$ tendant vers l'infini, tels que

$$f^*(s) = O(|s|^{-r}),$$

sur la réunion des segments $\{s \in \mathbb{C} \mid \Re(s) \in [\eta, \gamma], \text{Im}(s) = T_j\}$, quand $j \rightarrow +\infty$.

Si $f^*(s)$ admet comme partie singulière

$$f^*(s) \asymp \sum_{(\xi, k) \in A} d_{\xi, k} \frac{1}{(s - \xi)^k},$$

Alors le développement de $f(x)$ en $+\infty$

$$f(x) = \sum_{(\xi, k) \in A} d_{\xi, k} \left(\frac{(-1)^k}{(k-1)!} x^{-\xi} (\log x)^k \right) + O(x^{-\gamma}).$$

Le lecteur pourra trouver l'essentiel des résultats concernant la transformée de Mellin dans [72].

C.2 Poissonnisation

Le calcul s'inspire de [52, 53]. Pour calculer la hauteur moyenne des tries, on utilise une transformée de Poisson : le nombre N de mots infinis devient une variable aléatoire, avec la distribution

$$\mathcal{Pr}(N = n) = e^{-z} \frac{z^n}{n!}.$$

Dans le modèle de Bernoulli, soit N_u la variable aléatoire désignant le nombre d'apparitions du mot $u \in \mathbb{F}_2^k$ et soit p_u sa probabilité d'apparition ($p_u = p^i q^{k-i}$ où i est le poids de u et $q = 1 - p$).

Si $\sum_u n_u = n$, on a

$$\mathcal{Pr}(\forall u, N_u = n_u) = \frac{n!}{\prod_u n_u!} \prod_u p_u^{n_u}.$$

Dans le cas de la distribution de Poisson

$$\mathcal{Pr}(\forall u, N_u = n_u) = \frac{e^{-z} z^n}{n!} \frac{n!}{\prod_u n_u!} \prod_u p_u^{n_u} = e^{-z} \prod_u \frac{(z p_u)^{n_u}}{n_u!}.$$

La série génératrice correspondante vaut

$$\begin{aligned} (x_u) &\rightarrow e^{-z} \sum_{(n_u) \in \mathbb{N}^{2^k}} \left[\prod_u \frac{(z p_u x_u)^{n_u}}{n_u!} \right], \\ &= e^{-z} \prod_u \left[\sum_{n \in \mathbb{N}} \frac{(z p_u x_u)^n}{n!} \right], \\ &= e^{-z} \prod_u e^{z p_u x_u}. \end{aligned}$$

Or la probabilité $\hat{p}_k$ qu'on ait un arbre de hauteur inférieure ou égale à k est celle d'avoir k mots distincts, soit $\forall u, n_u = 0$ ou 1, dans le cas de Poisson

$$\hat{p}_k = e^{-z} \prod_u (1 + z p_u).$$

Ce qui donne pour la même probabilité dans le cas de Bernoulli, où n est fixé

$$p_{k,n} = [z^n] e^z n! \hat{p}_k = n! [z^n] \prod_u (1 + z p_u).$$

C.3 Dépoissonnisation

Nous rappelons tout d'abord la formule de Cauchy pour un chemin général. Soit U un ouvert connexe, C un chemin fermé inclus dans U et F holomorphe sur U , alors $\forall a \notin C$

$$F(a) = \frac{1}{2i\pi} \oint_C \frac{F(Z)}{z - a} dz.$$

Pour estimer $p_{k,n}$, nous appliquons la formule de Cauchy, en prenant une boule centrée en 0,

$$p_{k,n} = \frac{n!}{2i\pi} \oint \frac{\prod_u (1 + zp_u)}{z^{n+1}} dz. \quad (\text{C.1})$$

L'espérance de la hauteur des tries vaut donc

$$\sum_{k=0}^{\infty} \left[1 - \frac{n!}{2i\pi} \oint \frac{\prod_u (1 + zp_u)}{z^{n+1}} dz \right].$$

Nous remarquons d'abord que

$$\begin{aligned} \prod_u (1 + zp_u) &= \exp \left(\sum_u \log(1 + zp_u) \right) \\ &= \exp \left(\sum_u \sum_{l=1}^{\infty} \frac{(-1)^{l+1} z^l p_u^l}{l} \right) \\ &= \exp \left(\sum_{l=1}^{\infty} \frac{(-1)^{l+1} z^l}{l} (p^l + q^l)^k \right) \\ &= e^z e^{-\frac{z^2}{2}(p^2+q^2)^k} e^{R(z)}, \end{aligned} \quad (\text{C.2})$$

avec

$$R(z) = \sum_{l=3}^{\infty} \frac{(-1)^{l+1} z^l}{l} (p^l + q^l)^k.$$

Comme la formule de Cauchy, appliquée à la seule fonction e^z , vaut 1, cela nous conduit à approximer notre formule par une série double exponentielle de terme général $1 - e^{-\frac{n^2}{2}\lambda^k}$, avec $\lambda = p^2 + q^2 < 1$. Le comportement asymptotique de celle-ci nous sera donné grâce à la transformée de Mellin.

Plus précisément, il s'agit de montrer que

$$\sum_{k=0}^{\infty} \left[\frac{n!}{2i\pi} \oint \frac{\prod_u (1 + zp_u)}{z^{n+1}} dz - e^{-\frac{n^2}{2}\lambda^k} \right] \xrightarrow{n \rightarrow \infty} 0,$$

ou encore, en calculant l'intégrale sur $\Gamma_n = \{ne^{i\theta}, \theta \in [-\pi, \pi]\}$ et en regroupant les termes

$$\sum_{k=0}^{\infty} \left[\frac{n!}{2i\pi} \int_{\Gamma_n} \frac{\prod_u (1 + zp_u) - e^{-z\frac{n^2}{2}\lambda^k}}{z^{n+1}} dz \right] \xrightarrow{n \rightarrow \infty} 0. \quad (\text{C.3})$$

• *Définition de $k(n)$*

Soient $\varepsilon > 0$, $a_2 = \frac{|\log(p^2+q^2)|}{2-\varepsilon}$ et $a_3 = \frac{|\log(p^3+q^3)|}{3-\varepsilon}$. On définit $k(n)$ par

$$k(n) = \lfloor \frac{\log n}{a_2} \rfloor,$$

de sorte que $n^\varepsilon \leq \lambda^{k(n)} n^2 \leq \lambda^{-1} n^\varepsilon$.

Nous utilisons l'inégalité de Hölder :

$$\sum_{k=1}^n |x_k y_k| \leq \left(\sum_{k=1}^n |x_k|^p \right)^{\frac{1}{p}} \left(\sum_{k=1}^n |y_k|^p \right)^{\frac{1}{p}}.$$

Celle-ci nous assure que pour ε assez petit, $a_3 > a_2$. De plus, $k(n) \rightarrow \infty$, il existe donc n_0 tel que pour tout $n > n_0$,

$$k(n) > \frac{a_2}{a_3 - a_2},$$

et

$$k(n)a_3 > a_2(k(n) + 1) > \log(n).$$

Ce qui implique

$$n^3(p^3 + q^3)^{k(n)} \leq n^{-\varepsilon},$$

le terme cubique du développement (C.2) est donc négligeable. Montrons maintenant qu'il en est de même des suivants.

- *Étude de $R(z)$ quand $z \rightarrow \infty$*

D'après l'inégalité de Hölder,

$$\forall l \geq 3, \quad n^l(p^l + q^l)^k \leq [n(p^3 + q^3)^{k/3}]^l.$$

Comme $n^3(p^3 + q^3)^{k(n)} \rightarrow 0$, il existe donc n_0 tel que pour tout $n > n_0$, $n(p^3 + q^3)^{k(n)/3} \leq 1/2$ et $\forall k \geq k(n)$,

$$|R(ne^{i\theta})| \leq \sum_{l=3}^{\infty} n^l(p^l + q^l)^k \leq \sum_{l=3}^{\infty} [n(p^3 + q^3)^{k/3}]^l \leq 2n^3(p^3 + q^3)^k.$$

On en conclut que

$$\forall k \geq k(n), \quad |R(n)| \leq 2n^3(p^3 + q^3)^k \leq 2n^{-\varepsilon}. \quad (\text{C.4})$$

- $k \leq k(n)$

Par définition, nous avons $p_{k,n} \leq p_{k(n),n}$. On se contente donc d'étudier $k = k(n)$; ainsi, l'étude qui précède reste valable. On majore simplement $|\prod_u (1 + zp_u)|$ par $\prod_u (1 + np_u)$. De plus, la formule de Stirling nous donne un équivalent de $n!$,

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e} \right)^n.$$

L'intégrale (C.1) est ainsi majorée en module par le produit d'un terme en $\frac{n!e^n}{n^n}$, équivalent d'après la formule de Stirling à $\sqrt{2\pi n}$ et $\exp(-\frac{n^2}{2}\lambda^k + R(n))$. D'après (C.4), $R(n)$ tend vers 0 et $\exp(-\frac{n^2}{2}\lambda^k) \leq \exp(-n^\varepsilon)$. Ainsi,

$$\sum_{k \leq k(n)} p_{k,n} = \mathcal{O}(\log(n) \sqrt{n} e^{-n^\varepsilon}).$$

La partie correspondante de la série double exponentielle est elle aussi négligeable

$$\sum_{k \leq k(n)} e^{-\frac{n^2}{2}\lambda^k} \leq k(n)e^{-n^\varepsilon/2},$$

ce qui nous permet de conclure

$$\sum_{k \leq k(n)} p_{k,n} - e^{-\frac{n^2}{2}\lambda^k} \xrightarrow{n \rightarrow \infty} 0. \quad (\text{C.5})$$

• $k > k(n)$

On utilise ici la *méthode du col* [57, 206] : l'intégrale de (C.3) est divisée en deux parties, l'une, $I_{2,k}$ autour de n , sur $\gamma_n = \{ne^{i\theta}, \theta \in [-\theta_n, \theta_n]\}$, l'autre, $I_{1,k}$ sur $\Gamma_n \setminus \gamma_n$.

• **sur $\Gamma_n \setminus \gamma_n$**

Ici encore, on majore indépendamment la somme des produits et celle des doubles exponentielles pour montrer qu'elles sont toutes les deux négligeables. On effectue la décomposition dans (C.3),

$$\left| \prod_u (1 + zp_u) - e^{-\frac{n^2}{2}\lambda^k} \right| \leq |e^z - \prod_u (1 + zp_u)| + |e^z - e^{-z\frac{n^2}{2}\lambda^k}|.$$

D'une part, sur $\mathbb{R}^+$, $|1 - e^{-x}| \leq x$, ce qui implique

$$\frac{n!}{2\pi} \int_{\Gamma_n \setminus \gamma_n} \left| \frac{e^z}{z^{n+1}} (1 - e^{-\frac{n^2}{2}\lambda^k}) dz \right| \leq \frac{n!}{n^n} e^{n \cos(\theta_n)} n^2 \lambda^k,$$

d'autre part, sur $\mathbb{C}$, $|1 - e^z| \leq |z|e^{|z|}$, combiné avec l'équation (C.4) et $|z^2\lambda^k| \leq n^\varepsilon$, nous obtenons

$$\begin{aligned} & \frac{n!}{2\pi} \int_{\Gamma_n \setminus \gamma_n} \left| \frac{e^z}{z^{n+1}} (1 - e^{-\frac{z^2}{2}\lambda^k + R(z)}) dz \right| \\ & \leq \frac{n!}{2\pi} \int_{\Gamma_n \setminus \gamma_n} \left| -\frac{z^2}{2}\lambda^k + R(z) \right| e^{|\frac{z^2}{2}\lambda^k| + |R(z)|} \left| \frac{e^z}{z^{n+1}} dz \right| \\ & \leq \left(n^2 \lambda^k + 2n^3(p^3 + q^3)^k \right) \frac{n!}{n^n} e^{n \cos(\theta_n)} e^{2n^{-\varepsilon} + n^\varepsilon}. \end{aligned}$$

D'après la formule de Stirling,

$$\frac{n!}{n^n} e^{n \cos(\theta_n)} = \frac{n!e^n}{n^n} e^{n(\cos(\theta_n)-1)} = \mathcal{O}(\sqrt{n}e^{-n\theta_n^2}).$$

De plus, $k > k(n)$, ce qui nous assure que $\sum_{k > k(n)} \lambda^k = \mathcal{O}(\lambda^{k(n)}) = \mathcal{O}(n^\varepsilon)$ et $\sum_{k > k(n)} (p^3 + q^3)^k = \mathcal{O}(n^{-\varepsilon})$.

En sommant sur k ,

$$\sum_{k > k(n)} |I_{1,k}| = \mathcal{O}(n^\varepsilon \sqrt{n} e^{-n\theta_n^2 + n^\varepsilon}). \quad (\text{C.6})$$

En choisissant par exemple $\theta_n = n^{-\frac{1}{3}}$ et $\varepsilon < \frac{1}{3}$ cette partie de la somme tend vers 0.

· **sur** γ_n

$$I_{2,k} = \frac{n!}{2i\pi} \int_{\gamma_n} f_n(z) \frac{e^z}{z^{n+1}} dz,$$

avec

$$f_n(z) = e^{-\frac{z^2}{2}\lambda^k + R(z)} - e^{-\frac{n^2}{2}\lambda^k}.$$

D'où

$$\begin{aligned} |f_n(z)| &\leq \sup_{t \in [0,1]} \left| \left(R(z) - \frac{(z^2 - n^2)}{2} \lambda^k \right) \exp\left(-\frac{t(z^2 - n^2) + n^2}{2} \lambda^k + tR(z)\right) \right| \\ &\leq \left(|R(z)| + \lambda^k n |z - n| \right) e^{n^{-\varepsilon}} e^{-n^2 \cos(2\theta_n) \lambda^k}. \end{aligned}$$

On a par ailleurs, $|R(z)| \leq 2n^3(p^3 + q^3)^k$ et $|z - n| \leq n \sin(\theta_n) \leq n\theta_n$, d'où

$$|I_{2,k}| \leq \left[n^2 \theta_n \lambda^k + 2n^3(p^3 + q^3)^k \right] \frac{n!}{2\pi n^n} \int_{-\theta_n}^{\theta_n} e^{n \cos(\theta)} d\theta.$$

On voit facilement que le second terme est borné. En effet, la majoration $\cos(\theta) \leq 1 - \frac{\theta^2}{2} + \frac{\theta^4}{24}$ implique

$$\int_{-\theta_n}^{\theta_n} e^{n \cos(\theta)} d\theta \leq 2e^n e^{n\theta_n^4} \int_0^{\theta_n} e^{-n\theta^2/2} d\theta.$$

Par ailleurs, $\int_0^{\theta_n} e^{-n\theta^2/2} d\theta \leq \sqrt{\frac{2}{n}} \int_0^\infty e^{-t^2} dt$, et $n\theta_n^4 = n^{-\frac{1}{3}} \xrightarrow{n \rightarrow +\infty} 0$, ce qui nous donne

$$\frac{n!}{2\pi n^n} \int_{-\theta_n}^{\theta_n} e^{n \cos(\theta)} d\theta = \mathcal{O}\left(\frac{n! e^n}{n^n \sqrt{n}}\right) = \mathcal{O}(1).$$

En sommant sur k ,

$$\sum_{k > k(n)} |I_{2,k}| = \mathcal{O}(n^2 \lambda^{k(n)} \theta_n + n^3 (p^3 + q^3)^{k(n)}) = \mathcal{O}(n^\varepsilon \theta_n). \quad (\text{C.7})$$

Le choix $\theta_n = n^{-\frac{1}{3}}$ et $\varepsilon < \frac{1}{3}$ assure que cette somme tend vers 0.

Les trois sommes (C.5), (C.6) et (C.7) tendent vers 0 quand n tend vers l'infini ; la convergence (C.3) est ainsi démontrée.

C.4 Étude asymptotique

La transformée de Mellin de la somme

$$S(z) = \sum_0^\infty \left(1 - \exp\left(-\frac{z^2}{2} \lambda^k\right) \right)$$

vaut

$$S^*(s) = -2^{s/4-1} \frac{\Gamma(s/2)}{1 - \lambda^{-s/2}}.$$

La bande fondamentale est $\langle -2, 0 \rangle$ et le développement en $s = 0$ vaut

$$S^*(s) \asymp -\frac{2}{\log \lambda} \frac{1}{s^2} + \left(\frac{\gamma + \log(2)}{\log(\lambda)} - \frac{1}{2} \right) \frac{1}{s}.$$

La ligne $\Re(s) = 0$ contient d'autres pôles qui donnent une fluctuation périodique. D'après le théorème C.1,

$$S(z) \underset{+\infty}{\sim} \frac{2}{\log(\lambda)} \log(n) + Q_F(\log(n)) - \left(\frac{\gamma + \log(2)}{\log(\lambda)} - \frac{1}{2} \right) + o(1),$$

où Q_F est une fonction périodique de très petite amplitude. La relation (C.3) permet de conclure que la hauteur moyenne dans le modèle de Bernoulli possède le même comportement asymptotique.

Annexe D

Logiciels de stéganographie 2004-2006

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
Anahtar	-	4		400 \$
ce programme permet de générer une clé de protection anti-copie pour n'importe quel exécutable. Cette clé est cachée sur une disquette et empêche l'exécutable de démarrer si le mot de passe associé n'est pas correcte. Auteur : Murat AYDIN Homepage : -				
BackYard	1 M	1.3		35 \$
BackYard est un système de fichiers stéganographique qui peut cacher des lecteurs, des répertoires et des fichiers de la vue de Windows. Auteur : Talyasoft Remarque : NON DISPONIBLE				
Blindside	123 K - 222 K 47 K	0.9b	  SRC	Free
Blindside permet de cacher un fichier ou un ensemble de fichiers dans une image BMP. Il effectue un petit changement de couleur imperceptible pour l'œil. Une image peut contenir environ 50 kb de données. Auteur : John Collomosse Homepage : http://www.cs.bath.ac.uk/~jpc/blindside/index.htm				
BMP Secret	2 M	-		Free
programme pour cacher n'importe quel type de fichier dans une image BMP. Auteur : Parallel Worlds Homepage : http://www.pworlds.com/products/secrets.html				
bProtected 2000	-	-		19.95 \$
bProtected est un système de fichiers stéganographique. Il opère suivant 4 niveaux graduels de protection. Auteur : Clasys Homepage : http://www.clasys.com				

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
BuryBury	1.27 M	1.7		15 \$
BuryBury est un système de fichiers stéganographique qui utilise une interface Windows Explorer avec protection par mot de passe. Utilise l'algorithme de chiffrement TwoFish. Auteur : Paul Gerhart Homepage : http://mysite.verizon.net/pgerhart				
CameraShy	1.3 M - 33 K	0.2.23.1	 SRC	Free
application stéganographique basée sur un navigateur web. Utilise AES 256 bits et cache l'information sur les bits de poids faible d'image GIF. Auteur : Hacktivismo Homepage : http://www.hacktivismo.com				
Camouflage	2.6 M	1.21		Free
application qui permet de cacher n'importe quel fichier en le découpant et l'attachant aux fichiers de son choix. Permet l'utilisation d'un mot de passe pour le fichier. Auteur : Unfiction Homepage : http://camouflage.unfiction.com				
Camouflage Password Finder	4.4 K	2.0	 SRC	Free
application qui permet de retrouver le mot passe utilisé pour stéganographier avec Camouflage. Auteur : Guillermito Homepage : http://www.guillermito2.net				
Cloak	2.3 M	7		34.95 \$
Cloak permet de cacher tous type de fichier dans une image. Gère 7 formats de fichiers graphiques. Utilise les algorithmes Cloak-128, Blowfish, Mercury sur 256 bits. Guillermito nous indique comment retrouver les données cachées. Auteur : Insight-Concepts Homepage : http://www.insight-concepts.com				
Contraband 9i	240 K	9i		Free
programme pour cacher n'importe quel type de fichier dans une image bitmap de 24 bits. Auteurs : Julius Thyssen, Hens Zimmerman Homepage : http://www.jthz.com/puter				
Contraband hell	1,7 M	Hell	 SRC	Free
programme pour cacher n'importe quel type de fichier dans une image bitmap de 24 bits. Evolution de contraband qui utilise l'algorithme IDEA pour chiffrer le fichier à dissimuler. Auteurs : Julius Thyssen, Hens Zimmerman Homepage : http://www.jthz.com/puter				
Courier	188 K	1.0a		Free
cache du texte dans des images bitmap couleur. Auteurs : Kelce et Robyn Wilson Homepage : -				

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
Covert TCP	18 K	-	 SRC	Free
utilise des paquets TCP pour faire passer de l'information au travers d'un canal caché. Auteurs : Craig H. Rowland Homepage : http://www.firstmonday.dk/issues/issue2_5/rowland				
CryptArkan	349 K	-		29,99 \$
cache un fichier ou un répertoire dans des fichiers image ou audio. Homepage : http://cryptarkan.fromru.com/index.html				
Crypto 123	350 K	1.04		10 \$
cache un fichier dans un autre. Chiffre les fichiers à dissimuler et les protège par mot de passe. Auteur : kellysoftware Homepage : http://www.kellysoftware.com				
Dark Files	859 K	2.4.1.1		20 \$
système stéganographique de gestion de fichiers windows. Gère aussi les fichiers réseau et les supports amovibles. Possède trois niveaux de sécurité. Auteur : SSS Lab. Homepage : http://www.1securitycenter.com/df				
Data Stash	-	1.1b		19.95 \$
permet de cacher des fichiers dans un ensemble de fichiers supports. Fourni une protection par mot de passe et utilise BlowFish comme algorithme de chiffrement. Guillermito nous indique comment retrouver l'information cachée. Auteur : Skyjuice Software Homepage : http://www.skyjuicesoftware.com				
Digital Picture Enveloppe	1,2 M	0.2		Free
cache l'information dans des fichiers GIF en utilisant la méthode BPCS. Cette méthode permet de cacher une importante quantité d'informations. Auteur : KIT Digital Picture Enveloping Research Group Homepage : -				
Dound	2 M	1.6		Free
permet aux utilisateurs de coder et décoder des messages avec un mot de passe. Le codage se fait dans des images. Auteur : DGU Enterprises Homepage : http://evidence-eliminators.co.uk/dound.htm				
DPT 32	1 M	-		Free
chiffre les données et les dissimule dans une image BMP 24 bits. Auteur : Bernard Homepage : http://www.xs4all.nl/~bernard/home.f.html				

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
DriveCrypt	-	4.1		59.95 \$
effectue du chiffrement de partition et de fichier en temps-réel sur 1344 bits avec les algorithmes AES, Blowfish, Tea 16, Tea 32, Des, Triple Des, Misty 1 and Square. Cache de l'information dans des fichiers Wav. Il est également pourvu de nombreuses fonctionnalités de sécurité. Auteur : SecurStar Homepage : http://www.securstar.com/products_drivecrypt.php				
Drive Hider	217 K	-		Free
permet de cacher aux utilisateurs certaines lettre de disque. Auteur : Reed Suckow Homepage : http://www.papertrailsoftware.com/hidedrives/index.htm				
Easy File & Folder Protector	933 K	3.6		34 \$
protège et cache les fichiers et répertoires en agissant au niveau du noyau de windows. Auteur : Softstack Homepage : http://www.softstack.com				
EasyMemo	4.5 M	1.4		22.52 Eur
permet de regrouper et gérer tous les mots de passe à partir d'un seul. Les mots de passe sont chiffrés et caché dans d'autres fichiers. Auteur : JRSoftware Homepage : http://www.jrsoft.de				
Embed256	39 K	1.1	Amiga	Free
cache les données dans des images de 256 couleurs.				
EmbedIFF24	39 K	-	Amiga	Free
cache les données dans des images couleur IFF24.				
EmptyPic	6.8 K	-		Free
cache des images GIF dans des pages web en les transformant en image unie-couleur. Auteur : Robert Wallington Homepage : http://www.crtelco.com/~robertw				
EncryptPic	471 K	1.3		Shareware
cache l'information dans des images bitmap de 24 bits. Utilise une protection par mot de passe et l'algorithme Cast pour chiffrer les données. Auteur : Frederic Collin Homepage : http://www.softlookup.com/preview/dis24355.html				
EzStego	6.8 K	-	Java	Free
chiffre et cache l'information dans des images GIF. Auteur : Romana Machado Homepage : http://www.stego.com				

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
F5	239 K	12b	Java	Free
utilise l'algorithme stéganographique F5 pour dissimuler l'information dans des images en vraies couleurs BMP, GIF ou JPEG. Auteur : Andreas Westfeld Homepage : http://www.rn.inf.tu-dresden.de/~westfeld/f5.html				
FatMacPGP	1.3 M	2.6.3	Macintosh	Free
utilitaire stéganographique pour PGP. Auteur : Zbigniew Fiedorowicz Homepage : http://www.math.ohio-state.edu/~fedorow/PGP				
FFencode	12 K	-	DOS	Free
cache dans des fichiers texte en utilisant un "code Morse" de caractères NULL. Auteur : Homepage : http://www.burks.de/stegano/ffencode.html				
File Protector	451 K	5.1219		218 \$
protège les répertoires et les fichiers de différentes manières, notamment en les dissimulant. Auteur : Mikkotech Homepage : http://www.mikkotech.com				
Folder Guard	0.8 M	6.0		39.95 \$
protège les répertoires et les fichiers en les dissimulant. Auteur : winability Homepage : http://www.winability.com				
FortKnox	-	3.56.3		25 pounds
chiffre et cache n'importe quel type de fichier dans n'importe quel autre!!! Auteur : ClickOK Ltd Homepage : http://www.clickok.co.uk				
FortKnox Hidden Text Finder	3.4 K	-	 SRC	Free
outil qui permet de retrouver l'information cachée avec FortKnox. Auteur : Guillermito Homepage : http://www.guillermito2.net				
GhostHost	289 K	1.0a		Free
attache le fichier à dissimuler à la fin du fichier hôte. Auteurs : Kelce et Robyn Wilson Homepage : -				
Gif It Up	1.4 M	1.0		Free
programme de stéganographie pour W95 qui cache l'information dans des images GIF en changeant des couleurs en d'autres très proches. Auteurs : Lee Nelson Homepage :				

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
Gifshuffle	26 K - 14 K 26 K	2.0	  SRC	Free
programme en ligne de commandes qui cache des messages dans des images GIF en mélangeant la palette des couleurs. Les données sont compressées et chiffrées. Auteur : Matthew Kwan Homepage : http://www.darkside.com.au/gifshuffle/index.html				
GZSteg	267 K	1.2.4	  DOS OS/2 Amiga Atari SRC	Free
dissimule les données dans un fichier zip. Auteur : Preston Wilson Homepage :				
Hermetic Stego	-	-		29 \$
cache l'information dans des fichiers BMP. Auteur : Hermetic System Homepage : http://www.hermetic.ch				
Hide4PGP	26 K 20 K 41 K 53 K 31 K	2.0	  DOS OS/2 SRC	Free
outil stéganographique dissimulant l'information dans des fichiers BMP, Wav ou Voc. Utilise Stealth PGP pour chiffrer les données. Auteur : Heinz Repp Homepage : http://www.heinz-repp.onlinehome.de/Hide4PGP.htm				
Hide Drive	289 K	1.5.1		Free
permet de cacher à windows l'existence d'un média amovible. Auteur : Innovative Software Creations Homepage : http://www.geocities.com/iscwebsite/index.html				
Hide Drives	36 K	-		Free
permet de cacher à windows l'existence d'un média amovible. Auteur : Homepage :				
Hide Folders XP - Hide Folders	885 K - 720 K	1.5 - 2.4		24.95 \$
verrouille à l'aide d'un mot de passe jusqu'à 64 répertoires et les dissimule sur le disque. Auteur : FS Pro Labs. Homepage : http://www.fspro.net				
Hide In Picture	742 K - 288 K	2.1	 DOS SRC	Free
programme écrit en langage Euphoria. Chiffre, protège par mot de passe et cache n'importe quelle donnée dans un fichier BMP. Supporte le format GIF. Auteur : Davi Tassinari de Figueiredo Homepage : http://www16.brinkster.com/davitf/hip/				

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
Hide In Picture	255 K - 367 K 367 K - 255 K	2.0	 DOS SRC	Free
programme écrit en langage Euphoria. Chiffre, protège par mot de passe et cache n'importe quelle donnée dans un fichier BMP. Auteur : Davi Tassinari de Figueiredo Homepage : http://www16.brinkster.com/davitf/hip/				
Hide and Seek	209 K	5		Free
chiffre l'information à dissimuler avec l'algorithme IDEA et la cache dans des images GIF. Auteur : Colin Maroney				
Hydan	176 K	0.13	  SRC	Free
cache l'information dans des fichiers binaires en utilisant la redondance du jeu d'instructions d'Intel x86. Le fichier support ne change pas de taille. Auteur : Rakan El-Khalil Homepage : http://www.crazyboy.com/hydan				
Hydan	50 K	0.10	  SRC	Free
cache l'information dans des fichiers binaires en utilisant la redondance du jeu d'instructions d'Intel x86. Le fichier support ne change pas de taille. Auteur : Rakan El-Khalil Homepage : http://www.crazyboy.com/hydan				
ImageHide	1.1 M	-		Free
logiciel de stéganographie gérant plusieurs formats d'images. Auteur : Dancemammal Homepage : http://prem-01.portlandpremium.co.uk/p1-28/imagehide.htm				
ImageHide Hidden Text Finder	5 K	-	 SRC	Free
outil qui permet de détecter et retrouver l'information cachée avec ImageHide. Auteur : Guillermito Homepage : http://www.guillermito2.net				
Info Stego	900 K	3		19.95 \$
chiffre les données à cacher et les dissimule dans des images ou des flux audios. Auteur : Antiy Labs. Homepage : http://antiy.net				
In The Picture	1894 K	-		25 \$
chiffre les données à cacher et les dissimule dans des images BMP. Auteur : Intar Homepage : http://www.intar.com				

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
InThePicture Password Finder	4.3 K	-	 SRC	Free
application qui permet de retrouver le mot passe utilisé pour stéganographier avec InThePicture. Auteur : Guillermito Homepage : http://www.guillermito2.net				
Invisible Files 2000 Pro	670 K	-		59.95 \$
rend invisible des fichiers aux yeux de Windows et de Dos. Auteur : Raytown Corp. Homepage : http://www.softsecurity.com				
Invisible Secrets	2.65 M	4		39.95 \$
permet de dissimuler des données dans des fichiers JPEG, PNG, BMP, HTML et WAV. Chiffre les données avec AES-Rijdael, Blowfish, Twofish, RC4, Cast128, GOST, Diamond 2, Sapphire 2. Gestion à base de mots de passe et plein d'autres fonctionnalités de sécurité. Auteur : NeoBytes Homepage : http://www.neobytesolutions.com				
JP Hide and Seek	15 K - 181 K	0.3 - 0.5	  SRC DOS	Free
programme de stéganographie désigné pour dissimuler peu d'information, moins de 5 %, dans des images JPEG. Auteur : Allan Latham Homepage : http://linux01.gwdg.de/~alatham/stego.html				
JR-Stegano	4.6 M	1.0		39 Eur
cache l'information dans des fichiers BMP ou PNG de 24 millions de couleurs. Auteur : JR-Software Homepage : http://www.jrsoft.de				
JR-SteganoAX	-	1.1	 ActiveX	16.57 \$
cache l'information dans des fichiers BMP ou PNG de 24 millions de couleurs. Auteur : JR-Software Homepage : http://www.jrsoft.de				
Jsteg Shell	2,1 M	1.0		Free
cache les données dans des images JPEG. Utilise un chiffrement RC4 sur 40 bits. Auteur : Korejwa				
JSteg Data Extractor	8.2 K	-	 SRC	Free
application qui permet de retrouver l'information stéganographiée avec Jsteg. Auteur : Guillermito Homepage : http://www.guillermito2.net				

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
KPK File Pro	-	8.0		39 \$
chiffre les données sensibles et les cache dans des documents ouverts. Gère l'accès par mot de passe. Auteur : John ROSSI Homepage : http://www.kpkfile.com/index.html				
KPK File	1.7 M	7.1		Free
chiffre les données sensibles et les cache dans des documents ouverts. Gère l'accès par mot de passe. Auteur : John ROSSI Homepage : http://www.kpkfile.com/index.html				
Magic Folders / Encrypted Magic Folder	-	-		10 \$ - 30 \$
rend invisible et chiffre des dossiers. L'accès se fait à l'aide d'un mot de passe. Auteur : RSE Software Inc. Homepage : http://www.pc-magic.com				
Mandelsteg G-Extract	19 K	1.0	 SRC	Free
crée une image fractale de Mandelbrot en format GIF et y cache l'information. Auteur : Henry Hastur				
Masker	2.5 M	7.0		19.95 \$
rend invisible et chiffre des dossiers. L'accès se fait à l'aide d'un mot de passe. 7 algorithmes de chiffrement disponibles dont BLOWFISH, RIJNDAEL, TripleDES. Auteur : Softpuls Homepage : http://www.softpuls.com				
Merge Streams	32 K	1.0		Free
fusionne un flux MS Word et un flux MS Excel, l'un dissimulant l'autre. Auteur : ntkerne Homepage : http://www.ntkernel.com/utilities/merge.shtml				
Mimic Functions	185 K	-	Macintosh	Free
outil stéganographique basé sur les grammaires contextuelles. Il génère du texte à partir du fichier à cacher. Auteur : Peter Wayner				

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
MP3 Stego / MP3 Stego (GUI W32)	1.4 M - 339 K	1.1.17	  DOS OS/2 Amiga SRC	Free
compresse, chiffre les données et les dissimule dans un flux MP3. Auteur : Fabien Petitcolas Homepage : http://www.petitcolas.net/fabien/steganography/mp3stego/index.html				
MP3 Stego / MP3 Stego (GUI W32)	1.4 M - 339 K	1.1.16	  DOS OS/2 Amiga SRC	Free
compresse, chiffre les données et les dissimule dans un flux MP3. Auteur : Fabien Petitcolas Homepage : http://www.petitcolas.net/fabien/steganography/mp3stego/index.html				
NICETEXT	13 M	0.9	 SRC	Free
génère un texte qui ressemble à un texte dans un langage naturel à partir de données chiffrées. Utilise des grammaires non contextuelles. Auteur : Mark Chapman Homepage : http://www.ctgi.net/nicetext/index.html				
OutGuess	460 K	0.2	 SRC	Free
outil stéganographique pour les images JPEG qui préserve les statistiques sur les fréquences. Auteur : Niels Provos Homepage : http://www.outguess.org				
Parnoid	87 K	-	Macintosh	Free
outil qui chiffre avec IDEA et Triple DES et cache les données dans un fichier audio. Auteur : Nathan Mariels				
PC FileSafe	1.5 M	-		9.95 \$
verrouille et rend invisible des fichiers et dossiers. L'accès se fait à l'aide d'un mot de passe. Auteur : ShelbyComputer Homepage : http://www.shelbycomputer.com				
PGM.stealth	4,3 K	-	 SRC	Free
cache l'information dans des fichiers PGM. Auteurs : Timo Rinne et Cirion Oy				
Piilo	5.9 K	-	 SRC	Free
cache l'information dans des fichiers PGM en niveau de gris. Auteurs : Tuomas Aura Homepage : http://research.microsoft.com/users/tuomaura				

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
Point Lock PRO	3.26 M	2		34.92 Eur
verrouille et rend invisible des fichiers et dossiers. L'accès se fait à l'aide d'un mot de passe. Auteur : Ahranta Homepage : http://www.ahranta.com				
Pretty Good Envelope	25 K	1.0	DOS	Free
cache un fichier binaire en le concaténant à un autre. Auteur : Roche'Crypt Homepage :				
PrivateInfo	674 K	2.2		39.95 \$
rend invisible des fichiers et dossiers. Auteur : Raysion Software Co. Homepage : http://www.raysion.com				
RightClickHide	867 K	1.2		25 \$
rend invisible des fichiers et dossiers ainsi que des lecteurs. Auteur : Talyasoft Remarque : NON DISPONIBLE				
Safe and Quick Hide Files	2.2 M	-		19,6 \$
chiffre et cache des fichiers et répertoires. Guillermito explique une technique pour retrouver le mot de passe utilisé. Auteur : Microidea Software Studio. Remarque : NON DISPONIBLE				
Sam's Big Play Maker	182 K - 12 K	-	 SRC	Free
converti du texte arbitrairement en un scénario amusant. Auteur : SecurStar Homepage : http://www.scramdisk.clara.net/play/playmaker.html				
SandMark	-	3.3	  SRC	Free
permet de faire du watermarking sur des programmes Java. Auteur : Christian Collberg Homepage : http://www.cs.arizona.edu/sandmark				
Secret Spaces	21 K	1.0		Free
cache des messages courts dans des textes clairs en codant l'information sur les espaces. Auteur : Andrew Gray Homepage : http://www.andrewgray.com				

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
SecureEngine Professional	2.6 M	1.0		Free
permet de dissimuler des données dans des fichiers BMP, des WAV, des fichiers TEXT sans modifier leur taille ainsi que dans des fichiers JPEG. Chiffre les données avec 3-Way, Blowfish, Cast, GOST, Mars, Vernam à base de mots de passe et possède plein d'autres fonctionnalités de sécurité. Auteur : Adrien Pinet Homepage : http://secureengine.isecurelabs.com				
SecureEngine	2.6 M	4		Free
permet de dissimuler des données dans des fichiers BMP, des WAV, des fichiers TEXT sans modifier leur taille ainsi que dans des fichiers JPEG. Chiffre les données avec 3-Way, Blowfish, Cast, GOST, Mars, Vernam à base de mots de passe et possède plein d'autres fonctionnalités de sécurité. Auteur : Adrien Pinet Homepage : http://secureengine.isecurelabs.com				
Snow	14 K - 31 K 14 K - 11 K 24 K - 35 K	-	  SRC java+ classe applet	Free
Cache un message en le codant avec les espaces d'un texte ASCII. Auteur : Matthew Kwan Homepage : http://www.cs.mu.oz.au/~mkwan				
Snowdisk	8.5 K	1.0	 SRC	Free
chiffre l'information avec PGP et la stocke sur un média suivi d'aléa. Le média semble n'être rempli que d'aléa. Auteur : Scott G. Miller				
Snowdrop	32 K	0.02b	 SRC	Free
programme de watermarking dédié aux documents texte et aux codes sources en C. Auteur : Michal Zalewski Homepage : http://lcamtuf.coredump.cx				
Spam Mimic	-	-	HTML	Free
Cache un petit message dans des spams. Homepage : http://www.spammimic.com				
Stealth	55 K	2.01	 SRC	Free
utilitaire stéganographique pour PGP 2.x. Auteur : Adam Back Homepage : http://www.cypherspace.org/adam/stealth				
StealthDisk	1 M	3.6		29.95 \$
rend invisible des fichiers sur le disque dur. Auteur : Invisicom, Inc.				

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
Stegano GifPaletteOrder	138 K	0.2	Java	Free
cache l'information dans les fichiers GIF. Auteurs : David GLAUDE et Didier BARZIN Homepage : http://users.skynet.be/glu/sgpo.htm				
Steganografia	9,1 K	-	  SRC	Free
programme en Perl permettant de cacher de l'information dans des fichiers BMP sans en changer la taille. Auteur : Cers Homepage : http://www.cers.tk				
Steganography	1.02 M	1.6.1		24.95 \$
chiffre et cache l'information dans des fichiers audio et image. Guillermito nous montre comment retrouver les données stéganographiées. Auteur : Pipisoft Homepage : http://www.pipisoft.com				
Steganos	-	3		39.95 \$
chiffre les données avec des clés d'au moins 2048 bits et les dissimule dans des fichiers BMP, DIB, HTML, TXT, VOC et WAV. Auteur : Centurionsoft Remarque : NON DISPONIBLE				
Steganosaurus	12 K	-	 SRC	Free
remplace les bits d'un message clair par des mots d'un dictionnaire. Auteur : John Walker Homepage : http://www.fourmilab.ch				
Stegdetect	1,3 M	0.5	 SRC	Free
outil de détection d'images JPEG stéganographiées avec jsteg, jphide et outguess. Auteur : Niels Provos Homepage : http://www.outguess.org				
StegFs	185 K	1.1.4	 SRC	Free
système de fichiers stéganographique pour Linux. Auteur : Andrew D. McDonald Homepage : http://www.mcdonald.org.uk				
StegHide	663 K - 1.8 M - 663 K	0.5.1	  SRC	Free
programme de stéganographie qui dissimule l'information dans de nombreux types de fichiers image et audio. Résistant aux attaques statistiques du premier ordre. Auteur : Stefan Hetzl Homepage : http://steghide.sourceforge.net/index.php				

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
Stego	193 K	1.0a2	Macintosh	Free
cache l'information sur les bits les moins significatifs dans des images PICT. Auteur : Romana Machado				
Stegodos	22 K	-	DOS	Free
cache l'information sur les bits les moins significatifs dans des images GIF ou PCX. Auteur : Black Wolf				
StegoLame	3.6 M	MP3 at-tackers	 SRC	Free
outils de détection stéganographique sur des flux (MP3,Ogg, MPEG ...). Auteur : noch				
StegoTif	68 K	-	 SRC	Free
cache l'information sur les bits les moins significatifs dans des images TIFF. Auteur : Giovambattista Pulcini Homepage : http ://www.geocities.com/SiliconValley/9210				
StegoWav	68 K	-	 SRC	Free
cache l'information sur les bits les moins significatifs dans des fichiers audio Wav. Auteur : Giovambattista Pulcini Homepage : http ://www.geocities.com/SiliconValley/9210				
StegParty	104 K	-	 SRC	Free
cache les données dans un texte clair en utilisant la ponctuation et la prononciation. Auteur : Steven Hugg				
Stegtunnel	164 K	0.4	 SRC	Free
cache l'information dans des paquets TCP. Auteur : Todd MacDermid Homepage : http ://www.synacklabs.net				
Stella	18 M	-	Java	Free
cache l'information dans des fichiers JPG, GIF et BMP. Auteur : R. Rosenbaum et H. Schumann Homepage : http ://www.stella-steganography.de				
S-Tools	273 K	4	 SRC	Free
cache l'information dans des fichiers image ou audio ou même sur le disque dur. Utilise les algorithmes IDEA et DESS pour chiffrer les données à dissimuler. Implémente un générateur de pseudo-aléa pour choisir les bits supports. C'est un des outils les plus complet. Auteur : Andy Brown				

Nom du Logiciel	Taille du fichier	Version	Plateformes	Licence
TextHide	-	-		39 \$
reformule automatiquement le texte à cacher. Utilise une gestion asymétrique des clés avec RSA. Homepage : http://www.texthide.com				
Texto	59 K	-	 SRC	Free
génère du texte anglais à partir de texte non encodé ou d'ASCII PGP. Auteur : Kevin Maher				
The Third Eye	492 K	1.0		Free
stéganographie les informations à cacher sur les bits de poids faibles d'une image RGB à l'aide d'un générateur pseudo-aléatoire initialisé par un mot de passe. Auteur : Satya Kiran Homepage : -				
Voices	133 K	-	 SRC	Free
cache l'information dans des fichiers MP3. Auteur : Conundrum Homepage : http://www.soldierx.com				
Visual Cryptography	20 K	1.0	 SRC	Free
réparti une image en deux images ; l'image originelle se retrouve en superposant les deux images. Auteur : Jouko Holopainen Homepage : http://iki.fi/jhol				
WbStego4open	535 K - 442 K - 535 K	4.3	  SRC	Free
cache n'importe quel type de données dans des fichiers BITMAP, texte, HTML et PDF. Utilise les algorithmes Blowfish, Twofish, CAST et AES pour le chiffrement des données. Sources en Delphi5+ et Kylix 1+. Auteur : WbStego Homepage : http://wbstego.wbailer.com				
WeavWave	174 K	-		Free
cache l'information dans des fichiers Wav. Auteur : Jagen				
White Noise Storm	83 K	2.10	 SRC	Free
cache l'information dans des fichiers image ou audio de type PCX, TIFF, VOX ou AIFF. Grande capacité de dissimulation. Auteur : Ray (Arsen) Arachelian				

Bibliographie

- [1] Les efforts de la NSA vis-à-vis du Web : la stéganographie. *Le Monde du Renseignement*, 26 octobre 2000.
- [2] Des messages cachés sur l'internet pour préparer les attentats. *Agence Française de Presse*, 12 octobre 2001.
- [3] S.S. AGAIAN, B. RODRIGUEZ et J.P. PEREZ : Stego sensitivity measure and multibit plane based steganography using different color models. In E.J. DELP et P.W. WONG, éditeurs : *Proc. Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, pages 279–290, février 2006.
- [4] ANON : Child pornography on internet. <http://www.instant-essays.com>.
- [5] B.H. ASTROWSKY : “ steganography ”hidden images, a new challenge in the fight against child porn. *UPDATE*, 13(2), 2000.
- [6] I. AVICIBAŞ, N. MEMON et B. SANKUR : Steganalysis based on image quality metrics. In P. W. WONG et E. J. DELP, éditeurs : *Proc. SPIE, Security and Watermarking of Multimedia Contents III*, volume 4314, pages 523–531, 2001.
- [7] M. BACKES et C. CACHIN : Public-key steganography with active attacks. In J. KILIAN, éditeur : *Proc. 2nd Theory of Cryptography Conference (TCC 2005)*, volume 3378 de *Lecture Notes in Computer Science*, pages 210–226. Springer, 2005.
- [8] J. BARBIER : Reconstruction des turbo-codes. Mémoire de DEA, École Polytechnique, Palaiseau, France, 2003.
- [9] J. BARBIER : Reconstruction of turbo-code encoders. In *Proc. SPIE Security and Defense, Space Communication Technologies Symposium*, volume 5819, pages 463–473, Orlando, FL, USA, mars 2005.
- [10] J. BARBIER : La stéganographie moderne : d'Hérodote à nos jours. In *Computer & Electronics Security Application Rendez-vous, CESAR 2007*, Rennes, France, novembre 2007. (A paraître).
- [11] J. BARBIER et S. ALT : Practical insecurity for effective steganalysis. In *Proc. of 10th International Workshop on Information Hiding, IH 2008*, Lecture Notes in Computer Science, Santa Barbara (CA), USA, mai 2008. Springer. (A paraître).
- [12] J. BARBIER et P.E. CAILLARD : Procédés d'émission et de réception de données d ainsi que des données supplémentaires d' et dispositifs associés. Brevet 06/00080, janvier 2006.
- [13] J. BARBIER et P.E. CAILLARD : Procédés d'émission et de réception d'un message numérique et dispositifs associés. Brevet 06/00081, janvier 2006.

- [14] J. BARBIER et V. CAMION : Reconstruction de permutations. *In Proc. MAJESTIC'03*, Marseille, France, octobre 2003.
- [15] J. BARBIER, É. FILIOL et K. MAYOURA : New features for specific JPEG steganalysis. *In C. ARDIL, éditeur : Proc. 3rd International Conference on Computer, Information, and Systems Science, and Engineering, CISE 2006*, volume 16 de *Transactions on Engineering, Computing and Technology*, pages 72–77. World Enformatika Society, novembre 2006. ISBN : 975-00803-6-X.
- [16] J. BARBIER, É. FILIOL et K. MAYOURA : Universal JPEG steganalysis in the compressed frequency domain. *In Y. Q. SHI et B. JEON, éditeurs : Proc. Digital Watermarking, 5th International Workshop, IWDW 2006*, volume 4283 de *Lecture Notes in Computer Science*, pages 253–267, Jeju Island, Korea, novembre 2006. Springer.
- [17] J. BARBIER, É. FILIOL et K. MAYOURA : Universal detection of JPEG steganography. *Journal of Multimedia*, 2(2):1–9, avril 2007. ISSN : 1796-2048.
- [18] J. BARBIER et K. MAYOURA : Steganalysis of the Multi Bit Plane Image Steganography. *In Proc. of Digital Watermarking, 6th International Workshop, IWDW 2007*, Lecture Notes in Computer Science, Guangzhou, China, décembre 2007. Springer. (A paraître).
- [19] J. BARBIER, G. SICOT et S. HOUCKE : Algebraic approach for the reconstruction of linear and convolutional error correcting codes. *In C. ARDIL, éditeur : Proc. 3rd International Conference on Computer Science and Engineering CISE 2006*, volume 16, pages 66–71, Venice, Italy, novembre 2006. World Enformatika Society. ISBN : 975-00803-6-X.
- [20] A. BARG : *Handbook of coding theory.*, volume 1, chapitre 7, Complexity Issues in Coding Theory., pages 649–754. Elsevier Science, 1998.
- [21] G. BATTAIL, C. BERROU et A. GLAVIEUX : Pseudo-random recursive convolutional coding for near capacity performance. *In Proc. Globecom 1993*, pages 23–27, décembre 1993.
- [22] J.-P. BAY : Attention, une image peut en cacher une autre. *Lci*, septembre 2001.
- [23] G. BÉGIN et D. HACCOUN : High-rate punctured convolutional codes : structure properties and construction techniques. *IEEE Transactions on Communications*, COMM-37:1381–1385, décembre 1989.
- [24] G. BÉGIN, D. HACCOUN et C. PAQUIN : Further results on high-rate punctured convolutional codes for Viterbi and sequential decoding. *IEEE Transactions on Communications*, COMM-38, 1990.
- [25] M. BELLARE, A. DESAI, E. JOKIPII et P. ROGAWAY : A concrete security treatment of symmetric encryption : Analysis of the DES modes of operation. *In Proc. 38th Symposium on Foundations of Computer Science FOCS*. IEEE, 1997.
- [26] S. BENEDETTO et G. MONTORSI : Design of parallel concatenated convolutional codes. *IEEE Transactions on Communications*, COMM-44(5):591–600, mai 1996.
- [27] S. BENEDETTO et G. MONTORSI : Unveiling turbo code : some results on parallel concatenated coding schemes. *IEEE Transactions on Information Theory*, IT-42(2): 409–428, mars 1996.
- [28] E.R. BERLEKAMP : *Algebraic Coding Theory*. McGraw-Hill, New York, 1968.

- [29] E.R. BERLEKAMP, R.J. McELIECE et H.C. van TILBORG : On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, IT-24(3), mai 1978.
- [30] C. BERROU : *Codes and Turbocodes*. Springer, mars 2007. ISBN : 978-2-287-32739-1.
- [31] C. BERROU, A. GLAVIEUX et P. THITIMAJSHIMA : Near Shannon limit error-correcting coding and decoding : turbo-codes. *In Proc. of ICC 1993*, Geneva, Switzerland, mai 1993.
- [32] C. BERROU, C. LANGLAIS et F. SEGUIN : *New Directions in Statistical Signal Processing - From Systems to Brains.*, chapitre Turbo Processing. MIT Press, décembre 2006. ISBN : 0-262-08348-5.
- [33] P. BILINGSLEY : *Probability and Measure*. Probability and mathematical statistics. Wiley, 1995. ISBN : 0-471-00710-2.
- [34] C. W. BROWN et B. J. SHEPHERD : *Graphics File Formats, reference and guide*. Manning, 1995.
- [35] D. BROWN : *Da Vinci Code*. Jean-Claude Lattès, 2004. ISBN : 2709624931.
- [36] D. BROWN : *Forteresse Digitale*. Jean-Claude Lattès, février 2007. ISBN : 2709626306.
- [37] G. BUREL et R. GAUTIER : Blind estimation of encoder and interleaver characteristics in a non cooperative context. *In Proc. IASTED International Conference on Communications, Internet and Information Technology*, Scottsdale, AZ, USA, novembre 2003.
- [38] C. CACHIN : An information-theoretic model for steganography. *In* D. AUCSMITH, éditeur : *Proc. Information Hiding, 2nd International Workshop*, volume 1525 de *Lecture Notes in Computer Science*, pages 306–318, Portland, Oregon, USA, avril 1998. Springer.
- [39] C. CACHIN : An information-theoretic model for steganography. *Information and Computation*, 192(1):41–56, juillet 2004.
- [40] C. CACHIN : Digital steganography. *In* H.C.A. van TILBORG, éditeur : *Encyclopedia of Cryptography and Security*. Springer, 2005. ISBN : 978-0-387-23473-1.
- [41] J.B. CAIN, G.C. CLARK et J.M. GEIST : Punctured convolutional codes of rate $\frac{n-1}{n}$ and simplified maximum likelihood decoding. *IEEE Transactions on Information Theory*, IT-25, janvier 1979.
- [42] A. CANTEAUT et F. CHABAUD : A new algorithm for finding minimum-weight words in a linear code : application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, IT-44(1):367–378, janvier 1998.
- [43] F. CHABAUD : *Recherche de performance dans l'algorithmique des corps finis. Applications à la cryptographie*. Thèse de doctorat, École Polytechnique, Palaiseau, France, octobre 1996.
- [44] R. CHANDRAMOULI : Data hiding capacity in the presence of an imperfectly known channel. *In Proc. SPIE Security and Watermarking of Multimedia Contents II*, volume 4314, 2001.
- [45] R. CHANDRAMOULI : Mathematical theory for steganalysis. *In Proc. SPIE Security and Watermarking of Multimedia Contents IV*, 2002.

- [46] R. CHANDRAMOULI, M. KHARRAZI et N.D. MEMON : Image steganography and steganalysis : Concepts and practice. In T. KALKER, I. J. COX et Y. M. RO, éditeurs : *Proc. Digital Watermarking, Second International Workshop, IWDW 2003*, volume 2939 de *Lecture Notes in Computer Science*, pages 35–49, Seoul, Korea, octobre 2003. Springer. ISBN : 3-540-21061-X.
- [47] R. CHANDRAMOULI et N.D. MEMON : Steganography capacity : A steganalysis perspective. In *Proc. SPIE, Security and Watermarking of Multimedia Contents V*, volume 5020, pages 173–177, Santa Clara, CA, USA, janvier 2003.
- [48] M. CHAPMAN : *Hiding the Hidden : A Software System for Concealing Ciphertext in Innocuous Text*. Thèse de doctorat, The University of Wisconsin-Milwaukee, mai 1997.
- [49] M. CLUZEAU : Block code reconstruction using iterative decoding techniques. In *Proc. 2006 IEEE International Symposium on Information Theory, ISIT06*, Seattle, USA, juillet 2006.
- [50] M. CLUZEAU : Reconnaissance d'un code linéaire en bloc en utilisant un algorithme de décodage itératif. In *Journées Codage et Cryptographie*, Eymoutiers, France, octobre 2006.
- [51] M. CLUZEAU : *Reconnaissance d'un schéma de codage*. Thèse de doctorat, École Polytechnique, Palaiseau, France, novembre 2006.
- [52] J. CLÉMENT : *Arbres Digitaux et Sources Dynamiques*. Thèse de doctorat, Université de Caen, 2000.
- [53] J. CLÉMENT, P. FLAJOLET et B. VALLÉE : Dynamical sources in information theory : a general analysis of trie structures. Rapport technique 3645, INRIA Roquencourt, 1999.
- [54] D. COPPERSMITH : Solving linear equations over $\text{GF}(2)$: Block Lanczos algorithm. *Linear Algebra and its Applications*, 192:33–60, janvier 1993.
- [55] D. COPPERSMITH : Solving homogeneous linear equations over $\text{GF}(2)$ via block Wiedemann algorithm. *Mathematics of Computation*, 62(205):333–350, 1994.
- [56] R. CRANDRALL : Some notes on steganography. Posted on Steganography Mailing List, 1998. <http://os.inf.tu-dresden.de/~westfeld/crandall.pdf>.
- [57] N.G. de BRUIJN : *Asymptotic Methods in Analysis*. Dover Publications, novembre 1981. ISBN : 0486642216.
- [58] N. DEDIĆ, G. ITKIS, L. REYZIN et S. RUSSEL : Upper and lower bounds on black-box steganography. In *Proc. 2nd Theory of Cryptography Conference (TCC 2005)*, volume 3378 de *Lecture Notes in Computer Science*. Springer, 2005.
- [59] S. DUMITRESCU, X. WU et Z. WANG : Detection of LSB steganography via sample pair analysis. In Fabien A. P. PETITCOLAS, éditeur : *Proc. Information Hiding, 5th International Workshop*, volume 2578 de *Lecture Notes in Computer Science*, pages 355–372, Noordwijkerhout, The Netherlands, octobre 2002. Springer. ISBN : 3-540-00421-1.
- [60] EBU/CENELEC/ETSI JOINT TECHNICAL COMMITTEE BROADCAST : Framing structure channel coding and modulation for terrestrial services. Specifications. ETS 300-744.
- [61] EBU/CENELEC/ETSI JOINT TECHNICAL COMMITTEE BROADCAST : Interaction channel for DVB-S RCS. Specifications. EN 300-958.

- [62] EBU/CENELEC/ETSI JOINT TECHNICAL COMMITTEE BROADCAST : Interaction channel for DVB-T RCT. Specifications. EN 300-958.
- [63] EBU/CENELEC/ETSI JOINT TECHNICAL COMMITTEE BROADCAST : Framing structure channel coding and modulation for satellites services. Specifications, 1999. ETS 300-421.
- [64] P. ELIAS : Coding for noisy channels. In *1955 IRE International Convention Record (part 4)*, pages 37–46, 1955.
- [65] ETSI : Broadband radio acces network (BRAN); HYPERLAN type 2, physical (phy) layer. Specifications, octobre 2001. ETSI TS 101-475 V1.3.1A.
- [66] R.M. FANO : A heuristic discussion of probabilistic decoding. *IEEE Transactions on Information Theory*, IT-9:64–74, avril 1963.
- [67] H. FARID : Detecting hidden messages using higher-order statistical models. In *ICIP (2)*, pages 905–908, 2002.
- [68] H. FEISTEL : Cryptography and computer privacy. *Scientific American*, 228(5):15–23, 1973.
- [69] É. FILIOL : Reconstruction of convolutional encoders over $GF(q)$. In M. DARNELL, éditeur : *Proc. 6th IMA Conference on Cryptography and Coding*, numéro 1355 de Lecture Notes in Computer Science, pages 100–110. Springer Verlag, 1997.
- [70] É. FILIOL : Reconstruction of punctured convolutional encoders. In T. FUJIWARA, éditeur : *Proc. 2000 International Symposium on Information Theory and Applications*, pages 4–7. SITA and IEICE Publishing, 2000.
- [71] É. FILIOL : *Techniques de reconstruction en cryptologie et théorie des codes*. Thèse de doctorat, École Polytechnique, Palaiseau, France, mars 2001.
- [72] P. FLAJOLET, X. GOURDON et P. DUMAS : Mellin transforms and asymptotics : Harmonic sums. *Theoretical Computer Science*, 144(1-2):3–58, 1995.
- [73] C. FONTAINE : *Contribution à la recherche de fonctions booléennes hautement non linéaires, et au marquage d'images en vue de la protection des droits d'auteur*. Thèse de doctorat, Université Paris VI, novembre 1998.
- [74] C. FONTAINE et F. GALAND : How can Reed-Solomon codes improve steganographic schemes. In *Proc. International Workshop on Information Hiding, IH'07*, Lecture Notes in Computer Science, Saint-Malo, France, juin 2007. Springer.
- [75] J. FRIDRICH : Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes. In *Information Hiding, 6th International Workshop, IH 2004*, volume 3200 de *Lecture Notes in Computer Science*, pages 67–81, Toronto, Canada, mai 2004. Springer. ISBN : 3-540-24207-4.
- [76] J. FRIDRICH, M. GOLJAN et R. DU : Detecting LSB steganography in color and gray-scale images. *IEEE MultiMedia*, 8(4):22–28, 2001.
- [77] J. FRIDRICH, M. GOLJAN et R. DU : Reliable detection of LSB steganography in grayscale and color images. In *Proc. ACM Workshop on Multimedia and Security*, pages 27–30, Ottawa, Canada, octobre 2001.
- [78] J. FRIDRICH, M. GOLJAN et D. HOGGA : Steganalysis of JPEG images : breaking the F5 algorithm. In *Proc. Information Hiding, 5th International Workshop, IH 2002*, volume 2578 de *Lecture Notes in Computer Science*, pages 310–323, Noordwijkerhout, The Netherlands, octobre 2002. Springer. ISBN : 3-540-00421-1.

- [79] J. FRIDRICH, M. GOLJAN et D. HOGGA : New methodology for breaking steganographic techniques for JPEGs. *In Proc. SPIE, Security and Watermarking of Multimedia Contents V*, volume 5020, pages 143–155, Santa Clara, CA, USA, janvier 2003.
- [80] J. FRIDRICH, M. GOLJAN, P. LISONEK et D. SOUKAL : Writing on wet paper. *IEEE Transactions on Signal Processing*, 53(10):3923–3935, 2005. Special issue "Supplement on Secure Media III".
- [81] J. FRIDRICH, M. GOLJAN et D. SOUKAL : Efficient wet paper codes. *In Proc. 7th International Workshop on Information Hiding*, volume 3727 de *Lecture Notes in Computer Science*, pages 204–218. Springer, 2005.
- [82] J. FRIDRICH, M. GOLJAN et D. SOUKAL : Wet paper codes with improve embedding efficiency. *IEEE Transactions on Security and Forensics*, 1(1):102–110, 2006.
- [83] J. FRIDRICH et T. PEVNY : Multiclass blind steganalysis for JPEG images. *In Proc. SPIE, Security and Watermarking of Multimedia Contents VIII*, janvier 2006.
- [84] J. FRIDRICH et D. SOUKAL : Matrix embedding for large payloads. *IEEE Transactions on Security and Forensics*, 1(3):278–294, 2006.
- [85] F. GALAND et G. KABATIANSKY : Information hiding by coverings. *In Proc. ITW'03*, pages 151–154, 2003.
- [86] Jr. G.D. FORNEY : *Concatenated Codes*. MIT Press, décembre 1966. ISBN : 0262060159.
- [87] Jr. G.D. FORNEY : Convolutional codes I : Algebraic structure. *IEEE Transactions on Information Theory*, IT-16:720–738, novembre 1970.
- [88] Jr. G.D. FORNEY : Structural analysis of convolutional codes via dual codes. *IEEE Transactions on Information Theory*, IT-19:512–519, juillet 1973.
- [89] Jr. G.D. FORNEY : The Viterbi algorithm. *In Proc. IEEE*, volume 61, pages 268–276, mars 1973.
- [90] Jr. G.D. FORNEY : Minimal bases of rational vector spaces with applications to multivariable linear systems. *In SIAM J. Control*, volume 13, pages 493–502, 1975.
- [91] E.N. GILBERT : Capacity of a burst-noise channel. *Bell System Technical Journal*, 39:1253–1265, septembre 1960.
- [92] A. GLAVIEUX : *Channel Coding in Communication Networks : From Theory to Turbo Codes*. Digital Signal and Image Processing series. Iste Publishing Company, novembre 2005. ISBN : 190520924X.
- [93] I. GOHBERG, T. KAILATH et V. OLSHEVSKY : Fast gaussian elimination with partial pivoting for matrices with displacement structure. *Mathematics of Computation*, 64:1557–1576, 1995.
- [94] I. GOLDBERG, éditeur. *Operator Theory : Advances and Applications.*, volume 18, pages 31–88. Birkhäuser, Boston, 1986.
- [95] S. GOLDWASSER et S. MICALI : Probabilistic encryption. *Journal of Computer and System Science*, 28:270–299, 1984.
- [96] D.W. HAGELBARGER : Recurrent codes : Easily mechanized, burst-correcting binary codes. Rapport technique J. 38, Bell Syst., juillet 1959.
- [97] J. HAGENAUER : Rate compatible punctured convolutional codes and their applications. *IEEE Transactions on Communications*, COMM-36, avril 1988.

- [98] J.J. HARMSSEN et W. A. PEARLMAN : Kernel fisher discriminant for steganalysis of JPEG hiding methods. In *ACM Multimedia and Security*, New York, USA, août 2005.
- [99] N.J. HOPPER : *Toward a Theory of Steganography*. Thèse de doctorat, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, juillet 2004.
- [100] N.J. HOPPER : On steganographic chosen ciphertext security. In *Proc. International Colloquium on Automata Languages and Programming, ICALP 2005*, volume 3580 de *Lecture Notes in Computer Science*, pages 311–323, Lisboa, Portugal, 2005. Springer. ISBN : 3-540-27580-0.
- [101] N.J. HOPPER, J. LANGFORD et L. von AHN : Provably secure steganography. In M. YUNG, éditeur : *Proc. Crypto 2002*, volume 2442 de *Lecture Notes in Computer Science*, pages 77–92, Santa Barbara, CA, USA, août 2002. Springer. ISBN : 3-540-44050-X.
- [102] D.A. HUFFMAN : A method for the construction of minimum redundancy codes. In *Proc. IRE*, volume 40, pages 1098–1101, septembre 1952.
- [103] W.C. HUFFMAN et V. PLESS : *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003. ISBN : 0521782805.
- [104] IEEE : Part 11 : Wireless lan medium access control (MAC) and physical layer (PHY) specifications : High-speed physical layer in the 5 GHz band. Specifications, 1999. IEEE Std 802.11a.
- [105] É. INCERTI : *Compression d'image. Algorithmes et standards*. Vuibert, 2003. ISBN : 2-7117-4815-4.
- [106] INTELSAT EARTH STATION STANDARDS : Performance characteristics for intermediate data rate digital carriers using convolutional encoding/Viterbi encoding and QPSK modulation. Specifications, novembre 1998. IESS-308 (Rev. 9).
- [107] INTELSAT EARTH STATION STANDARDS : Performance characteristics for intermediate data rate digital carriers using rate $\frac{3}{2}$ TCM/8PSK and Reed-Solomon outer coding (TCM/IDR). Specifications, février 2000. IESS-310 (Rev. 2).
- [108] F. JELINEK : A fast sequential decoding algorithm using a stack. Rapport technique 13, IBM Res. Develop., novembre 1969.
- [109] R. JOHANNESSON et K.Sh. ZYANGIROV : *Fundamentals of Convolutional Coding*. IEEE Press, 1999.
- [110] N.F. JOHNSON, Z. DURIC et S. JAJODIA : *Information Hiding - Steganography and watermarking - Attacks and countermeasures*. Advances in Information Security. Kluwer Academic. ISBN : 0-7923-7204-2.
- [111] N.F. JOHNSON et S. JAJODIA : Exploring steganography : Seeing the unseen. *IEEE Computer*, 31(2):26–34, 1998.
- [112] J.C. JUDGE : *Steganography : Past, Present and Future*. SANS, 2001.
- [113] D. KAHN : *The Codebreakers*. MacMillan, New York, 1967.
- [114] S. KATZENBEISSER et F.A.P. PETITCOLAS : *Information Hiding. Techniques for steganography and digital watermarking*. Computer Science. Artech House. ISBN : 1-5853-035-4.
- [115] S. KATZENBEISSER et F.A.P. PETITCOLAS : Defining security in steganographic systems. In *Proc. SPIE Security and Watermarking of Multimedia contents IV*, volume 4675, pages 50–56, 2002.

- [116] J. KELLEY : Terror groups hide behind Web encryption. *USA Today*, mai 2001.
- [117] J. KELLEY : Terrorist instructions hidden online. *USA Today*, mai 2001.
- [118] A.D. KER : Improved detection of LSB steganography in grayscale images. In J. FRIDRICH, éditeur : *Proc. Information Hiding, 6th International Workshop*, volume 3200 de *Lecture Notes in Computer Science*, pages 97–115, Toronto, Canada, mai 2004. Springer. ISBN : 3-540-24207-4.
- [119] A.D. KER : The ultimate steganalysis benchmark? In *MM&Sec '07 : Proceedings of the 9th workshop on Multimedia & security*, pages 141 – 148, Dallas, Texas, USA, septembre 2007. ACM. ISBN : 978-1-59593-857-2.
- [120] A. KERCKHOFFS : La cryptographie militaire. *Journal des Sciences Militaires*, février 1883.
- [121] G.C. KESSLER : Steganography : Implications for the prosecutor and computer forensics examiner. Rapport technique, American Prosecutors Research Institute, avril 2004.
- [122] G. KIPPER : *Investigator's guide to steganography*. Information Security. Auerbach, 2004. ISBN : 0-8493-2433-5.
- [123] K.J. LARSEN : Short convolutional codes with maximal free distance for rates $\frac{1}{2}$, $\frac{1}{3}$ and $\frac{1}{4}$. *IEEE Transactions on Information Theory*, IT-18:371–372, mai 1973.
- [124] A. LATHAM : Steganography : JPHIDE AND JPSEEK, 1999. <http://linux01.gwdg.de/~alatham/stego.html>.
- [125] A. LAUDER et K. PATERSON : Computing the error linear complexity spectrum of a binary sequence of period 2^n . *IEEE Transactions on Information Theory*, IT-49:273–280, 2003.
- [126] G.S. LAUER : Some optimal partial-unit-memory codes. *IEEE Transactions on Information Theory*, IT-25:240–243, mars 1979.
- [127] J. Sam LEE et L.E. MILLER : *CDMA Systems Engineering Handbook*. Mobile Communications. Artech House Publishers. ISBN : 0-89006-990-5.
- [128] P.J. LEE et E.F. BRICKELL : An observation on the security of McEliece's public-key cryptosystem. In *Advances in Cryptology - EUROCRYPT'88*, volume 330 de *Lecture Notes in Computer Science*, pages 275–280. Springer Verlag, 1988.
- [129] J.S. LEON : A probabilistic algorithm for computing the minimum weight of large error-correcting codes. *IEEE Transactions on Information Theory*, IT-34(5):1354–1359, septembre 1988.
- [130] T.V. LEVAN et K. KUROSAWA : Efficient public key steganography secure against adaptative chosen stegotext attacks. In *Proc. Information Hiding, 8th International Workshop*, Old Town Alexandria, Virginia, USA, juillet 2006.
- [131] N. LEVINSON : The Weimer RMS criterion in filter design and prediction. *Journal of Math. Phy.*, 25:261–278, 1947.
- [132] R. LIDL et H. NIEDERREITER : *Finite Fields*. Cambridge University Press, 1983.
- [133] G.S. LIN, C.H. YEH et C.C.J. KUO : Data hiding domain classification for blind image steganalysis. In *Proc. ICME*, pages 907–910, 2004.
- [134] P. LU, X. LUO, Q. TANG et L. SHEN : An improved sample pairs method for detection of LSB embedding. In J. FRIDRICH, éditeur : *Proc. Information Hiding, 6th International Workshop*, volume 3200 de *Lecture Notes in Computer Science*, pages 116–127, Toronto, Canada, mai 2004. Springer. ISBN : 3-540-24207-4.

- [135] S. LYU et H. FARID : Detecting hidden messages using higher-order statistics and support vector machines. In *Proc. Information Hiding, 5th International Workshop, IH 2002*, volume 2578 de *Lecture Notes in Computer Science*, pages 340–354, Noordwijkerhout, The Netherlands, octobre 2002. Springer. ISBN : 3-540-00421-1.
- [136] S. LYU et H. FARID : Steganalysis using color wavelet statistics and one-class support vector machines. In *Proc. SPIE, Security and Watermarking of Multimedia Contents VI*, San Jose, CA, USA, 2004.
- [137] S. LYU et H. FARID : Steganalysis using higher-order image statistics. *IEEE Transactions on Information Forensics and Security*, 1, 2006.
- [138] F.J. MACWILLIAMS et N.J.A. SLOANE : *The theory of error-correcting codes*. North-Holland, 1977.
- [139] J.L. MASSEY : *Threshold decoding*. MIT Press, Cambridge, MA, 1963.
- [140] J.L. MASSEY : Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, IT-15:122–127, 1969.
- [141] J.L. MASSEY et M.K. SAIN : Inverses of linear sequential circuits. *IEEE Transactions on Computers*, COM-17:330–337, 1968.
- [142] K. MAYOURA : Analyse stéganographique d’une image JPEG. Mémoire de DESS, Université du mans, 2004.
- [143] R.J. MCELIECE : *Handbook of coding theory.*, volume 2, chapitre 12, The algebraic theory of convolutional codes, pages 1065–1138. Elsevier Science, 1998.
- [144] W. MEIER, E. PASALIC et C. CARLET : Algebraic attacks and decomposition of Boolean functions. In C. CACHIN et J. CAMENISCH, éditeurs : *Proc. Eurocrypt 2004*, numéro 3027 de *Lecture Notes in Computer Science*, pages 474–491, Interlaken, Switzerland, 2004. Springer Verlag.
- [145] P.L. MONTGOMERY : A block Lanczos algorithm for finding dependencies over $GF(2)$. In *Advances in Cryptology - EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques*, volume 921 de *Lecture Notes in Computer Science*, pages 106–120. Springer Verlag, mai 1995.
- [146] P. MÜHLETHALER : *802.11 et les réseaux sans fils*. Editions Eyrolles, 2002. ISBN : 2-212-11154-1.
- [147] B.C. NGUYEN, S.M. YOON et H.-K. LEE : Multi bit plane image steganography. In Y. Q. SHI et B. JEON, éditeurs : *Proc. Digital Watermarking, 5th International Workshop, IWDW 2006*, volume 4283 de *Lecture Notes in Computer Science*, pages 61–70, Jeju Island, Korea, novembre 2006. Springer.
- [148] J.P. ODENWALDER : *Optimal decoding of convolutional codes*. Thèse de doctorat, UCLA System Sciences Dept., Los Angeles, 1970.
- [149] C. O’DONOGHUE et C. BURKLEY : New algorithm to identify rate $\frac{k}{n}$ catastrophic punctured convolutional encoders. In *Proc. of the Workshop in Coding and Cryptography 99*, 1999.
- [150] S. OHMORI, H. WAKANA et S. KAWASE : *Mobile Satellite Communications*. Artech House, 1998. ISBN : 0-89006-843-7.
- [151] J.K. OMURA : On the Viterbi decoding algorithm. *IEEE Transactions on Information Theory*, IT-15:177–179, janvier 1969.

- [152] E. PAASKE : Short binary codes with maximal free distance. *IEEE Transactions on Information Theory*, IT-20:683–688, septembre 1974.
- [153] B. PFITZMANN : Information hiding terminology. In *Proceedings of the Workshop on Information Hiding*, numéro 1174, pages 347–350, Cambridge, England, mai 1996. Springer Verlag.
- [154] G. PLANQUETTE : *Identification de trains binaires codés*. Thèse de doctorat, Université de Rennes I, France, décembre 1996.
- [155] N. PROVOS : Universal steganography., août 1998. <http://www.outguess.org/>.
- [156] N. PROVOS : Defending against statistical steganalysis. In *10th USENIX Security Symposium*, Washington, DC, USA, 2001.
- [157] N. PROVOS et P. HONEYMAN : Detecting steganographic content on the internet. In *Proc. ISOC NDSS'02*, San Diego, CA, USA, février 2002.
- [158] S. RAUDYS : *Statistical and Neural Classifiers : An Integrated Approach to Design*. Advances in Pattern Recognition. Springer-Verlag, janvier 2001. ISBN : 1852332972.
- [159] F. RAYNAL, F. PETITCOLAS et C. FONTAINE : L'art de dissimuler les informations. *Pour la Science*, été 2002. Dossier “ L'art du secret ”.
- [160] J.A. REEDS : Solved : The ciphers in book III of Trithemius's *Steganographia*. *Cryptologia*, (22):291–319, octobre 1998.
- [161] E. RENOLD, S.J. CREIGHTON, C. ATKINSON et J. CARR : Images of abuse : A review of the evidence on child pornography. Rapport technique, National Society for the Prevention of Cruelty to Children (NSPCC), octobre 2003.
- [162] B. RICE : Determining the parameters of a rate $\frac{1}{n}$ convolutional encoder over $GF(q)$. In *Proc. Third International Conference on Finite Fields and Applications*, Glasgow, 1995.
- [163] A. SALAGEAN : On the computation of the linear complexity and k -error linear complexity of binary sequences with period a power of two. *IEEE Transactions on Information Theory*, 51:1145–1150, 2005.
- [164] G. SAPORTA : *Probabilité, Analyse des Données et Statistiques*. Technip, 1990.
- [165] L. SCHIFF et A. CHOCKALINGAM : *Wireless Networks 6*, chapitre Design and system operation of Globalstar versus IS-95 CDMA - Similarities and differences, pages 47–57. Science Publishers, 2000.
- [166] B. SCHNEIER : Description of a new variable-length key, 64-bits block cipher (Blowfish). In *Proc. Fast Software Encryption, Cambridge Security Workshop*, pages 191–204. Springer-Verlag, décembre 1993.
- [167] D. SCHÖNEFELD et A. WINKLER : Embedding with syndrome coding based on BCH codes. In *Proc. ACM Multimedia and Security Workshop 2006*, pages 214–223. ACM, 2006.
- [168] D. SCHÖNEFELD et A. WINKLER : Reducing the complexity of syndrome coding for embedding. In *Proc. International Workshop on Information Hiding, IH'07*, Lecture Notes in Computer Science, Saint-Malo, France, juin 2007. Springer.
- [169] RSA Data SECURITY : The RC4 encryption algorithm., mars 1992.
- [170] R. SEDGEWICK et P. FLAJOLET : *Introduction à l'analyse d'algorithme*, chapitre Chaînes et arbres digitaux. Thomson publisher, 1996.

- [171] C.E. SHANNON et W. WEAVER : *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, 1949.
- [172] I. SHUR : Über potenzreihen, die im Inneren des Einheitskreises beschränkt sind. *Journal für die Reine und Angewandte Mathematik.*, 147:205–232, 1917. English translation in [94].
- [173] G. SICOT et S. HOUCKE : Blind detection of interleaver parameters. *In Proc. ICASSP 2005*, Philadelphia, USA, 2005.
- [174] G. SICOT et S. HOUCKE : Etude statistique du seuil dans la détection d'entrelaceur. *In Proc. GRETSI 2005*, Louvain la Neuve, Belgium, 2005.
- [175] G. SICOT et S. HOUCKE : Theoretical study of the performance of a blind interleaver estimator. *In Proc. ISIVC 2006*, Hammamet, Tunisia, 2006.
- [176] D. SIEBERG : Bin Laden exploits technology to suit his needs. *CNN*, septembre 2001.
- [177] G.J. SIMMONS : The prisoners' problem and the subliminal channel. *In Proc. CRYPTO'83*, pages 51–67, 1983.
- [178] H.J.S SMITH : On systems of linear indeterminate equations and congruences. *Philos. Trans. Roy. Soc. London*, (151):293–326, 1861.
- [179] M. STAMP et C. MARTIN : An algorithm for the k -error linear complexity of binary sequences of period 2^n . *IEEE Transactions on Information Theory*, 39:1398–1401, 1993.
- [180] N. STEPHENSON : *Le Cryptonomicon*. Payot, avril 2000.
- [181] J. STERN : A method for finding codewords of small weight. *In G. COHEN et J. WOLFMANN, éditeurs : Proc. Coding Theory and Applications*, numéro 388 de Lecture Notes in Computer Science, pages 106–113, Toulon, France, 1989. Springer Verlag.
- [182] P-T. SUN : Similarity of discrete Gilbert-Elliott and Polya channel models to continuous Rayleigh fading channel model. Master thesis, National Chiao Tung University, Taiwan, juin 2002. <http://shannon.cm.nctu.edu.tw/html/thesis/pen-ting-s.pdf>.
- [183] TECHNICAL SPECIFICATION GROUP RADIO ACCESS NETWORK : Multiplexing and channel coding (TDD) (release 6). Specifications, 3rd Generation Partnership Project, décembre 2004. 3GPP TS 25.222 V6.2.0.
- [184] TECHNICAL SPECIFICATION GROUP RADIO ACCESS NETWORK : Multiplexing and channel coding (FDD) (release 6). Specifications, 3rd Generation Partnership Project, juin 2005. 3GPP TS 25.212 V6.5.0.
- [185] TECHNICAL SPECIFICATION GROUP RADIO ACCESS NETWORK : Physical channels and mapping of transport channels onto physical channels (FDD) (release 6). Specifications, 3rd Generation Partnership Project, juin 2005. 3GPP TS 25.211 V6.5.0.
- [186] TECHNICAL SPECIFICATION GROUP RADIO ACCESS NETWORK : Physical channels and mapping of transport channels onto physical channels (TDD) (release 6). Specifications, 3rd Generation Partnership Project, juin 2005. 3GPP TS 25.221 V6.4.1.
- [187] TELECOMMUNICATIONS INDUSTRY ASSOCIATION : Mobile station - base station compatibility standard for dual-mode wideband spread spectrum cellular system. Specifications, 3rd Generation Partnership Project, 1993. TIA/EIA/IS-95.

- [188] TELECOMMUNICATIONS INDUSTRY ASSOCIATION : Physical layer standard for CDMA2000 spread spectrum systems. Specifications, 3rd Generation Partnership Project, août 1999. TIA/EIA/IS-2000-2.
- [189] TELECOMMUNICATIONS INDUSTRY ASSOCIATION : Physical layer standard for CDMA2000 spread spectrum systems (release 0). Specifications, 3rd Generation Partnership Project, juin 2001. 3GPP2 C.S 0002 V3.0.
- [190] TELECOMMUNICATIONS INDUSTRY ASSOCIATION : Physical layer standard for CDMA2000 spread spectrum systems (revision). Specifications, 3rd Generation Partnership Project, avril 2002. TIA/EIA/IS - 2000.2-A-2.
- [191] Technical Specification Group GSM/EDGE Radio Access NETWORK : Channel coding (release 1999). Specifications, 3rd Generation Partnership Project, janvier 2005. 3GPP TS 05.03 V8.9.0.
- [192] E. THOMÉ : Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm. *Journal of Symbolic Computation*, 33(5):757–775, juillet 2002.
- [193] D. UPHAM : Jsteg, 1997. <http://zooid.org/paul/crypto/jsteg/>.
- [194] P.P. VAIDYANATHAN : *Multirate Systems and Filter Banks*, chapitre 13. Englewood Cliffs, 1993.
- [195] A. VALEMBOIS : *Détection, reconnaissance et décodage de codes linéaires binaires*. Thèse de doctorat, Université de Limoges, France, septembre 2000.
- [196] A. VALEMBOIS : Detection and recognition of a binary linear code. *Discrete Applied Mathematics*, 111:199–218, 2001.
- [197] A.J. VITERBI : Error bounds for convolutional codes and asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–269, avril 1967.
- [198] L. von AHN et N. J. HOPPER : Public-key steganography. In C. CACHIN et J. CAME-NISCH, éditeurs : *Proc. Eurocrypt 2004*, volume 3027 de *Lecture Notes in Computer Science*, pages 323–341, Interlaken, Switzerland, mai 2004. Springer. ISBN : 3-540-21935-8.
- [199] B. VUCETIC et J. YUAN : *Turbo Codes : Principles and Applications*. Kluwer Academic Publishers, janvier 2000. ISBN : 0792378687.
- [200] G.K. WALLACE : The JPEG still picture compression standard. *Commun. ACM*, 34(4):30–44, 1991.
- [201] P. WAYNER : *Disappearing cryptography - Information Hiding : steganography & watermarking*. Morgan Kaufmann, 2002. ISBN : 1-55860-769-2.
- [202] A. WESTFELD : The steganographic algorithm F5., 1999. <http://www.rn.inf.tu-dresden.de/~westfeld/f5.html>.
- [203] A. WESTFELD : F5-a steganographic algorithm. In I.S. MOSKOWITZ, éditeur : *Proc. Information Hiding, 4th International Workshop, IHW 2001*, volume 2137 de *Lecture Notes in Computer Science*, pages 289–302, Pittsburgh, PA, USA, avril 2001. Springer. ISBN : 3-540-42733-3.
- [204] A. WESTFELD et A. PFITZMANN : Attacks on steganographic systems. In A. PFITZMANN, éditeur : *Proc. Information Hiding, Third International Workshop, IH'99*, volume 1768 de *Lecture Notes in Computer Science*, pages 61–76, Dresden, Germany, septembre 1999. Springer. ISBN : 3-540-67182-X.

- [205] D.H. WIEDEMANN : Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, IT-32:54–62, janvier 1986.
- [206] R. WONG : *Asymptotic Approximations of Integrals*. Academic Press, octobre 1989. ISBN : 0127625356.
- [207] J.M. WOZENCRAFT et B. REIFFEN : *Sequential decoding*. MIT Press, Cambridge, MA, 1961.
- [208] Y. YASUDA, K. KASHIKI et Y. HIRATA : High-rate punctured convolutional codes for soft decision Viterbi decoding. *IEEE Transactions on Communications*, COMM-32, 1984.
- [209] J. ZÖLLNER, H. FEDERRATH, H. KLIMANT, A. PFITZMANN, R. PIOTRASCHKE, A. WESTFELD, G. WICKE et Gritta WOLF : Modeling the security of steganographic systems. In D. AUCSMITH, éditeur : *Proc. Information Hiding. Second International Workshop, IH'98*, volume 1525 de *Lecture Notes in Computer Science*, pages 344–354, Portland, Oregon, USA, avril 1998. Springer-Verlag.

Glossaire

AFD : Analyse Factorielle Discriminante.

CCA1 : Choosen Cipher Attack.

CCA2 : Choosen Cipher Adaptative Attack.

CHA : Choosen Hidden text Attack.

CMA : Choosen Message Attack.

CPA : Choosen Plaintext Attack.

BMP : BitMaP format.

bpp : bit per pixel ou bit par pixel.

CBS : Canal Binaire Symétrique.

CDMA : Code Division Multiple Access.

CGC : Codage de Gray Canonique.

COMSEC : COMmunication SECurity.

DCT : Discret Cosinus Transform.

Densité d'une matrice : La densité d'une matrice est la proportion de coefficients non nuls de la matrice.

DFC : Domaine Fréquentiel Compressé.

DGA : Délégation Générale pour l'Armement.

Distance de Hamming : La distance de Hamming $d_H(x, y)$ de deux vecteurs de symboles x et y est donnée par le nombre de symboles qui diffèrent entre x et y . Si x et y sont binaires, elle correspond au poids de Hamming du *ou-exclusif* de x et y .

$$d_H(x, y) = w(x \oplus y) \text{ .}$$

DVB-T : Digital Video Broadcasting-Terrestrial.

DVB-S : Digital Video Broadcasting-Satellite.

ESAT : École Supérieure et d'Application des Transmissions.

FFT : Fast Fourier Transform.

GIF : Graphics Interchange FormatTM.

GPA : Générateur Pseudo-Aléatoire.

GSM : Global System for Mobile Communication.

IDCT : Inverse Discret Cosinus Transform.

INMARSAT : INternational MARitime SATellite organisation system.

INTELSAT : INternational TELEcommunications SATellite organization.

ISO : International Standards Organisation.

ITU : International Telecommunication Union.

IV : Init Vector.

IWDW : International Workshop on Digital Watermarking.

JPEG : Joint Picture Experts Group.

LDPC : Low Density Parity Check.

LSB : Least Significant Bit.

Matrice creuse : Une matrice binaire est dite creuse si sa densité est faible.

Matrice unimodulaire : Matrice carrée à coefficients dans $F(D)$ de déterminant appartenant à F^* .

MDS : Maximum Separable Distance.

MBPIS : Multi Bit Plane Image Steganography.

MEM : Multiple Embedding Method.

MGP : Matrice Génératrice Polynomiale.

MSM : Méthode de Stéganographie Multiple.

PA : Passive Attack.

PCPC : Matrice PolyCyclique PseudoCirculante.

PGCD : Plus Grand Commun Diviseur.

Poids de Hamming : Le poids de Hamming d'un vecteur binaire $b = (b_i)_{i=1\dots l}$ de longueur l , noté $w(b)$, est égal au nombre de bits b_i , pour $i = 1 \dots l$, égaux à 1.

$$w(b) = \sum_{i=1}^l b_i .$$

PPCM : Plus Petit Commun Multiple.

RLE : Run Length Encoding.

ROC : Receiver Operating Characteristic.

RVB : Rouge, Vert, Bleu.

SSA : Specific Steganalysis Attack.

SVM : Support Vector Machines.

TEB : Taux d'Erreur par Bit.

TIC : Technologies de l'Information et de la Communication.

TNT : Télévision Numérique Terrestre.

TRANSEC : TRANsmission SECurity.

UMTS : Universal Mobile Telecommunication System.

USA : Universal Steganalysis Attack.

VLC : Variable Length Coding.

YCbCr : Luminance, Chrominance bleue, Chrominance rouge.

Index

Symboles

802.11 258

A

acrostiche 136

algorithme

de Berlekamp-Massey . 88, 95, 105–106, 113

de Gauss randomisé .. 57–60, 65, 70–77, 87, 105, 108

de Huffman 158

de Levinson 113

de Shur 113

de Smith 36

de Viterbi 26, 116

analyse

discriminante 189–194

factorielle discriminante 191

analyse du train binaire 15

attaquant

adaptatif 179

CCA1 179

CCA2 179

CHA 181

CMA 181

CPA 179

PA voir attaquant passif

passif 181

spécifique 182–184

universel 185–186

attaque par discrimination 176

B

bande fondamentale 261, 268

BMP 144, 299

borne de Shannon 173

bpp voir taux stéganographique

burst d'erreurs 21

C

canal

à évanouissement 21

additif gaussien 21

binaire 20

binaire symétrique 21

burst 21

coopératif 14

de Rayleigh voir canal à évanouissement, 115

de type burst 21

gaussien 21

non coopératif 14

subliminal 137, 138

capacité 140

CDMA 299

CDMA 2000 256–257

chaîne de communication 19, 20

chrominance 151

codage

arithmétique 157

de canal 19

de Gray canonique 147

de Huffman 157–159

de longueur variable 157

par syndrome 163–167

préfixé 157

RVB 144, 300

YCbCr 151, 300

code

convolutif 25–45, 87–113, 254–259

cyclique 253–259

dual 50

en bloc linéaire 26, 27, 49–86

MDS 167

- optimal..... 38
- parent 39, 112
- parfait..... 167
- poinçonné..... 39–45, 112
- récurrent voir code convolutif
- codeur
 - récuratif..... 31, 112
 - systématique..... 33, 112
- coefficient
 - AC 154–159
 - DC 154–159
 - DCT 154
 - DCT quantifié 154–157
- compression avec perte 152
- consistance 97
- contexte
 - coopératif..... 20, 132
 - non coopératif..... 20
- critère
 - d’avalanche 212, 214, 235
 - du rang..... 63
 - du rang randomisé..... 65
- cube RVB 144
- D**
 - décomposition
 - polynomiale 42
 - polyphase 43
 - DCT ... voir transformée en cosinus discret
 - degré
 - d’un code convolutif 33
 - d’un codeur..... 27
 - d’une MGP 33, 94
 - externe 33, 96
 - interne..... 33, 91
 - densité
 - d’un vecteur 54, 59
 - d’une matrice..... 41, 54, 60, 299
 - spectrale de puissance du bruit 21
 - DFC ... voir domaine fréquentiel compressé
 - diagramme RS..... 198
 - discriminant de Fisher... 203–207, 217–218, 232–233
 - dissimulation 138
 - distance
 - de Kullbak-Liebler..... 216
 - de Kullbak-Liebler..... 174
 - de Hamming..... 54, 299
 - libre 38
 - minimale 38, 167
 - statistique..... 176
- distingueur 177
- distribution
 - binomiale 55, 56, 125
 - de Rayleigh-Rice..... 21
 - gaussienne..... 21
- domaine
 - fréquentiel 152–156
 - fréquentiel compressé..... 142, 212–215
 - spatial..... 153
- DVB-S..... 258, 299
- DVB-T..... 258–259, 299
- E**
 - ε -sûr 174
 - effondrement 163
 - encre sympathique..... 135
 - entopie
 - relative 216
 - entrelaceur bloc 50–51, 112, 120–129
 - entropie
 - binaire 214
 - relative 174
 - erreur
 - de 1^{ère} espèce..... 187
 - espace de couleurs..... 144, 151–152
 - étalement de spectre..... 138
 - évasion de fréquence 138
 - extraction 139
- F**
 - F5..... 161–167
 - facteur
 - de consistance 98
 - de désynchronisation..... 52, 92
 - de précision 125
 - de qualité 154
 - discriminant 207, 218, 232
 - invariant..... 35
 - facteur discriminant 191
 - facteurs invariants 35
 - fonction
 - d’inversion..... 196, 202
 - de discrimination 196, 202
 - de Fisher 192
 - génératrice 28

- holomorphe.....261
- localement sommable.....261
- méromorphe.....262
- formule
 - de Cauchy.....264
 - de Stirling.....265, 266
- G**
- Générateur Pseudo-Aléatoire.....148
- GIF.....145, 299
- GLOBALSTAR.....258
- GPA.....voir Générateur Pseudo-Aléatoire
- grand maxima.....125
- GSM.....254–255, 300
- H**
- HYPERLAN-2.....258
- I**
- inégalité
 - de Hölder.....265
 - de Markov.....74
- indétectabilité.....140
- indices de Forney.....34
- indistingabilité indéniable.....141
- information souple.....115, 132
- INMARSAT.....257, 300
- insécurité
 - cryptographique.....180
 - stéganographique.....182
- INTELSAT.....257, 300
- IS95.....256
- ISO.....151, 300
- ITU.....151, 300
- J**
- JPEG.....151–159, 300
- JPHide and JPSeek.....168–170
- L**
- LDPC.....80–82, 300
- logiciels de stéganographie.....269–283
- longueur de contrainte.....33, 258, 259
- LSB.....300
- luminance.....151
- M**
- médium support voir support de couverture
- mémoire d'un codeur.....26, 33, 94
- Méthode de Stéganographie Multiple . 221–231
- méthode du col.....266
- métrique de Mahalanobis.....192
- malléabilité.....182
- masque.....196
- masque négatif.....197
- matrice
 - bruitée.....53, 57
 - creuse.....54, 300
 - d'erreur.....53, 54
 - de confusion.....187
 - de Hankel.....112
 - de poinçonnage.....41
 - de Toeplitz.....113
 - indicatrice de plus haut degré.....35
 - non bruitée.....54
 - polycyclique pseudocirculante.....43
 - unimodulaire.....34, 300
- matrice cumulative des votes.....125
- matrice génératrice.....29, 33–37
 - basique.....33, 99
 - canonique.....34, 96
 - catastrophique.....36–37
 - polynomiale.....32
 - réduite.....34
 - systématique.....33, 37
- MBPIS.....145–149, 300
- MEM.....voir Méthode de Stéganographie Multiple
- MGP . voir matrice génératrice polynomiale
- modèle d'indistingabilité.....178–182
- modèle de Gilbert-Elliott.....21
- mot
 - d'information.....26
 - de code.....26
- motif de poinçonnage.....41
- MSM.....voir Méthode de Stéganographie Multiple
- N**
- notation octale.....38
- noyau randomisé.....59, 70, 104
- O**
- Outguess.....159–161

P

pôle 262
 palette 144
 partie singulière 262
 PCPC voir polycyclique
 phase
 d'apprentissage 204–207, 217–218, 232–233
 de challenge 205, 218, 233
 pixel 144, 151
 poids d'une série de Laurent 28
 poids de Hamming 50, 85, 108, 216
 poinçonnage 41
 polynôme générateur 253
 précision 152
 probabilité
 de détection 68, 72, 75, 77–80, 108
 de fausse alarme 67–68, 70–73, 76–80, 108, 187
 de faux négatifs 187
 de faux positifs 187
 de non détection 76, 187
 de succès 188
 de vrais négatifs 187
 de vrais positifs 77, 187
 problème du prisonnier 137
 profondeur d'un entrelaceur 50, 112

Q

quantification 154–156
 quotient de Rayleigh 192

R

règle de Mahalanobis-Fisher 192
 rang
 normalisé 62–63
 randomisé 53–60, 104
 randomisé normalisé 64–65
 reconstruction
 d'algorithme 19
 des codes convolutifs 87–113
 des codes en blocs linéaires 49–86
 des turbo-codeurs 115–130
 regroupement 39, 41–45
 relation de MacWilliams 54
 relation de parité 50, 60–65, 70, 75, 108
 rendement 38, 63, 65

RLE voir Run Length Encoding
 robustesse 140
 ROC 77, 188, 205, 207, 208, 221, 233, 235, 300
 Run Length Encoding 156–157

S

sécurité
 de communication 138
 de transmission 138, 173
 séquence Zig-Zag 156
 série de Laurent 28, 262
 causale 28
 réalisable 28
 rationnelle 28
 schéma de stéganographie 138–139
 sensibilité 187
 sous-échantillonnage 152
 spécificité 187
 stéganalyse 137, 170–247
 RS 196–198
 RS locale 201–203
 spécifique 182–184, 195–210, 221–233
 universelle 15, 185–186, 215–218
 stéganalyse par discrimination 177
 stéganalyse RS 146
 stéganographie 135–247
 JPEG 159–170
 LSB 145
 stégo médium 138
 support voir support de couverture
 support de couverture 138
 SVM voir machine à support de vecteurs
 syndrome 166

T

table de quantification 154
 taux d'erreur binaire 21, 50, 60, 88, 123, 127, 132
 taux stéganographique 148–149
 TEB voir taux d'erreur binaire
 Technologies de l'Information et de la Communication 13
 théorème du rang 54, 61
 TIC voir Technologies de l'Information et de la Communication
 TNT 258–259, 300
 traitement aveugle 14

transformée
 de Fourier discrète 153
 de Mellin 261–262, 267–268
 de Poisson 263
 en z 28
 en cosinus discret 152–154
 en cosinus discret inverse 154
trie dynamique 117–120
turbo-code 115–130

U

UMTS 255–256, 300

V

valuation 28
variable
 discriminante 191
 explicative 190
variable centrée 192
variance
 interclasse 190
 intraclasse 191
vecteur statistique 176
VLC voir codage à longueur variable

Table des figures

1	Communication scheme	5
2	Schéma synthétique d'un système de communication	14
3	Chaîne de communication simplifiée	19
4	Chaîne de communication du point de vue de l'observateur	21
5	Diagramme d'état-transition du modèle de Gilbert-Elliot	22
1.1	Registre avec rétroaction	31
1.2	Représentation physique du codeur récursif de matrice génératrice G'	32
2.1	Répartition des mots de code dans un code de longueur 100	58
2.2	Algorithme de Gauss randomisé pour les matrices bruitées	59
2.3	Représentation de $C(n, d)$ et $C(n, d_e)$ avec deux relations de parité	61
2.4	Représentation de $C(n_e, d_e)$ avec deux relations de parité	62
2.5	Algorithme de reconstruction des codes en blocs linéaires dans un canal sans erreur	64
2.6	Algorithme de reconstruction des codes en blocs linéaires dans un canal avec erreur	66
2.7	Lois suivies par W	76
2.8	Courbes ROC en fonction du poids de la relation de parité	78
2.9	Probabilité de détection en fonction du seuil, paramètre w	78
2.10	Probabilité de fausse alarme en fonction du seuil, paramètre w	79
2.11	Probabilité de détection en fonction du seuil, paramètre N	79
2.12	Probabilité de fausse alarme en fonction du seuil, paramètre N	80
2.13	Probabilité de détection en fonction du seuil, paramètre ε	80
2.14	Probabilité de fausse alarme en fonction du seuil, paramètre ε	81
2.15	Reconstruction des codes cycliques	82
2.16	Reconstruction de codes LDPC	82
2.17	Nombre d'itérations en fonction de la longueur de codes cycliques	85
2.18	Nombre d'itérations en fonction de la longueur de codes LDPC	85
3.1	(n, k, m_e) -codeur convolutif	88
3.2	Algorithme de reconstruction des $(n, 1)$ -code convolutifs dans un canal sans erreur	96
3.3	Procédure de vérification de la consistance des solutions	99
3.4	Algorithme de résolution du système $(S_j)_{j=1\dots(n-k)}$ dans le cas sans erreur .	100

3.5	Algorithme de reconstruction des (n, k) -codes convolutifs dans le cas sans erreur	101
4.1	Schéma d'un turbo-codeur parallèle	116
4.2	Exemple de trie fini	119
4.3	$trie(X)$ et $trie(Y)$	121
4.4	Procédure de mise à jour de la liste des indices i de 1 à n	123
4.5	Algorithme de reconstruction des entrelaceurs dans le cas sans erreur	124
4.6	Algorithme de reconstruction des entrelaceurs dans le cas avec erreur	127
4.7	Probabilité d'arrêt de l'algorithme de reconstruction dans le cas sans erreur	128
4.8	Évolution du nombre d'observations en fonction de l'erreur pour $n = 256$.	130
4.9	Jean Trithème et <i>Steganographia</i>	136
4.10	Correspondance entre George Sand et Alfred de Musset	137
4.11	Étape de dissimulation pour une image fixe	139
4.12	Étape d'extraction pour une image fixe	140
4.13	Compromis entre la capacité, l'indétectabilité et la robustesse	141
5.1	Même image en noir et blanc, niveaux de gris et et couleurs	145
5.2	Cube RVB	145
5.3	Image couleur avec une palette de 16 couleurs	146
5.4	Stéganographie LSB pour une image non compressée	146
5.5	Décomposition en plans de bits d'une image codée en CGC	147
5.6	Décomposition en plans de bits d'une image codée en plans de bits usuels .	148
5.7	Algorithme d'insertion MBPIS	149
5.8	Algorithme d'extraction MBPIS	150
5.9	Images stéganographiées par MBPIS pour différents taux	150
5.10	Schéma de compression/décompression JPEG	151
5.11	Décomposition suivant les composantes RVB et YCbCr	152
5.12	Étape de sous-échantillonnage	153
5.13	Découpage en blocs de 8×8 valeurs	153
5.14	Exemple d'un bloc de coefficients DCT de la luminance	154
5.15	$c(Q)$ en fonction du facteur de qualité	155
5.16	Séquence Zig-Zag	156
5.17	Algorithme de construction du code de Huffman	158
5.18	Algorithme de Huffman pour $S = \text{« abracadabra »}$	158
5.19	Image <i>chinois.jpg</i>	160
5.20	Modifications de l'histogramme de <i>chinois.jpg</i> par Jsteg	160
5.21	Algorithme de correction statistique d' <i>Outguess</i>	162
5.22	Algorithme d'insertion d' <i>Outguess</i>	163
5.23	Algorithme d'extraction d' <i>Outguess</i>	164
5.24	Image <i>tahiti.jpg</i>	164
5.25	Modifications de l'hitogramme de <i>tahiti.jpg</i> par <i>Outguess</i>	165
5.26	Différence des histogrammes avant et après insertion	165
5.27	Images stéganographiées par <i>Outguess</i> pour différents taux stéganographiques	165
5.28	Codage des bits du message par les coefficients DCT	166
5.29	Algorithme d'insertion de $F5$	168
5.30	Algorithme d'extraction de $F5$	169
5.31	Image <i>soldat.jpg</i>	169

5.32	Modifications de l'histogramme de l'image <i>soldat.jpg</i> par <i>F5</i>	170
5.33	Différence de l'histogramme de l'image <i>soldat.jpg</i> après et avant insertion . .	170
5.34	Images stéganographiées par <i>F5</i> pour différents taux	171
5.35	Image <i>ecrire.jpg</i>	171
5.36	Modifications de l'histogramme de l'image <i>ecrire.jpg</i> par <i>JPHide</i>	171
5.37	Différence de l'histogramme de l'image <i>ecrire.jpg</i> après et avant insertion . .	172
5.38	Images stéganographiées par <i>JPHide</i> pour différents taux	172
6.1	Expérience en indistingabilité avec un attaquant A de type ATK contre Γ .	180
6.2	Expérience en indistingabilité avec un attaquant A de type ATK contre Σ .	181
6.3	Expérience en indistingabilité avec un attaquant A de type IND-SSA contre Σ	183
6.4	Expérience en indistingabilité avec un attaquant A de type IND-SSA(V, D_V) contre Σ	184
6.5	Expérience en indistingabilité avec un attaquant A de type IND-USA contre Σ_j	185
6.6	Expérience en indistingabilité avec un attaquant A de type IND-USA(V, D_V) contre Σ_j	186
6.7	Famille de courbes ROC d'un détecteur	189
6.8	Représentation des nuages pour $p = 3$ et $k = 2$	190
6.9	Pouvoir discriminant des axes de projection	191
6.10	Étape de décision pour un nouvel individu	193
7.1	Diagramme RS de image <i>acarrien.bmp</i> de taux stéganographique p avec $M = [0110]$	198
7.2	Diagramme MBPIS-RS de l'image <i>acarrien.bmp</i>	199
7.3	Diagrammes LSB-RS pour l'image <i>acarrien.bmp</i> stéganographiée à 5, 10, 15 et 20 %	200
7.4	Diagramme MBPIS-RS local de <i>acarrien.bmp</i>	204
7.5	Projection spatiale de $\mathcal{C}$ et $\mathcal{S}$ sur (R_M, R_{-M}, Q_R)	205
7.6	Courbes ROC du détecteur MBPIS, stéganalyse RS locale	206
7.7	Projection spatiale de $\mathcal{C}$ et $\mathcal{S}$ sur (S_M, S_{-M}, R_M)	207
7.8	Projection spatiale de $\mathcal{C}$ et $\mathcal{S}$ sur (S_M, S_{-M})	207
7.9	Courbes ROC du détecteur MBPIS, stéganalyse RS classique	208
7.10	Comparaison des courbes ROC pour des taux de 3%, 5%, 10% et 15% . . .	209
8.1	Étapes du JPEG et domaines associés	213
8.2	Lois de probabilité de P pour des supports et des stégo images générées par <i>JPHide</i>	214
8.3	Critère d'avalanche de l'étape de compression sans perte du JPEG	215
8.4	Média support et lois suivies par W avant et après insertion avec <i>Outguess</i> , $s = 8$	216
8.5	Distribution des coordonnées de V pour <i>Outguess</i> , $s = 32$	222
8.6	Distribution des coordonnées de V pour <i>F5</i> , $s = 32$	223
8.7	Distribution des coordonnées de V pour <i>JPHide</i> , $s = 32$	224
8.8	Meilleures courbes ROC pour D_1, D_2, D_3, D_4, D_5 et D_6	225
8.9	Distribution des coordonnées de V pour <i>Outguess</i> , $s = 32$	226
8.10	Distribution des coordonnées de V pour <i>F5</i> , $s = 32$	227
8.11	Distribution des coordonnées de V pour <i>JPHide</i> , $s = 32$	228
8.12	Meilleures courbes ROC pour D_1, D_2, D_3, D_4, D_5 et D_6	229

8.13	Image <i>coccinelle.jpg</i>	231
8.14	Distribution des coordonnées de V pour <i>Outguess</i> et $F5$	236
8.15	Distribution des coordonnées de V pour <i>JPHide</i>	237
8.16	Meilleures courbes ROC pour <i>Outguess</i> , $F5$ et <i>JPHide</i>	237
8.17	Distribution de Δ et Q pour <i>Outguess</i> et $F5$	238
8.18	Distribution de Δ et Q pour <i>JPHide</i>	239
8.19	Meilleures courbes ROC pour <i>Outguess</i> , $F5$ et <i>JPHide</i>	239

Liste des tableaux

1	Paramètres G-E pour un TEB donné	22
1.1	$(2, 1, m)$ et $(3, 1, m)$ -codes et convolutifs optimaux	39
1.2	$(4, 1, m)$, $(3, 2, m)$ et $(4, 3, m)$ -codes convolutifs optimaux	40
1.3	Codes poinçonnées de taux $\frac{2}{3}$	46
1.4	Codes poinçonnés de taux $\frac{3}{4}$	46
1.5	Codes poinçonnés de taux $\frac{4}{5}$	46
1.6	Codes poinçonnés de taux $\frac{5}{6}$	47
1.7	Codes poinçonnés de taux $\frac{6}{7}$	47
1.8	Codes poinçonnées de taux $\frac{7}{8}$	47
2.1	Reconstruction des codes en blocs des normes GSM, UMTS et CDMA 2000, $\gamma = 0.15$	83
2.2	Reconstruction de codes LDPC ($\gamma = 0.2$) et d'un code aléatoire ($\gamma = 0.15$) .	84
3.1	Reconstruction des codes convolutifs, TEB de 5.10^{-4} à 2.10^{-3}	110
3.2	Reconstruction des codes convolutifs 5.10^{-3} à 2.10^{-2}	111
4.1	Simulations dans un CBS $n = 512$ bits	128
4.2	Simulations dans un CBS $n = 1024$ bits	129
4.3	Simulations dans un CBS $n = 2048$ bits	129
4.4	Simulations dans un CBS $n = 256$ bits	129
5.1	Tables de référence pour la luminance (à gauche) et la chrominance (à droite)	155
5.2	Exemple d'un bloc de coefficients DCT quantifiés pour la luminance	156
5.3	Exemple de valeurs codées sur 3 bits	157
5.4	Exemple de codage de 4 coefficients DCT quantifiés	157
5.5	Code de Huffman pour $\mathcal{S}$ =« abracadabra »	159
6.1	Matrice de confusion d'un détecteur stéganographique	188
7.1	Taux de détection de MBPIS, stéganalyse MBPIS-RS locale, variables centrées	205
7.2	Taux de détection de MBPIS, stéganalyse LSB-RS classique, variables centrées	208
7.3	Comparaison stéganalyse MBPIS-RS locale et LSB-RS classique	210
8.1	Évolution de V en dissimulant 1 octet dans l'image <i>monde.jpg</i> , $s = 32$. . .	217
8.2	Distingueurs pour la stéganalyse universelle dans le DFC	219

8.3	Performances de D_1 et D_2	220
8.4	Performances de D_3 et D_4	221
8.5	Performances de D_5 et D_6	230
8.6	Statistiques pour <i>coccinelle.jpg</i> , un octet dissimulé par <i>Outguess</i> , <i>F5</i> et <i>JPHide</i>	232
8.7	Distingueurs pour la stéganalyse spécifique dans le DFC	232
8.8	Détection de <i>Outguess</i> et <i>F5</i>	233
8.9	Détection de <i>JPHide</i>	234

Table des matières

Remerciements	1
Contributions	5
Introduction	13
I Techniques de reconstructions de codes correcteurs d'erreurs	17
Introduction	19
1 Représentation algébrique des codes convolutifs et linéaires	25
1.1 L'approche algébrique	26
1.1.1 Codeurs et codes convolutifs	26
1.1.2 Fonctions génératrices	28
1.1.3 Définition d'un code convolutif	29
1.1.4 Codeurs convolutifs rékursifs	31
1.2 Propriétés de codeurs convolutifs	32
1.2.1 La notion de degré	32
1.2.2 Matrices génératrices	33
1.2.3 Matrices catastrophiques	36
1.2.4 Les codes optimaux	37
1.3 Les codes convolutifs poinçonnés	39
1.3.1 Définition des codes poinçonnés	39
1.3.2 Construction par regroupement	42
1.3.3 Approche algébrique du regroupement	43
1.3.4 Poinçonnage	45
1.3.5 Les « bons » codes poinçonnés	45
2 Reconstruction des codes linéaires	49
2.1 Étude algébrique	51
2.1.1 Formalisation du problème de reconstruction	51
2.1.2 Algorithme de Gauss randomisé	57
2.2 Algorithmes de reconstruction	60
2.2.1 Canal sans erreur	60

2.2.2	Canal avec erreur	63
2.3	Analyse des algorithmes	65
2.3.1	Canal sans erreur	67
2.3.2	Canal avec erreur	69
2.4	Résultats expérimentaux	77
2.4.1	Impact du poids d'une relation de parité	77
2.4.2	Impact du nombre de mots de code interceptés	78
2.4.3	Impact de l'erreur	79
2.4.4	Des codes bien réels	80
2.5	Conclusion et perspectives	82
3	Reconstruction des codes convolutifs	87
3.1	Étude algébrique	89
3.1.1	Retrouver les mineurs d'ordre k	90
3.1.2	Déterminer une matrice génératrice	93
3.2	Algorithmes de reconstruction	94
3.2.1	Canal sans erreur	95
3.2.2	Canal avec erreur	104
3.3	Analyse des algorithmes	105
3.3.1	Canal sans erreur	105
3.3.2	Canal avec erreur	108
3.4	Résultats expérimentaux	109
3.5	Conclusion et perspectives	112
4	Reconstruction des turbo-codes	115
4.1	Tries dynamiques	117
4.2	Algorithmes de reconstruction	120
4.2.1	Canal sans erreur	120
4.2.2	Canal avec erreur	123
4.3	Analyse des algorithmes	126
4.3.1	Canal sans erreur	126
4.3.2	Canal avec erreur	127
4.4	Résultats expérimentaux	128
4.5	Conclusion et perspectives	130
	Conclusion et perspectives	131
II	Étude de techniques de stéganalyse	133
	Introduction	135
5	Stéganographie adaptée aux images non compressées et au JPEG	143
5.1	Stéganographie dans le domaine spatial	144
5.1.1	Les images non compressées	144
5.1.2	Stéganographie dans des plans de bits multiples	145
5.2	Le format JPEG	151
5.2.1	Changement de l'espace des couleurs	151
5.2.2	Transformation DCT (Discrete Cosinus Transform)	152

5.2.3	Quantification	154
5.2.4	Codage RLE	156
5.2.5	Codage de Huffman	157
5.3	Stéganographie adaptée au format JPEG	159
5.3.1	Outguess	159
5.3.2	F5	161
5.3.3	JPHide and JPSeek	168
6	Introduction à la stéganalyse	173
6.1	Un problème de discrimination	175
6.2	Modèles d'attaquants	178
6.2.1	Modèle d'indistingabilité	178
6.2.2	Attaquant spécifique	182
6.2.3	Attaquant universel	185
6.2.4	Évaluation de la sécurité contre une attaque	187
6.3	Distingueur de Fisher	189
6.3.1	L'analyse discriminante	189
6.3.2	Discrimination particulière restreinte à deux classes	192
6.4	Conclusion	193
7	Stéganalyse dans le domaine spatial	195
7.1	Stéganalyse RS	196
7.2	Relative immunité de MBPIS	198
7.3	Stéganalyse RS locale	201
7.4	Résultats expérimentaux	203
7.4.1	Phase d'apprentissage, stéganalyse MBPIS-RS locale	204
7.4.2	Phase de challenge, stéganalyse MBPIS-RS locale	205
7.4.3	Phase d'apprentissage, stéganalyse LSB-RS classique	206
7.4.4	Phase de challenge, stéganalyse LSB-RS classique	207
7.4.5	Comparaison des détecteurs	208
7.5	Conclusion	209
8	Stéganalyse dans le domaine fréquentiel compressé	211
8.1	Le domaine fréquentiel compressé	212
8.2	Stéganalyse universelle dans le DFC	215
8.2.1	Schéma de stéganalyse universelle	215
8.2.2	Résultats expérimentaux	217
8.3	Stéganalyse spécifique dans le DFC	221
8.3.1	Schéma de stéganalyse spécifique	221
8.3.2	Résultats expérimentaux	232
8.4	Conclusion et perspectives	235
	Conclusion et perspectives	241
	Conclusion	245

Annexes	248
A Encadrement de $\prod_{i=0}^{n-1}(1 - 2^{i-M})$	249
B Quelques standards et normes	253
B.1 Téléphonie mobile terrestre	254
B.1.1 Le GSM	254
B.1.2 L'UMTS	255
B.1.3 IS95	256
B.1.4 CDMA 2000	256
B.2 Transmissions satellites	257
B.2.1 INMARSAT	257
B.2.2 INTELSAT	257
B.2.3 GLOBALSTAR	258
B.2.4 DVB-S	258
B.3 Réseaux sans fils	258
B.3.1 La norme 802.11a et HYPERLAN-2	258
B.4 Télévision Numérique Terrestre	258
C Hauteur moyenne d'un trie dynamique	261
C.1 Transformée de Mellin	261
C.2 Poissonnisation	263
C.3 Dépoissonnisation	263
C.4 Étude asymptotique	267
D Logiciels de stéganographie 2004-2006	269
Bibliographie	284
Glossaire	299
Index	300
Table des figures	304
Liste des tableaux	306
Table des matières	307

Résumé

Dans cette thèse, nous étudions les canaux de communication dans un contexte non coopératif sous l'angle des codes correcteurs d'erreurs, d'une part, et de la stéganographie, d'autre part. Nous prenons la place d'un observateur non légitime qui veut avoir accès à l'information échangée entre deux protagonistes. Nos travaux sur les algorithmes de reconstruction de codes correcteurs, nous ont amenés à proposer un formalisme commun pour l'étude des codes linéaires, des codes convolutifs et des turbo-codes. Nous analysons tout d'abord finement l'algorithme de Sicot-Houcke, puis l'employons ensuite comme brique de base pour concevoir une technique de reconstruction des codes convolutifs totalement automatique et de complexité meilleure que les techniques existantes. Enfin, nous utilisons ces résultats pour retrouver les paramètres des turbo-codeurs. Dans le cadre de l'analyse stéganographique, nous proposons tout d'abord deux nouveaux modèles de sécurité qui mettent en œuvre des jeux avec des attaquants réels. Nous adaptons ensuite l'analyse RS en un schéma de détection pour l'algorithme Multi Bit Plane Image Steganography pour le domaine spatial, proposé par Nguyen *et al.* à IWDW'06. Enfin, nous développons une approche nouvelle pour analyser les images JPEG. En étudiant les statistiques des coefficients DCT compressés, nous mettons en évidence des détecteurs possédant des performances élevées et indépendantes en pratique de la quantité d'information dissimulée. Nous illustrons ces résultats par un schéma de stéganalyse universelle d'une part, et un schéma de stéganalyse spécifique pour *Outguess*, *F5* et *JPHide* and *JPSeek*, d'autre part.

Mots-clefs : reconstruction d'algorithme, code correcteur d'erreurs, stéganalyse, domaine fréquentiel compressé.

Abstract

In this thesis, we study communication channels in a non cooperative context and more particularly, error correcting codes reconstruction on one hand, and steganalysis on the other hand. We play an unauthorized observer who wants to have access to the information exchanged between two legal users. Our researches on algorithms to reconstruct error correcting codes lead us to propose a generic approach for linear block codes, convolutional and turbo codes. We analyze the Sicot-Houcke algorithm and use it as a base to design an entirely automatic technique to reconstruct convolutional codes. Moreover, this technique has a better complexity than existing ones. Finally, we adapt these results to retrieve the parameters of turbo code encoders. In the context of steganographic analysis, we first propose two new models of security which play with a real-life adversary. Then, we adapt the RS analysis into a detection scheme for the Multi Bit Plane Image Steganography for the spatial domain. This algorithm was proposed by Nguyen *et al.* at IWDW'06. Finally, we develop a new approach to analyze JPEG files. We study statistics of compressed DCT coefficients and point out new detectors with high performances which are independent, in practice, of the amount of hidden information. We illustrate these results with an universal steganalysis scheme and a specific one to detect *Outguess*, *F5* and *JPHide* and *JPSeek*.

Keywords : algorithm reconstruction, error-correcting code, steganalysis, compressed frequency domain.