THÈSE de DOCTORAT de l'UNIVERSITÉ PARIS XI

Spécialité :

INFORMATIQUE

présentée par

Frédéric RAYNAL

pour obtenir le grade de DOCTEUR de l'UNIVERSITÉ PARIS XI

Sujet de la thèse :

ÉTUDES D'OUTILS POUR LA DISSIMULATION D'INFORMATION : APPROCHES FRACTALES, PROTOCOLES D'ÉVALUATION ET PROTOCOLES CRYPTOGRAPHIQUES

Soutenue le 1 Mars 2002, devant le jury composé de :

Evelyne	Lutton	Directrice
Sami	Harari	Rapporteurs
Jean	Louchet	
Jean-Paul	Allouche	Examinateurs
Caroline	Fontaine	
Françoise	Levy-dit-Vehel	
Fabien	Petitcolas	

Table des matières

Liste des tableaux

Remerciements

Ècrire ma thèse s'est avéré bien plus difficile que je m'y attendais. Ces quelques mots qui ouvrent ce volume sont en réalité les derniers que j'ai rédigés. Ils ne constituent pas les plus faciles et personne ne les a corrigés. Ainsi, le style *geek* et les fautes d'orthographe sont bien les miens.

Cette thèse a été réalisée sous la direction d'Evelyne Lutton. Je tiens à la remercier pour la pertinence de ses conseils scientifiques. Tous les membres, permanents ou de passage, du projet Fractales m'ont également bien aidé. Je n'en citerai qu'un seul, celui dont je suis le plus proche : mon collègue de bureau Benoît Leblanc. Nous constituions un superbe duo artistique ; l'un chantait pendant que l'autre dansait. Cela va me manquer. Bon, finalement, je compatis aussi avec Amine Boumaza qui a dû supporter ce terrible spectacle pendant quelques temps et dont la bonne humeur quotidienne est une véritable bénédiction.

Le projet Codes m'a également beaucoup apporté. Outre l'excellent café et l'ambiance chaleureuse, les conseils de Nicolas Sendrier et Pascale Charpin ont toujours été précieux pour mon travail.

Je remercie également Jean-Paul Allouche, Caroline Fontaine, Françoise Levy-dit-Vehel, Fabien Petitcolas, et plus particulièrement mes rapporteurs Sami Harari et Jean Louchet de m'avoir fait l'honneur de participer à mon jury de thèse.

Un des apports les plus enrichissants d'une thèse vient de la multitude des personnes rencontrées. Deux d'entre elles occupent une place à part : Caroline Fontaine et Fabien Petitcolas. L'amicale collaboration que nous avons développée pendant ma thèse m'a permis d'acquérir plus de recul sur les problèmes liés à la dissimulation d'information. Nos nombreuses discussions nous ont conduit à tisser des liens qui s'étendent au-delà d'une simple «coopération de travail».

Citer toutes les personnes auxquelles je tiens, et qui m'ont soutenu pendant ces dures années de labeur, transformerait ma thèse en annuaire. Pour faire court, je me limiterai donc à mes parents (Pierrounet et Gabi), mon neveux Martin (je te garde un exemplaire pour que tu le lises dès que tu sauras) et ses parents, Laurent et Aude, et surtout Sarah (bon courage pour la tienne). Les autres, vous saurez vous reconnaître dans cette liste de prénoms : Arnaud, Barbara, Luca, Nicolas, Anne, Victoire, Arnaud, Daniel, Fred, Brigite, Lucie, Vincent, David, Olivier, Hélène, Vincent, Carine, Dominique, Christophe, François, Pierre ...

Chapitre 1

Introduction Générale

1.1 Préambule

L'information a toujours constitué une denrée prisée. Comment l'acquérir ? Comment la protéger ? Comment en vérifier la provenance et l'intégrité ? Avec l'évolution des technologies et des connaissances, les réponses à chacune de ces questions ont évolué. De nouvelles défenses ont contré de nouvelles attaques, qui s'opposaient elles-mêmes à d'anciennes défenses ... La lutte entre l'épée et le bouclier est probablement loin d'être terminée.

1.1.1 Bref historique

Que signifie le terme *information* ? Il ne s'agit pas forcément de la notion rigoureuse définie par Shannon [Sha49b], mais plutôt de renseignements, de données qu'une personne ne souhaite pas divulguer : les plans d'une opération militaire, un numéro de carte bleue, des résultats d'analyses médicales ...

Deux grandes tendances visent à protéger l'information :

1. le chiffrement dont l'objectif est de rendre l'information incompréhensible à une personne ne possédant pas les connaissances adéquates : une personne surveillant le canal de communication par lequel transite le message sait qu'un échange a lieu, mais est ainsi incapable d'en interpréter le contenu ;
2. la stéganographie qui cherche plutôt à dissimuler la présence même d'informations pertinentes au sein de plusieurs autres sans réel intérêt pour le destinataire du message : le message secret est caché dans un support de manière à passer inaperçu lors de la communication.

Le premier procédé de chiffrement militaire est apparu en Grèce, dans la cité de Sparte au $5^{ème}$ siècle avant Jésus-Christ : le *scytale*. Il est constitué d'un bâton autour duquel l'auteur du message enroule un ruban de papyrus, de parchemin ou de cuir. Il écrit sur les bandes parallèlement à l'axe. Le ruban est alors déroulé puis transmis au destinataire, qui doit disposer d'un bâton de

même diamètre pour reconstituer le message. Cette méthode est la première par *transposition*, dont le principe est de mélanger les lettres.

Polybe, écrivain grec du $2^{ème}$ siècle avant Jésus-Christ, est à l'origine du premier procédé de chiffrement par *substitution*. Il propose l'utilisation d'un tableau à deux entrées afin de représenter chaque lettre par un nombre :

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	IJ	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

TABLEAU 1.1 : Carré de Polybe

Ces nombres devaient être transmis sur de longues distances à l'aide de torches, la ligne indiquant le nombre de torches à tenir dans la main droite, la colonne le nombre à tenir dans la gauche. Polybe ne relate néanmoins aucune situation dans laquelle son système soit employé.

Les premières utilisations confirmées de méthodes de substitution remontent aux Romains, en particulier à Jules César. Cette méthode substitue à une lettre donnée celle qui se trouve trois rangs plus loin dans l'alphabet.

FIGURE 1.1 : Chiffrement par substitution

Les premiers emplois de stéganographie sont relatés par Hérodote. Les cheveux d'un esclave de confiance sont tondus puis le message est tatoué sur son crâne chauve. Une fois que les cheveux ont repoussé, l'esclave est envoyé au destinataire qui lui rase de nouveau la tête afin de lire le message. Une autre approche consiste à graver le message sur une tablette de bois, ensuite recouverte de cire afin d'obtenir une tablette d'écriture normale. Plus tard, l'encre sympathique permet d'écrire de manière invisible sur du papier. Une exposition du papier de couverture devant une flamme révélait le message écrit avec cette encre. Un autre procédé consiste à marquer dans un document, comme un livre par exemple, certaines lettres qui formaient ainsi le message secret. La marque

peut être un simple trou d'épingle (méthode employée par les Allemands lors de la première guerre mondiale), ou bien un caractère de largeur différente ...

1.1.2 L'information aujourd'hui

Aujourd'hui, qui contrôle l'information détient énormément de pouvoir. Des exemples sont fournis par la guerre du Golfe ou les affrontements en Afghanistan. Les forces en présence distillent les renseignements de leur choix à leur population respective.

Par ailleurs, de grandes multinationales se partagent le contrôle de l'information :

- les constructeurs de matériels (Alcatel, Thalès, Erickson...) ;
- les exploitants (France Télécom, Deutsche Telekom, AT&T...) ;
- les fournisseurs de contenu numérique (Vivendi, AOL-Time-Warner...).

Bien souvent, ces groupes ne se contentent pas de développer une activité horizontale, mais tendent à contrôler plus globalement la chaîne de l'information, des infrastructures à la diffusion en passant par le contenu (Vivendi avec son portail Vizzavi par exemple).

Dans ces situations où l'information représente un tel enjeu stratégique et économique, il est devenu nécessaire de mettre en œuvre des outils adaptés aux nouvelles technologies : une protection renforcée de la vie privée et des droits d'auteur. La dissimulation d'information concerne donc aussi bien les individus que les entreprises ou les états.

L'explosion d'Internet ces dernières années effraie les États. Ils craignent une recrudescence d'activités illégales, protégées par une cryptographie trop forte. Ils limitent alors ces outils, et imposent le dépôt des clés de déchiffrement auprès d'une autorité de contrôle, mais de nombreuses associations s'opposent à cette démarche, craignant pour le respect de la vie privée. On évoque également de plus en plus les systèmes de surveillance, à l'échelle de pays entiers, capables d'enregistrer l'ensemble des communications numériques (e-mails, téléphones portables ...). Le plus connu est celui dont disposent les État-Unis, Echelon. De nombreux témoignages laissent supposer que son utilisation n'est pas uniquement vouée à la sécurité nationale, mais l'est également à des fins d'espionnage économique et industriel. Même si des échanges chiffrés ont lieu entre deux entités (sociétés, pays ...), l'existence même de ces communications fournit déjà de nombreux renseignements. En Nouvelle-Zélande, un projet de loi, actuellement en discussion, permettrait aux autorités d'intercepter les courriers électroniques, ainsi que leurs versions instantanées (Chat, IRC, Messengers), et d'accéder à n'importe quel ordinateur pour en contrôler le contenu. En Grande-Bretagne, cette loi existe déjà : la *Regulation of Investigatory Powers Act* (ou RIP Act) autorise les autorités à fouiller dans les boîtes-aux-lettres électroniques sans même recourir à un mandat. Toujours outre Manche, la loi anti-terroriste vise certains internautes : toute personne détenant des informations relatives à la *sécurité informatique* peut être suspectée d'aider les pirates de l'Internet, voire d'en être un. Ainsi, le simple fait de recevoir un message parlant de la

dernière vulnérabilité présente dans un serveur web ou une mise en garde contre un virus suffit. Pour protéger la vie privée, la solution est alors de dissimuler l'existence même d'une transaction aux regards des autorités.

De plus, un autre phénomène, induit par le «tout numérique», inquiète les industries du disque et du cinéma. Il est devenu extrêmement simple de reproduire parfaitement n'importe quel médium. Ces multinationales craignent de perdre leur contrôle, particulièrement avec l'essor des formats non propriétaires :

- `Ogg` pour les fichiers audios ;
- `png` (*Portable Network Graphics*) pour les images ;
- `openDivX` pour la vidéo.

Les éditeurs de logiciels ont longtemps recherché des solutions pour prévenir les copies illicites. Maintenant, la plupart d'entre eux ont renoncé. Pour contrer les effets du piratage individuel, ils développent un nouveau modèle qui combine support technique, fréquence élevée de mises à jour et poursuite des pirates «industriels» (contre-façon). De plus en plus de services en lignes (jeux, serveurs d'applications ...) permettent également de diminuer le coût de distribution. Ces évolutions sont dues à la nature numérique des logiciels. Dans le cas des média numériques (son, image et vidéo), les recherches se dirigent vers une solution technique : insérer une marque dans le médium afin d'identifier l'ayant-droit légitime.

1.2 Notions de cryptographie

Tout au long de ce document, nous faisons référence à des notions issues de la cryptographie. Dans cette partie, nous rappelons brièvement les domaines principaux de cette discipline, ainsi que leurs objectifs. Des descriptions plus complètes sont disponibles en annexe B page 189.

La première publication fondamentale en cryptographie a été l'article de C. Shannon «*Communication Theory of Secrecy Systems*» en 1949 [Sha49b]. Il y pose les bases d'un système de communications chiffrées décrit par la *théorie de l'information*.

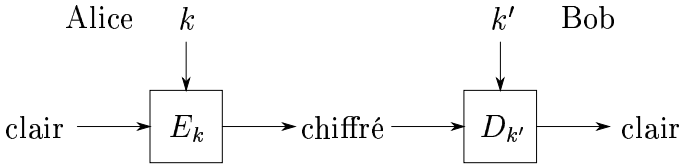


FIGURE 1.2 : Schéma de chiffrement

On distingue deux catégories d'algorithmes de chiffrement (cf. fig. 1.2) : celle à *clé secrète* et celle à *clé publique*. Les algorithmes de chiffrement à clé secrète (ou *symétriques*) nécessitent le partage d'un secret, la clé, pour chiffrer et déchiffrer un message. L'emploi d'un tel algorithme lors d'une communication demande alors l'échange préalable de cette clé entre les deux protagonistes. Un

paramètre essentiel pour la sécurité d'un tel système est la taille de l'espace des clés. En effet, il est toujours possible de mener une attaque dite *exhaustive* pour retrouver la clé. Cette attaque consiste simplement à essayer toutes les clés possibles du système.

En 1976, W. Diffie et M. Hellman ont introduit le concept révolutionnaire de chiffrement *à clé publique* dans [DH76]. Même si les auteurs ne donnent pas de réalisation pratique d'un tel système, ses propriétés sont clairement énoncées. Ils présentent également un protocole qui permet à deux entités de convenir d'une clé secrète uniquement à partir de données publiques : la clé secrète ne transite donc plus dans le canal de communication. Le premier système à clé publique est l'œuvre de R. L. Rivest, A. Shamir et M. Adelman [RSA77, RSA78] : le système RSA.

La cryptographie à clé publique (ou *asymétrique*) évite le partage d'un secret entre les deux interlocuteurs. Avec ce système de chiffrement, chaque utilisateur dispose d'un couple de clés. La clé publique est mise à la disposition de tous, dans un annuaire par exemple. L'utilisateur conserve soigneusement la *clé secrète* pour lui seul. Pour envoyer un message à Bob, Alice le chiffre à l'aide de la clé publique de Bob. Seul ce dernier est en mesure de déchiffrer le message reçu, grâce à sa clé secrète.

Plus récemment, pour faire face aux nouvelles menaces induites par le développement des réseaux et la numérisation massive des documents, la cryptographie a dû offrir de nouvelles fonctionnalités.

L'*intégrité des données* revient à vérifier que des données n'ont pas été modifiées depuis leur création. L'*authentification de source* (aussi appelée *authentification de message*) garantit qu'une entité est bien la source des données créées à un moment quelconque dans le passé. Ces domaines sont liés et l'un ne va pas sans l'autre.

Avec les protocoles d'*identification*, une entité est capable de prouver qu'elle est bien celle qu'elle prétend être. On distingue une version *faible*, comme les mots de passe, et une *forte* fondée sur les algorithmes asymétriques similaires à ceux utilisés en chiffrement, qui protège mieux le secret garant de l'identité.

Une autre application de plus en plus répandue concerne les schémas de *signature*. Ils sont composés d'une fonction de signature et d'une de vérification. La fonction de signature est paramétrée par une clé secrète propre au signataire ; elle associe au message en clair une signature. La fonction de vérification ne requiert la connaissance d'aucun secret. Elle permet, à partir du message en clair et de sa signature de vérifier l'authenticité de cette dernière. Un schéma de signature garantit donc :

- l'identité de la personne ;
- l'intégrité des données reçues, c'est-à-dire l'assurance que le message n'a pas été modifié durant sa transmission ;
- la non-répudiation du message, qui interdit au signataire de nier *a posteriori* en être l'auteur.

Nous avons vu jusqu'à présent des schémas où seules deux personnes intervenaient. Cependant, il est parfois souhaitable de *partager un secret*, comme une

clé, entre plusieurs entités. Le secret est alors découpé en *parts*, distribuées à chaque entité concernée. Chaque part ne donne aucune information sur le secret. Des structures d'accès définissent les groupes d'utilisateurs autorisés à accéder au secret lorsqu'ils rassemblent tous leur part.

1.3 Contenu de la thèse et plan du mémoire

Nous nous intéressons, dans le cadre de cette thèse, à la dissimulation d'information. Cette discipline se subdivise en plusieurs catégories en fonction des objectifs recherchés :

- la *stéganographie* cherche à dissimuler un *message secret* dans un médium chargé du transport afin qu'une personne qui surveille la communication ne le remarque pas ;
- le *tatouage* (ou *watermarking*) s'intéresse au problème de la protection des droits d'auteur : une *marque* personnelle est insérée dans un médium afin d'en identifier l'ayant-droit ;
- le *fingerprinting* a pour but de tracer la source d'une copie illégale d'un médium : une *empreinte*, propre à l'utilisateur, est déposée dans le médium lorsqu'il y accède ; si une copie illicite du médium est retrouvée, l'empreinte qu'elle contient révèle la provenance de la copie.

Pour le tatouage de nombreuses méthodes ont vu le jour, tant dans le domaine spatial que dans le domaine fréquentiel ou multi-résolutions. À l'inverse, les approches fondées sur des outils fractals ont été bien moins étudiées. Elles n'agissent dans aucun des domaines précédents mais se présentent plutôt comme un intermédiaire. C'est pourquoi il nous a semblé intéressant d'explorer cette direction, en particulier à l'aide des *systèmes de fonctions itérées* (*Iterated Functions System* - IFS) et de l'*analyse multifractale*.

Par ailleurs, la lecture des nombreux articles publiés sur le thème du tatouage montre un manque dans l'évaluation des algorithmes. En effet, chacun présente ses propres tests effectués sur les média de son choix. Il est ainsi extrêmement difficile de comparer des solutions car les caractéristiques mesurées diffèrent. L'élaboration d'un protocole expérimental commun apparaît donc comme nécessaire afin de faciliter la lecture des performances des schémas proposés.

Lors de la mise au point de cet outil, nous avons échafaudé des tests pour évaluer, entre autre, la robustesse du tatouage. Cependant, un algorithme de watermarking ne se limite pas à cet aspect et le cadre dans lequel il intervient influence également sa fiabilité. Le domaine de la cryptographie présente cette même caractéristique. Une fois un algorithme mis au point, il est inséré dans un protocole qui détaille les conditions nécessaires à son utilisation sécurisée. Nous avons donc voulu étudier cette démarche afin de la transposer à la dissimulation d'information. Cela nous a également permis d'inspecter les liens qui existent réellement entre ces deux disciplines.

Enfin, nous avons poursuivi notre analyse avec une application réelle : le protocole réseau SSH. Celui-ci utilise des algorithmes de cryptographie pour

sécuriser les échanges sur un réseau. La stéganographie sur ce support a été très peu abordée. La solution la plus couramment décrite consiste à dissimuler un message dans une image puis à la faire transiter *via* une page web par exemple. Le protocole HTTP qui régit la toile ne constitue cependant pas la seule solution pour entretenir une communication indétectable.

La diversité des thèmes abordés a été rendue possible grâce à une collaboration avec des spécialistes de chaque domaine. Elle illustre parfaitement la richesse, et la difficulté qui en découle, de la problématique de la dissimulation d'information. Ce champ d'investigation révèle de multiples facettes : l'étude de chacune est nécessaire pour parvenir à des solutions satisfaisantes.

Dans ce mémoire, nous commençons par présenter la terminologie et les objectifs de la dissimulation d'information. La stéganographie, le tatouage et le fingerprinting sont abordés en détail au chapitre 2 page 21. La suite de cette étude est divisée en plusieurs parties.

La première s'intéresse à deux domaines issus de la théorie fractale : les IFS et l'analyse multifractale. Tout d'abord, nous présentons un travail réalisé à partir de l'étude du schéma de tatouage imaginé par Puate et Jordan dans [PJ96] qui repose sur la compression fractale. L'idée en est de contraindre la résolution du problème inverse pour un IFS. Après une analyse de cette solution, nous développons nos travaux sur les IFS. Nous introduisons une nouvelle classe d'IFS, dits *polaires*, dont la contractance est plus facilement contrôlable que celles des IFS mixtes. Nous développons ensuite un algorithme évolutionnaire particulièrement bien adapté à la manipulation d'IFS. Nous le testons sur deux problèmes différents de génération d'IFS sous contraintes. Les résultats obtenus améliorent significativement ceux donnés sur ces mêmes problèmes par des algorithmes évolutionnaires classiques. Néanmoins, les performances ne sont pas encore suffisantes pour que cette approche soit utilisable. Enfin, nous abordons une nouvelle direction dans l'utilisation de l'analyse multifractale et des spectres mutuels en étudiant les paramètres caractéristiques de ceux-ci pouvant alors servir de base dans le cadre d'un schéma de tatouage par substitution.

Nous abordons ensuite des aspects connexes à la dissimulation d'information. Tout d'abord, nous évoquons la nécessité de la mise en œuvre d'une méthode d'évaluation commune afin de permettre une comparaison et une analyse pertinente des méthodes proposées. Nous détaillons notre outil d'évaluation, `Stirmark Benchmark`, résultat d'une collaboration Européenne entre l'INRIA, le LIFL et Microsoft Research. Ce logiciel a été conçu indépendamment de la nature du médium. Nous avons commencé le développement pour les images. Par la suite, notre collaboration s'est étendue à l'Université de Darmstadt afin d'ajouter le support audio. Nous analysons les liens supposés entre la cryptographie et la dissimulation d'information. Au travers de travaux antérieurs ainsi que de nouvelles directions de prospective, nous montrons le bien-fondé de cette relation. Enfin, dans un dernier chapitre, nous nous intéressons à la problématique d'un canal caché dans un réseau. Nous démontrons l'existence d'un tel canal dans le protocole SSH, ainsi que ses utilisations potentielles.

Notations et présentation des acteurs

Les paramètres intervenant tant en cryptographie qu'en dissimulation d'information sont très semblables. Afin de faciliter la lecture, nous conservons au fil de ce mémoire des notations identiques entre ces deux domaines, mais changeons simplement la police utilisée. Celle employée pour la cryptographie est celle définie par défaut en LaTeX. En revanche, toutes les notations relatives à la dissimulation d'information sont en sans serif.

Cryptographie	**Dissimulation d'information**
message : m	médium : m
chiffrement : $E_k(m)$	insertion[1] : $\mathsf{E_k(m, d)}$
déchiffrement : $D_{k'}(\tilde{m})$	détection : $\mathsf{D_{k'}(\tilde{m}, d)}$

TABLEAU 1.2 : Notations

Par ailleurs, la cryptographie est peuplée de personnages qui servent à présenter les protocoles :

- Alice et Bob sont les utilisateurs normaux du protocole. En général, Alice utilise la partie publique et Bob la partie privée ;
- Ève est une attaquante *passive* qui se contente d'écouter sur le canal de communication employé par Alice et Bob ;
- Charlie est un attaquant *actif* qui non content d'écouter la communication, cherche aussi à intervenir directement sur le contenu du canal.

Nous essaierons de respecter les rôles dédiés à ces personnages tout au long de cette étude, aussi bien pour la cryptographie que pour les autres chapitres.

Enfin, l'*oracle* est une entité, dont on ne sais rien, à qui on soumet une question et qui en fournit la réponse. Il peut s'agir, par exemple, d'une machine de chiffrement à clé secrète à qui on donne un message en clair et qui retourne le message chiffré correspondant.

[1] La lettre E est conservée de l'Anglais *embedding*

Chapitre 2

Dissimulation d'information : terminologie et problématiques

Le terme *dissimulation d'information* est très général ; il désigne simplement le fait de cacher une information dans un support. Il s'agit d'une libre adaptation de l'expression Anglaise *information hiding*[1] couramment utilisée dans la littérature. Cependant, selon les objectifs, et les contraintes qui en découlent, on distingue différentes variantes.

2.1 Problématiques en dissimulation d'information

2.1.1 Terminologie

Dans la suite de ce mémoire, nous conserverons toujours la terminologie définie ici, afin de distinguer les différents éléments qui interviennent en dissimulation d'information.

Tout d'abord, le médium vierge dans lequel des informations sont cachées est le *médium de couverture*, ou plus prosaïquement le *médium*. Au contraire, une fois que les informations sont insérées, nous utilisons alors l'expression *stégo-médium*. D'une manière générale, nous appelons *données* l'information dissimulée dans le stégo-médium.

Le processus complet de dissimulation d'information repose sur deux opérations :

1. la *dissimulation*, qui consiste à insérer l'information dans le médium ;
2. l'*extraction*, qui récupère cette information. Le mot *détection* est également utilisé lorsqu'il s'agit de vérifier la présence d'une information[2] dans le stégo-médium, sans pour autant vouloir l'extraire.

[1] Ce terme est apparu au moment du *first international workshop on information hiding"* en 1996.

[2] Il peut s'agir d'un signal, d'une caractéristique particulière du médium ...

2.1.2 Objectifs

La *stéganographie*[3] cherche à cacher un *message secret*[4] dans un médium de sorte que personne ne puisse distinguer un médium vierge d'un stégo-médium. La nature de l'information dissimulée ne revêt pas d'importance : il peut tout aussi bien s'agir d'un texte en clair que de sa version chiffrée. Ce message n'a *a priori* aucun lien avec le stégo-médium qui le transporte.

Lorsqu'un attaquant tente uniquement de détecter si un message transite dans un médium sur le canal de communication, on dit de lui qu'il est *passif*. La plupart des solutions de stéganographie ne considèrent que ce type d'attaquant, au contraire des deux domaines suivants où il est *actif* : l'attaquant sait alors que le stégo-médium contient une information et il tente de la modifier ou de la retirer.

Les deux applications suivantes sont parfois considérées comme descendant de la stéganographie. Cependant, la contrainte d'imperceptibilité y est bien moins forte. C'est pourquoi nous préférons utiliser dans ce mémoire la terminologie courante qui consiste à placer ces trois domaines au même niveau, comme des cas particuliers de dissimulation d'information.

Le *tatouage* cherche à répondre au problème de la protection des droits d'auteur. Il tente de fournir une solution pour prouver qu'une entité est bien le véritable propriétaire d'un médium. Il s'agit bien de dissimulation d'information puisque, pour y parvenir, on insère un *tatouage* (ou *marque*, ou *filigrane*) dans le médium spécifique au propriétaire. Comme celui-ci souhaite protéger son médium et non une version trop déformée, l'insertion doit minimiser les modifications subies par le médium afin d'être imperceptible. Ensuite, chaque copie du stégo-médium contient la même marque, celle du propriétaire légal. Ici, la dissimulation ne signifie pas la même chose qu'en stéganographie : un attaquant sait qu'un tatouage est présent dans le stégo-médium, mais cette connaissance ne doit cependant pas lui permettre de le retirer.

Enfin, le *fingerprinting* cherche à permettre la détection des copies illégales d'un stégo-médium. Chaque utilisateur authentifié reçoit sa propre copie du médium qui contient une *empreinte* l'identifiant. Ainsi, lorsqu'une copie illégale est découverte, la lecture de l'empreinte indique la source de la fuite. À la différence du tatouage où l'origine du médium importe, le fingerprinting se préoccupe plutôt de l'utilisateur final. Chaque copie du médium contient une information différente, relative à son utilisateur, rendant alors chaque stégo-médium différent.

Chacun de ces trois domaines est très spécifique, et peut encore être affiné. Néanmoins, nous nous limitons à ces trois grandes classes d'applications.

[3] Du grec *steganos*, caché ou secret, et *graphy*, écriture ou dessin.
[4] Nous emploierons le terme *message*, plus bref, par la suite.

2.1.3 Schéma général

Malgré leurs objectifs distincts, ces trois variantes n'en requièrent pas moins des paramètres communs :

- chaque approche nécessite des données, que ce soit un message, un tatouage ou une empreinte ;
- ces données sont dissimulées dans un support, le médium, qui possède plus ou moins d'importance selon le schéma : aucune pour la stéganographie, capitale pour les deux autres ;
- il est indispensable de pouvoir distinguer des personnes différentes, utilisant des données identiques dans un même médium : chacune doit donc posséder sa propre *stégo-clé* (ou plus simplement *clé*) afin que l'insertion de ces données identiques permettent quand même de différencier les protagonistes.

	Données	**Médium**	**Clé**
Stéganographie	le message à transmettre	sans importance	utilisée pour insérer/récupérer le message
Tatouage	une marque dépendant du médium et/ou du propriétaire	le médium dont on veut protéger les droits	utilisée pour insérer/détecter la marque dans le stégo-médium et, éventuellement, chiffrer le message
Fingerprinting	une empreinte dépendant du médium **et** de son utilisateur	le médium dont on souhaite prévenir la diffusion de copie illégale	utilisée pour insérer/détecter l'empreinte dans le stégo-médium

TABLEAU 2.1: Utilisation des données, média et clés en dissimulation d'information

Le tableau 2.1 montre que le rôle de la clé ne change pas dans ces applications : une clé est nécessaire pour insérer les données dans le médium, et les extraire (ou en détecter la présence) dans le stégo-médium.

Dans le cas du tatouage et du fingerprinting, les données sont construites, entre autre, à l'aide d'un générateur aléatoire. Celui-ci est initialisé avec une *graine*, parfois à tort appelée clé, mais qui n'a *a priori* aucun lien avec celle évoquée précédemment, même si elles se confondent dans certaines méthodes.

La figure 2.1 présente un schéma général pour l'insertion des données dans le médium.

Le comportement de la procédure d'extraction/détection dépend des besoins de l'application. Signalons également que la réponse fournie change : alors que la récupération du message est le but même de la stéganographie, une mesure de confiance sur la présence ou non de données dans le stégo-médium peut suffire en tatouage ou en fingerprinting.

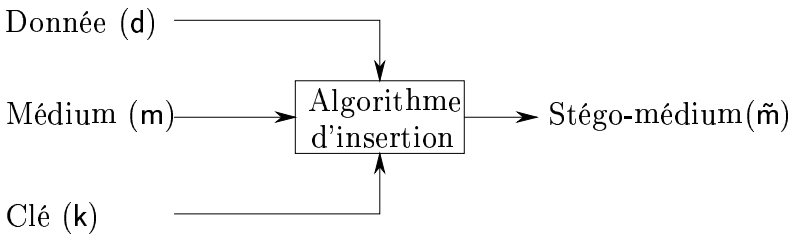


FIGURE 2.1 : Insertion de données dans un médium

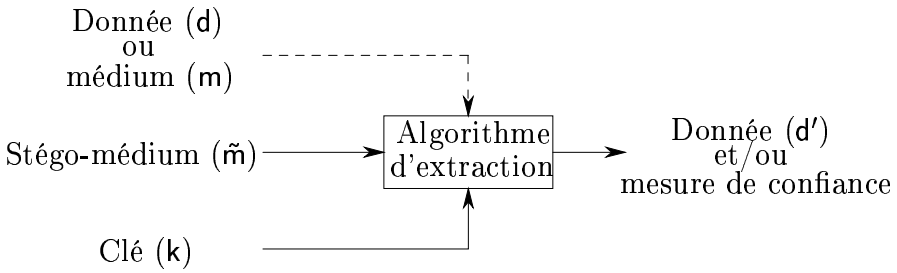


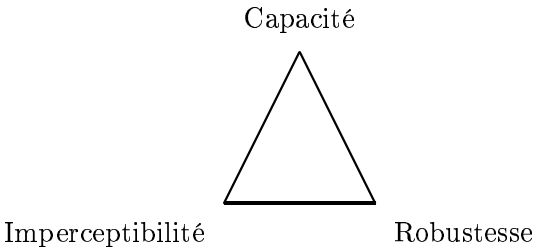
FIGURE 2.2 : Fonction d'extraction

2.1.4 Conditions requises

Nous avons vu que les objectifs de la dissimulation d'information peuvent changer de manière subtile. Classiquement, les applications sont triées en fonction de trois critères :

1. l'*imperceptibilité* : les données ne doivent pas être «perceptibles» dans le stégo-médium. Pour le tatouage ou le fingerprinting, l'objectif est de ne pas détériorer le stégo-médium protégé. Cependant, la contrainte est plus forte en stéganographie où il s'agit plutôt d'une indétectabilité statistique afin qu'une personne surveillant le canal ne remarque pas la présence du message ;
2. la *capacité* est la quantité de bits significatifs dissimulés dans le stégo-médium par unité d'accès (par exemple, le nombre de bits par seconde en musique) ;
3. la *robustesse* correspond à l'aptitude de préservation des données cachées face aux modifications du stégo-médium.

En stéganographie, une propriété essentielle est l'indétectabilité statistique puisqu'une personne surveillant le canal de communication ne doit pas pouvoir différencier un médium d'un stégo-médium (voir [Cac98] pour une explication en termes de théorie de l'information). De plus, comme le message constitue l'information principale, la capacité doit aussi être assez élevée. Quant à la robustesse, elle constitue une défense contre les modifications subies par le stégo-médium. Néanmoins, la meilleure défense reste l'incapacité de l'adversaire à détecter le



message. Ainsi, la plupart du temps, le canal ne modifie pas le stégo-médium et les besoins en robustesse sont minimes. En revanche, des mesures doivent être prises lorsque que l'adversaire est actif, soit en terme de robustesse, soit pour contrôler l'intégrité du message afin de détecter un éventuel changement dans celui-ci.

En tatouage, les contraintes diffèrent largement. Tous les utilisateurs savent, ou soupçonnent très fortement, qu'une marque est dissimulée dans le stégo-médium, et il n'est donc nul besoin de chercher à en détecter la présence. Le stégo-médium doit toutefois rester aussi proche[5] que possible de l'original afin de ne pas être dénaturé. La capacité dépend étroitement de l'application. Si le tatouage est suffisamment discriminatoire, un bit d'information suffit à répondre à la question : cette marque est-elle présente dans ce stégo-médium ? Même lorsque les données sont extraites et une mesure de confiance calculée, l'utilisation d'un seuil pour valider ou non la présence de la marque ne fournit toujours qu'un bit d'information. Au contraire, lorsque les données extraites servent ensuite à diriger une action, on considère alors que le tatouage transporte de l'information. C'est, par exemple, ce que font les *images intelligentes* (*smart images* - [Ala00]) : une telle image est placée devant une entrée (caméra, appareil photo numérique, scanner...) puis est traitée par un logiciel qui en extrait les données avant d'entreprendre les actions associées. Le site de Digimarc[6] donne des exemples pratiques comme montrer la photo d'un paysage pour se voir proposer ensuite toutes les offres promotionnelles de voyage correspondant à la région décrite sur l'image. Une autre variante du tatouage, qualifié alors de *faible*, ne demande que peu de robustesse afin de détecter rapidement qu'un médium, ou une partie de celui-ci, a été modifié.

Enfin, les besoins du fingerprinting sont à peu près identiques à ceux du tatouage pour l'imperceptibilité et la robustesse (pour ce dernier critère, les raisons diffèrent : on ne souhaite pas voir un utilisateur distribuer sa propre copie ... avec l'empreinte de quelqu'un d'autre insérée). En revanche, la capacité est importante car un médium doit contenir une empreinte spécifique à

[5] Au sens d'une mesure de similitude sur l'espace des média.
[6] http://www.digimarc.com/imaging/smartimages.htm

un utilisateur. Dans ces conditions, il n'est pas réaliste de se contenter, comme en tatouage, d'une réponse binaire sur la présence d'une empreinte dans un stégo-médium car il faudrait alors tester toutes les empreintes pour un stégo-médium. Ainsi, tout comme en stéganographie, l'extraction de l'empreinte est indispensable.

2.2 Stéganographie

Simmons [Sim84] présente un modèle de communication indétectable dans son *problème des prisonniers*. Alice et Bob sont arrêtés et jetés en prison dans des cellules différentes. Ils ne souhaitent pas y rester et veulent échafauder un plan d'évasion. Malheureusement, tous leurs échanges sont soumis à l'attention d'une gardienne nommée Wendy. Elle ne les laisse pas communiquer dès lors que le message est chiffré. De même, si elle remarque que les messages qu'elle laisse transiter contiennent une information suspecte, elle interdit immédiatement toute forme de communication.

Alice et Bob cherchent donc à échanger des messages sans éveiller les soupçons de Wendy : ils utilisent un *canal subliminal*[7]. Une solution pratique consiste à dissimuler le message important dans un vecteur qui semble quelconque à toute autre personne.

Dans cette partie, nous commençons par donner quelques exemples d'applications existantes avant de détailler les trois classes de schémas de stéganographie définies selon leur mode d'utilisation.

2.2.1 Applications

Il est possible de présenter les techniques de stéganographie selon différentes classifications. Par exemple, N. Johnson et S. C. Katzenbeisser dans [KP99a] le font d'après les modifications induites sur le support. Ils distinguent alors six catégories :

- un *système par substitutions* remplace les parties redondantes du support par le message ;
- les *techniques par transformations* dissimulent l'information dans une transformée du support, comme par exemple, le domaine des ondelettes ;
- les *techniques par étalement de spectre* reposent sur le schéma du même nom ;
- les *méthodes statistiques* modifient plusieurs statistiques du support (fréquences des lettres, distribution des pixels...) pour cacher le message, et le récupèrent en testant ces hypothèses ;
- les *techniques par distorsions* altèrent le support, la différence avec le support initial constituant alors le message ;
- les *méthodes par génération de support* construisent un support autour du message pour le dissimuler.

[7] On dit aussi *canal caché* ou, plus rarement, *canal stéganographique*.

Nous avons opté pour une présentation reposant sur le choix du support de dissimulation. La stéganographie est une discipline ancienne et les schémas proposés actuellement tentent de reproduire dans l'univers digital ce qui se faisait autrefois avec de l'encre et du papier. Pour cette raison, les algorithmes actuels utilisent majoritairement du texte et des média comme support. Nous détaillons aussi l'emploi de quelques supports plus modernes. Nous verrons au chapitre 6 que l'utilisation d'un canal caché est également envisageable avec un protocole réseau.

2.2.1.1 Stéganographie à l'aide d'un texte

De nombreuses solutions ont été proposées pour dissimuler le message directement dans le texte. Les trois premières méthodes sont facilement détectables pour un observateur attentif.

Ajout d'espaces à la fin des phrases

On définit le codage suivant :
- bit 0 : aucun espace en fin de ligne ;
- bit 1 : un espace en fin de ligne.

Ainsi, la première strophe de la *Chanson d'Automne* de Paul Verlaine contient ici la chaîne binaire `110110` (les espaces de fin de lignes ont été remplacés par le caractère '_') :

Les sanglots longs_
Des violons_
De l'Automne
Blessent mon cœur_
D'une langueur_
Monotone

Espacement entre les mots

Brassil *et al.* [BLM99] proposent de jouer sur la mise en forme d'un texte pour dissimuler des informations. Ils changent l'espacement entre les lettres, voire des mots, dans deux versions d'un même texte. Le message secret apparaît ensuite lorsque les deux textes sont superposés.

Ainsi, l'œil ne distingue-t-il aucune différence entre cette phrase

A l'envers, cette figure nuit à la compréhension.
A l'endroit, son exposition semble trop convenue.

et celle-ci

A l'envers, cette figure nuit à la compréhension.
A l'endroit, son exposition semble trop convenue.

La superposition des deux met en relief une phrase :

A l'envers, **cette** figure **nuit à** la compréhension.
A **l'endroit**, son exposition semble trop **convenu**e.

Méthode des synonymes

Avec cette méthode, une table de synonymes est construite :

bit 0	bit 1
éternel	impérissable
pluie	ondée
charité	bienfaisance
complet	exhaustif
limite	borne

Le message est ensuite écrit en utilisant les mots clés préalablement choisis. Cette technique trompe facilement une machine, mais la sémantique du texte ainsi généré risque de ne pas abuser un humain.

D'autres propositions ont été faites pour dissimuler le message dans les balises du langage. Par exemple, le HTML se prête parfaitement à ce type de manipulations : le texte qui apparaît sur l'écran est mis en forme à l'aide de balises, mais toutes ne sont pas intégralement interprétées. Ainsi, le code de déchiffrement des DVDs est présent dans la page http://decss.zoy.org/ de S. Hocevar, dissimulé sous forme de commentaires[8]. Il est alors possible de le visualiser en regardant les sources de la page :

```
<!--
/*****************************************************************************
 * css_unscramble.c : unscrambling function
 *****************************************************************************
 * Copyright (C) 1999 Derek Fawcus
 * ...
 *****************************************************************************/

...

void CSSdescramble(unsigned char *sec,unsigned char *key) {
  unsigned int t1,t2,t3,t4,t5,t6;
  unsigned char *end=sec+0x800;
  t1=key[0]^sec[0x54]|0x100;
  t2=key[1]^sec[0x55];
  t3=(*((unsigned int *)(key+2)))^(*((unsigned int *)(sec+0x56)));
  t4=t3&7;
  t3=t3*2+8-t4;
  sec+=0x80;
  t5=0;
  while(sec!=end) {
        t4=CSSt2[t2]^CSSt3[t1];
        t2=t1>>1;
```

[8]Cette page présente 42 manières originales de distribuer DeCSS. Signalons également les travaux de P. Carmody (http://asdf.org/~fatphil/maths/illegal1.html) qui a converti DeCSS en un nombre premier.

```
        t1=((t1&1)<<8)^t4;
        t4=CSSt5[t4];
        t6=(((((((t3>>3)^t3)>>1)^t3)>>8)^t3)>>5)&0xff;
        t3=(t3<<8)|t6;
        t6=CSSt4[t6];
        t5+=t6+t4;
        *sec++=CSSt1[*sec]^(t5&0xff);
        t5>>=8;
  }
}
...

/*************************************************************************
 * This little piece of software was brought to you by an HTTP server.
 *************************************************************************/
-->
```

Le principal problème de ces solutions réside dans leur manque total de robustesse. Un adversaire actif peut simplement choisir de reformater le texte et détruire ainsi les informations dissimulées dans l'agencement. Par exemple, il n'a qu'à changer la justification et la taille de l'interligne.

De plus, un texte existe indépendamment du format utilisé pour le sauvegarder (HTML, LaTeX, Postscript, RTF, ...) et la conversion de l'un à l'autre détruit souvent, outre la mise en page, les instructions de formatage puisque celles-ci dépendent justement de la représentation utilisée.

Utilisation de grammaires

Wayner propose dans [Way92, Way95] une solution plus robuste fondée sur l'emploi de grammaires algébriques [9] pour la génération de texte.

Les règles de production sont déterminées puis pondérées par une probabilité. On construit ainsi des arbres identiques à ceux utilisés dans l'algorithme de compression de Huffman. Les règles données respectivement dans la figure 2.3 pour le sujet S, 2.4 pour le verbe V et 2.5 pour l'attribut A permettent de construire des phrases à partir de l'axiome S en fonction de la chaîne binaire à dissimuler.

Par exemple, si Alice souhaite envoyer à Bob la chaîne `11011`, elle construit une phrase avec sujet (S), verbe (V) et attribut (A) à partir des arbres prédéfinis : le `11` donne «La terre» dans l'arbre S du sujet, le `0` le verbe «devient» puis finalement «blanche.» pour l'attribut. Elle envoie donc la phrase «La terre

[9] Une *grammaire algébrique* (ou *libre de contexte*) est un système formel qui décrit un langage afin de construire du texte. Ce système contient un ensemble de *règles de production*, chacune indiquant le remplacement à effectuer pour un symbole du langage. Le texte est construit à partir d'un *axiome* initial, constitué soit d'un symbole, soit d'une suite de symboles. Le texte est construit en appliquant les règles de dérivations sur les symboles.

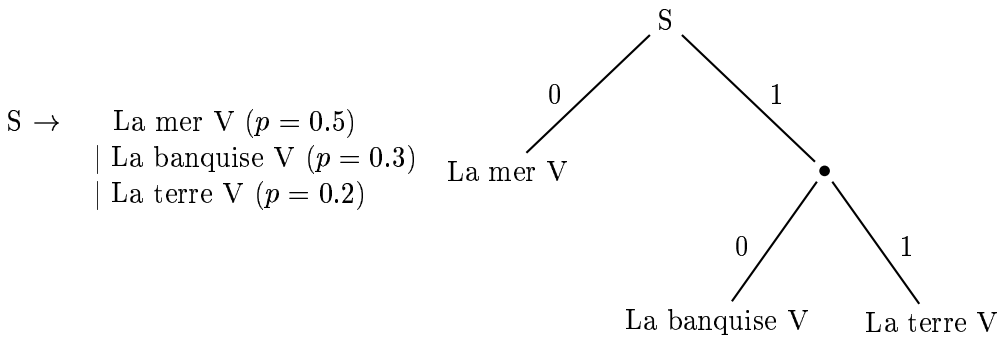


FIGURE 2.3 : Règle de dérivation pour le sujet S et arbre de Huffman associé

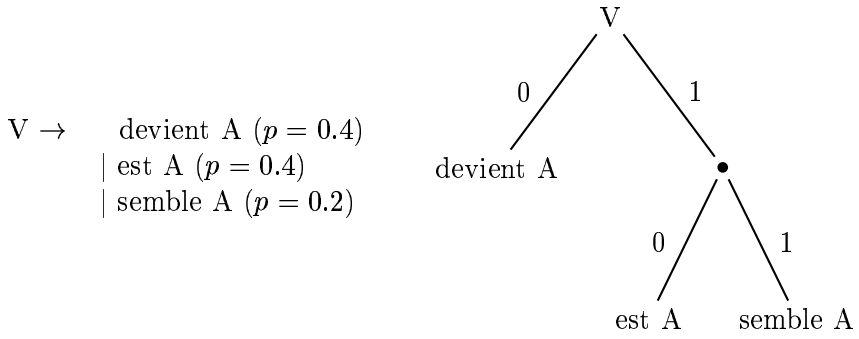


FIGURE 2.4 : Règle de dérivation pour le verbe V et arbre de Huffman associé

devient blanche.» à Bob. Il reconstruit la chaîne binaire à l'aide des arbres. Si la grammaire n'est pas ambiguë, cette décomposition est alors unique.

2.2.1.2 Stéganographie à l'aide d'un médium

De nombreuses solutions ont été proposées pour des fichiers audios, des images et des vidéos. Deux conditions sont recommandées pour obtenir un schéma sécurisé lorsqu'un médium est utilisé [ZFK+98] :

1. la clé secrète servant à la dissimulation n'est pas connue de l'adversaire ;

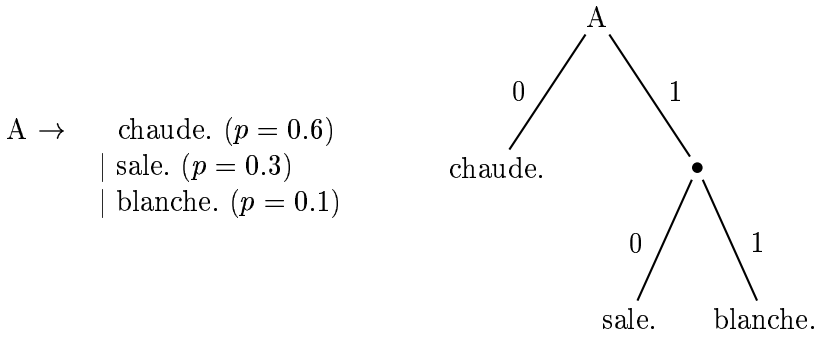


FIGURE 2.5: Règle de dérivation pour l'attribut A et arbre de Huffman associé

2. le médium initial n'est pas disponible pour l'adversaire, ce qui est facilement réalisable à l'aide d'un appareil photo numérique ou d'un scanner.

Ainsi, pour les images, le logiciel `outguess`[10] de N. Provos [Pro01] propose deux méthodes pour dissimuler des données dans une image.

La première utilise un générateur aléatoire. Selon la graine utilisée, différents bits sont sélectionnés pour être modifiés afin de cacher le message. La solution qui modifie le moins l'image initiale est conservée. Des codes correcteurs constituent la base de la seconde méthode. Ces deux solutions sont utilisables conjointement.

L'auteur distingue deux étapes dans l'insertion :

1. l'identification des bits redondants[11] car ils sont potentiellement modifiables, sans endommager l'image initiale ;
2. la sélection des bits dans lesquels l'information sera placée.

Cette séparation offre l'avantage de pouvoir travailler indépendamment d'un type de médium donné. De plus, il est ainsi facile de modifier l'une ou l'autre des deux étapes afin de mesurer la pertinence de ce changement. En revanche, l'inconvénient vient de la perte d'information entre les deux étapes. En effet, le résultat de l'identification est ensuite transmis à la sélection : celle-ci s'effectue donc sans connaissance de la provenance des bits dans l'image et ne tient donc pas compte du fait qu'une région est peut-être moins sensible qu'une autre à des altérations. Les bits sélectionnés sont modifiés pour correspondre à ceux du message.

Pour pallier ceci, la sélection des bits à modifier ne tend pas uniquement à minimiser le nombre de ces bits, mais tient également compte de l'accroissement de la probabilité de détection que son changement induit.

2.2.1.3 Stéganographie à l'aide d'autres supports

Un ordinateur, pour fonctionner, s'appuie sur de nombreuses couches situées entre le matériel (*hardware*) et le logiciel (*software*) en passant par le système d'exploitation. Nous présentons ici deux ressources utilisées pour faire de la stéganographie.

Fichier binaire exécutable

Une fois compilées, les sources d'un programme sont transformées en un fichier d'instructions compréhensibles par le système d'exploitation. Ce fichier, appelé *binaire*, possède sa propre structure composée de différentes sections, chacune ayant sa propre utilité. Lorsque le programme doit être exécuté, le système d'exploitation lit toutes les informations dont il a besoin dans ces différentes sections.

[10] http://www.outguess.org

[11] L'auteur appelle *bit redondant* un bit dont il est possible de changer la valeur sans modifier la qualité de l'image.

En particulier, une de ces sections comporte les instructions en langage machine. Par exemple, le tableau 2.2 présente un petit programme et le contenu de la section `.text` qui renferme, sous Linux, les instructions en langage machine. On constate sur cet exemple très simple que de nombreux octets ne servent à rien : les `NOP` signifient *no operation*. Cet espace est donc «perdu» puisque les instructions n'engendrent aucune opération. Il est alors possible de remplacer ces octets par ceux de notre choix, sous réserve qu'à aucun moment le registre d'instructions[12] ne viennent dans cette zone (on parle alors de *code mort*). En effet, les octets placés ont peu de chance de correspondre à une instruction valide et provoqueraient alors un échec du programme.

```
/* hello.c */
main() {
  printf("Bonjour\n");
}
```

```
[raynal]$ gcc -o hello hello.c
[raynal]$ gdb -q hello
(gdb) disass main
Dump of assembler code for function main:
0x80483c8 <main>:       push   %ebp
0x80483c9 <main+1>:     mov    %esp,%ebp
0x80483cb <main+3>:     push   $0x8048430
0x80483d0 <main+8>:     call   0x8048308 <printf>
0x80483d5 <main+13>:    add    $0x4,%esp
0x80483d8 <main+16>:    leave
0x80483d9 <main+17>:    ret
0x80483da <main+18>:    nop
0x80483db <main+19>:    nop
0x80483dc <main+20>:    nop
0x80483dd <main+21>:    nop
0x80483de <main+22>:    nop
0x80483df <main+23>:    nop
End of assembler dump.
```

TABLEAU 2.2: Programme C et sa version Assembleur dont les instructions sont écrites dans la section `.text`

Une autre solution est proposée par P. Collberg dans [CTL98]. Elle consiste à transformer un programme P en un autre P' qui adopte le même comportement. Le principe est d'obscurcir les sources du programme initial en modifiant les instructions. Par exemple, l'insertion de nouvelles instructions est réalisée en ajoutant un branchement conditionnel dont les deux parties sont constituées de code équivalent. Le programme ainsi modifié se comporte de manière identique à l'original. Cependant, la suite d'opérations réalisées pour l'obscurcissement correspond en fait à un message pour qui connaît ces opérations.

Système de fichiers

[12] Le registre `%eip` pointe dans la section `.text` sur la prochaine instruction à exécuter par le processeur.

Un *système de fichiers* désigne la manière dont les données sont stockées sur un support et gérées par le système d'exploitation. Il en existe une pléthore, certains plus utilisés que d'autres : New Technology File System (NTFS), High Performance File System (HPFS), DOS, FAT 12/16/32, VFAT, Macintosh Hierarchical File System (HFS), ISO 9660 (pour les CD-ROM), extended File System (ext, ext2, ext3), et de nombreux autres encore.

On peut par exemple envisager tout support physique de données (un disque dur, un cédérom...) comme une suite de petites cases, appelées *blocs*, contenant des informations. Chaque système de fichiers gère ces blocs différemment.

Il est possible de rajouter des fonctionnalités au dessus de ces systèmes, comme le chiffrement des fichiers : le support physique ne contient alors que des données chiffrées, illisibles pour qui ne possède pas la clé appropriée. Parmi les exemples, citons CFS [Bla93] pour Unix, TCFS [CP98] pour Linux et les BSDs, le patch kerneli [ker] pour Linux qui permet de chiffrer des partitions et des fichiers au travers du device `loop`, EFS [EFS98] qui chiffre des fichiers sous Windows 2000 ou SFS [Gut96] des partitions.

Lorsqu'un adversaire a connaissance de l'existence d'un fichier chiffré, il va mettre en œuvre tous les moyens dont il dispose pour récupérer la clé de déchiffrement et accéder ainsi aux données. Pour éviter cette situation, Anderson, Needham et Shamir [ANS98] proposent deux méthodes théoriques qui empêchent un attaquant de découvrir l'existence d'un fichier :

- l'utilisateur génère beaucoup de fichiers contenant des données aléatoires. Ces fichiers sont modifiés de sorte à ce que les données dissimulées soient reconstruites en faisant un XOR de tous les fichiers modifiés. Une clé secrète initialise un générateur aléatoire qui désigne alors les fichiers à employer ;
- les blocs d'un système de fichiers sont remplis aléatoirement. Lorsqu'un utilisateur désire cacher un fichier, il le chiffre, ce qui le fait ressembler à des données aléatoires. Il remplace ensuite certains blocs, désignés à l'aide d'une clé secrète et d'un générateur aléatoire, par ceux du fichier chiffré.

Chacune de ces méthodes possède des inconvénients. La première nécessite beaucoup de fichiers pour éviter qu'un attaquant n'essaye toutes les combinaisons et parvienne ainsi à reconstruire le fichier initial. Dans la seconde, rien n'indique qu'un bloc est occupé par des données ou de l'aléa. L'écriture d'un nouveau fichier peut donc entraîner des pertes dans les données déjà dissimulées puisque les blocs sont choisis sans possibilité de distinguer les fichiers utiles des factices. Cette méthode nécessite donc une grande redondance pour permettre de reconstruire les données dissimulées.

StegFS, proposé par McDonald et Kuhn dans [MK99], s'inspire de cette seconde solution. Ce système de fichiers ne crée pas de partition particulière, mais place ses données dans les blocs inutilisés du système de fichiers ext2[13]. Pour cela, une table d'allocation des blocs est créée exactement comme avec un système de fichiers normal, à la vue de tous ; elle contient les adresses des blocs

[13] ext2 est le système de fichiers de Linux.

qui dissimulent les informations. Cette table révèle que StegFS est installé sur la machine, mais d'autres traces font de même, comme la présence des outils nécessaires à son utilisation[14]. Si un attaquant est capable de constater que ce système de fichiers est présent sur une machine, il lui est néanmoins impossible de savoir s'il est utilisé pour dissimuler des informations.

2.2.2 Classification des schémas de stéganographie

Le contexte dans lequel se situe un schéma de stéganographie permet de le classer dans une des catégories suivantes :

- *stéganographie pure* : aucune entente préalable, autre que le choix de l'algorithme, n'est nécessaire, Alice et Bob utilisent le canal pour échanger des informations ;
- *stéganographie à clé secrète* : Alice et Bob conviennent au préalable d'une clé qui leur sert à insérer puis extraire le message du stégo-médium ;
- *stéganographie à clé publique* : tout comme en cryptographie, Alice utilise la clé publique de Bob lorsqu'elle souhaite lui envoyer un message. Bob, pour sa part, l'extrait à l'aide de sa clé privée.

Nous retrouvons également Wendy dans le même rôle que celui décrit par Simmons [Sim84] : elle surveille les échanges entre Alice et Bob.

2.2.2.1 Stéganographie pure

Un système stéganographique est pur lorsqu'il ne requiert aucun échange préalable d'information, comme une clé par exemple. S. Katzenbeisser propose dans [KP99a] une modélisation.

Soit la fonction d'insertion $\mathsf{E} : \mathsf{C} \times \mathsf{M} \rightarrow \mathsf{C}$, où C est l'ensemble de tous les média vierges possibles, et M l'ensemble des messages. Soit $\mathsf{D} : \mathsf{C} \rightarrow \mathsf{M}$ la fonction d'extraction du message. L'émetteur et le récepteur doivent tous deux posséder les fonctions E et D, mais l'algorithme général n'a pas besoin d'être public.

Définition 2.1 (Stéganographie pure) *Le quadruplet* $\mathcal{S} = (\mathsf{C}, \mathsf{M}, \mathsf{D}, \mathsf{E})$*, tel que* $\mathsf{D}(\mathsf{E}(\mathsf{c}, \mathsf{m})) = \mathsf{m}, \forall \mathsf{m} \in \mathsf{M}$*, et* $\mathsf{c} \in \mathsf{C}$*, est appelé* système de stéganographie pure.

Ce schéma nécessite que les média destinés à contenir un message ne soient pas rendus publics avant de transiter par le canal. En effet, si Wendy possède, par ailleurs, un médium m et qu'Alice décide d'envoyer un médium $\mathsf{m_A}$, alors Wendy peut savoir en comparant m et $\mathsf{m_A}$ si un message est caché.

2.2.2.2 Stéganographie à clé secrète

Avec la stéganographie pure, aucune information en dehors des fonctions E et D n'est requise : la sécurité du système repose entièrement sur le secret de l'algorithme, ce qui va à l'encontre du principe de Kerckhoffs énoncé au

[14] Les commandes classiques de manipulation de fichiers (`ls, rm, ...`) doivent aussi être adaptées à ce système.

$19^{ème}$ qui stipule que la sécurité d'un algorithme ne doit résider que dans un paramétrage par une clé, et non dans la non-divulgation de celui-ci.

La stéganographie est similaire au chiffrement à clé secrète (cf. B.2 page 190). Alice choisit donc à la fois le médium m et la clé secrète k. Si Bob, recevant le stégo-médium $\tilde{\mathsf{m}}$, ne dispose pas de la même clé k, il est dans l'incapacité d'extraire le message. Il est donc nécessaire, avec ce type de schéma, de prévoir également une solution pour transmettre la clé d'Alice à Bob, ce qui suppose l'existence d'un autre canal, sécurisé, entre Alice et Bob.

On note alors K l'ensemble de toutes les clés possibles. Les fonctions d'insertion et d'extraction sont réécrites ainsi :

$$\mathsf{E_K} : \mathsf{C} \times \mathsf{M} \times \mathsf{K} \rightarrow \mathsf{C}$$
$$\mathsf{D_K} : \mathsf{C} \times \mathsf{K} \rightarrow \mathsf{M}$$

Définition 2.2 (Stégo-Médium à clé secrète)
Le quintuplet $\mathcal{S} = (\mathsf{C}, \mathsf{M}, \mathsf{K}, \mathsf{E_K}, \mathsf{D_K})$*, tel que* $\mathsf{D_k}(\mathsf{E_k}(\mathsf{c}, \mathsf{m})) = \mathsf{m}, \forall \mathsf{m} \in \mathsf{M}$*, et* $\mathsf{c} \in \mathsf{C}$*, est appelé* système de stéganographie à clé secrète.

2.2.2.3 Stéganographie à clé publique

Là encore, le schéma est similaire à celui proposé en cryptographie à clé publique : un utilisateur dispose de deux clés, une publique pour dissimuler le message, l'autre privée pour l'extraire.

Une définition similaire à la précédente est obtenue en considérant que les clés sont différentes pour les fonctions d'insertion et d'extraction.

Dans la suite, nous noterons $\mathsf{k_{pu}}(\mathsf{X})$ la clé publique de X et $\mathsf{k_{pr}}(\mathsf{X})$ sa clé privée.

Soulignons néanmoins tout de suite que cette stéganographie pose plusieurs problèmes. Tout d'abord, comme la plupart des schémas asymétriques, elle est vulnérable à l'attaque dite de *l'homme au milieu* (ses étapes sont résumées dans le tableau 2.3 page suivante). Elle suppose que Wendy est active. Cette attaque se produit soit lors de l'établissement d'une clé de session comme avec le protocle d'échange de clé de Diffie-Hellman, soit lorsqu'Alice prend la clé publique de Bob mais qu'aucun certificat n'est disponible pour en vérifier la validité.

Un autre problème, plus particulier à la stéganographie, est détaillé en section 5.2.2 page 132. Précisons ici que Bob a besoin de tester chaque médium reçu pour savoir s'il contient ou non un message puisqu'aucun échange préalable n'est requis pour le prévenir que sa clé publique va servir à dissimuler une information.

2.3 Tatouage

La multiplication des données numériques, et plus encore la facilité avec laquelle il est possible de les reproduire, pose la question de la protection des droits d'auteur.

Deux approches sont possibles pour tenter de résoudre ce problème :

Homme au milieu

1. Alice envoie à Bob la paire $(\tilde{m}_0, k_{pu}(A))$, où $\tilde{m}_0$ est un médium contenant un message secret inséré à l'aide de la clé publique de Bob $k_{pu}(B)$
2. Wendy la remplace par $(\tilde{m}_0, k_{pu}(W))$ qu'elle fait suivre à Bob
3. Bob extrait le message de $\tilde{m}_0$ puis répond en utilisant $k_{pu}(W)$, qu'il prend pour $k_{pu}(A)$, et construit $\tilde{m}_1$
4. Wendy extrait à l'aide de sa clé privée le message de Bob
5. Wendy reconstruit un nouveau stégo-médium $\tilde{m}'_1$, grâce à la clé publique d'Alice qu'elle connaît, et le lui transmet
6. Alice lit la réponse de Wendy, qu'elle pense être celle de Bob

TABLEAU 2.3 : Attaque de l'homme au milieu

- brider l'outil qui effectue la copie ou la lecture du médium pour empêcher la reproduction : c'est le cas par exemple des lecteurs DVD qui nécessitent une clé pour déchiffrer le disque. De toute façon, le signal apparaît *en clair* à un moment donné (au moins sur l'écran), laissant libre cours au pirate pour le dupliquer ;
- protéger le médium lui-même en y insérant des données qui identifient son propriétaire.

La dissimulation d'information correspond exactement à cette seconde catégorie. Par la suite, cette idée a évolué pour proposer des applications variées.

Dans cette partie, nous présentons d'abord quelques applications du tatouage, avant de présenter une classification des solutions actuelles.

2.3.1 Applications du tatouage

2.3.1.1 Indexation et images intelligentes

Dans ce contexte, l'information dissimulée facilite la recherche de documents multi-média. L'indexation se décompose en deux processus :

- extraction des vecteurs caractéristiques (ou *feature vectors*) du médium ;
- recherche des média similaires à un médium servant de requête.

Il existe un grand nombre d'attributs représentatifs d'un médium. Par exemple, pour une image, on peut citer la décomposition dans différentes bases (DCT, Fourier, ondelettes ...), l'histogramme des couleurs, de l'orientation des arêtes...

Le but est de parvenir à décrire suffisamment une image pour que la recherche soit pertinente. Ainsi, lorsqu'on présente, en guise de requête, une image représentant la mer, on espère que la description faite au moyen des vecteurs

caractéristiques permettra de retrouver toutes les images possédant ces mêmes caractéristiques et donc un contenu identique.

Le tatouage sert alors à ajouter une information supplémentaire, comme un mot clé par exemple, qui facilite la recherche.

La société Digimarc[15] propose même d'aller plus loin : le tatouage inséré dans l'image contient un lien qui, lorsque l'image est présentée devant une caméra, est activé et dirige un navigateur vers des renseignements complémentaires. Cette société suggère d'utiliser ce mécanisme à des fins publicitaires.

Cette application du tatouage est celle qui demande le moins de robustesse. En effet, on suppose qu'un utilisateur ne va pas chercher à modifier l'information, qu'il s'agisse d'un index ou de publicité. Toutefois, un minimum de robustesse est quand même requis pour empêcher un autre annonceur de détourner la publicité initiale, soit en modifiant son contenu (les prix, les conditions de vente ...), soit en dirigeant le client vers sa propre annonce. Signalons toutefois le cas exceptionnel où la marque permet d'obtenir des informations auxquelles l'utilisateur ne doit normalement pas avoir accès. Les besoins en robustesse sont donc assez faibles. Au contraire, la capacité requise est élevée afin de dissimuler suffisamment de données pour que celles-ci soient exploitables.

Cette application ressemble donc plutôt à de la stéganographie, telle que nous l'avons décrite dans la partie précédente. Cependant, il existe une différence majeure qui justifie son appartenance au domaine du tatouage : le médium et les données ont un rapport.

2.3.1.2 Tatouage faible

Dans cette application, le but est de détecter les modifications subies par le médium. On distingue deux approches pour résoudre ce problème :

- le tatouage *fragile* : si le médium a subi plus de changements qu'autorisés, la détection de la marque échoue ;
- le tatouage *fragile évolué* : si le médium a été altéré, la marque indique l'endroit où se sont opérées les modifications.

La différence majeure entre les deux utilisations provient des besoins en capacité. En effet, la seconde approche nécessite une capacité plus élevée puisqu'il faut parvenir à décrire le médium dans la marque qui est dissimulée.

Par exemple, [SC96] propose d'extraire des vecteurs caractéristiques d'une image, de hacher cette information pour en réduire la taille, puis de la chiffrer avec sa clé secrète avant de la dissimuler dans l'image. Puisque l'insertion de la signature modifie l'image, celle-ci doit se faire dans des régions qui n'interviennent pas dans le processus de vérification. Les auteurs proposent par exemple de la dissimuler dans les coefficients moyens des blocs 8×8 d'une DCT. L'inconvénient de cette méthode est qu'il n'est pas facile de sélectionner les caractéristiques. Il faut en effet en prendre assez pour décrire l'image, mais pas trop non plus afin de pouvoir insérer cette marque dans l'image dans une région suffisamment robuste. D'autres solutions ont été proposées et reposent

[15] http://www.digimarc.com

sur des $m-$sequences[16] ([WD96]) ou sur une indexation des couleurs des pixels ([YM97]).

Les travaux plus récents portent davantage sur la seconde approche. Ainsi, [KH99] propose une solution pour détecter le type de modifications subies par une image. De plus, comme elle est fondée sur une transformée en ondelettes discrète, elle permet de localiser en espace et en fréquence les emplacements concernés.

2.3.1.3 Protection des droits d'auteur

Dans un cadre non militaire, il s'agit de la première application pour laquelle la dissimulation d'information est utilisée. Dans la suite de ce mémoire, nous nous concentrons essentiellement sur cet aspect lorsque nous traitons de tatouage.

2.3.2 Classification des méthodes de tatouage

Nous introduisons tout d'abord les différents types de schémas définis selon les paramètres dont ils ont besoin lors de l'extraction. Nous présentons ensuite une classification des algorithmes de tatouage.

2.3.2.1 Types de schémas

Le propriétaire d'un médium possède une clé qui lui est propre, et qui l'associe à ce médium lors de l'insertion de la marque.

On distingue deux types de fonction de détection D :

- type I : la fonction D extrait la marque elle-même ;
- type II : la fonction D vérifie uniquement la présence de la marque à l'aide d'une mesure de confiance ;

D'autres caractéristiques hormis celles liées à l'extracteur entrent en jeu. C'est ainsi qu'on distingue les algorithmes de tatouage suivants :

- *le tatouage privé*, où le médium initial, la marque à tester et la clé sont donnés à l'extracteur. Dans ce type de tatouage, on compare l'original au stégo-médium récupéré pour extraire la marque ;
- *le tatouage semi-aveugle* utilise une fonction de détection qui nécessite la marque à tester et la clé ;
- *le tatouage aveugle*, où l'extracteur n'a pas connaissance ni du médium original, ni de la marque. Seule la clé secrète lui est nécessaire pour détecter ou extraire la marque du stégo-médium ;
- *le tatouage asymétrique*, où l'extraction de la marque ne nécessite pas la connaissance d'un secret. Cela implique que tout le monde est capable de lire la ou les marques du stégo-médium sans pouvoir les effacer. Cela pourrait se faire par un tatouage sans clé ou alors par un tatouage avec

[16] Il s'agit de vecteurs binaires construits de sorte à ce qu'ils soient orthogonaux deux à deux.

clé secrète et une extraction avec la clé publique correspondante (dans un schéma analogue à celui de la cryptographie asymétrique).

Les deux premiers modèles d'algorithmes de tatouage utilisent donc la connaissance d'un secret pour l'extraction de la marque. Les protocoles mis en place au-dessus de ces algorithmes nécessitent l'existence d'un tiers de confiance dépositaire de ce secret.

Le tatouage privé reste très lourd à manier puisque le tiers de confiance doit posséder l'original de tous les stégo-média marqués. La base de données risque alors d'être gigantesque et peu pratique à utiliser de façon confortable et sécurisée.

Dans le cas du tatouage aveugle, la taille de la clé est beaucoup plus petite que celle du médium à protéger. C'est donc une solution plus satisfaisante que celle du tatouage privé, qui ne convient *a priori* pas pour des applications viables économiquement.

Le tatouage asymétrique représente la panacée. Plus besoin de tiers de confiance, la marque est une propriété du médium que tout le monde peut lire sans pour autant pouvoir la retirer.

2.3.2.2 Choix du domaine

Traditionnellement, on distingue les schémas de tatouage selon le domaine sur lequel ils agissent. On dispose alors essentiellement de méthodes *spatiales*, *fréquentielles* ou *multi-résolutions*. Néanmoins, un autre critère de séparation existe en fonction de la manière dont est inséré le tatouage : soit il est ajouté au médium, soit il en remplace certains coefficients (de plus amples détails sont fournis dans la thèse de P. Bas [Bas00]).

Chaque espace de travail utilisé en tatouage possède ses propres avantages et inconvénients.

Les méthodes agissant dans le domaine spatial modifient directement les valeurs des pixels. Comme aucun traitement initial n'est requis, ces algorithmes sont très rapides et permettent de travailler en temps réel. De plus, elles offrent souvent une bonne résistance aux opérations géométriques.

Le schéma du «patchwork», proposé initialement par Bender *et al.* [BGML96] puis amélioré par Pitas [Pit96], se situe dans ce domaine. L'utilisateur dispose d'une clé secrète dont il se sert pour initialiser un générateur aléatoire. Celui-ci permet alors de sélectionner n paires de pixels (a_i, b_i), lesquelles sont modifiées de la manière suivante :

$$\begin{aligned} \tilde{a}_i &= a_i + 1 \\ \tilde{b}_i &= b_i - 1 \end{aligned}$$

Pour la détection, les n paires sont de nouveau sélectionnées à l'aide de la clé secrète et la somme $S = \sum_{i=1}^{n} (\tilde{a}_i - \tilde{b}_i)$ est calculée. Si la marque est présente, alors la moyenne $E(S)$ est égale à $2n$, 0 dans la cas contraire. Ce mécanisme de détection repose sur l'hypothèse que lorsqu'on choisit aléatoirement plusieurs paires de pixels, la valeur moyenne de leur différence est nulle.

Cependant, un tel schéma n'offre qu'une protection minime contre une compression : il suffit par exemple de compresser l'image en JPEG avec un faible taux de compression pour détruire la marque.

L'intérêt des schémas de tatouage s'est ensuite naturellement porté vers les domaines fréquentiels (DCT - *Discrete Cosine Transform* - et DFT - *Discrete Fourier Transform*). Ces espaces de travail sont fréquemment utilisés, par exemple dans les normes JPEG et MPEG2. Les schémas agissant dans ces domaines gagnent alors une certaine robustesse contre la compression.

Les algorithmes fonctionnant avec la DCT ne sont pas très résistants aux transformations géométriques comme les translations ou les rotations car celles-ci affectent grandement les coefficients de la DCT. Au contraire, l'espace de Fourier possède des propriétés d'invariance aux translations et rotations qui augmentent la fiabilité de la méthode. De plus, l'utilisation d'une modulation de phase entre la marque et l'image offre d'autres avantages [ODB96] :

- Hayes [Hay82] a montré que la perception de la composante liée à la phase est plus significative que celle liée à l'amplitude. Ainsi, l'insertion de la marque dans cette composante, avec une grande redondance, contraint un attaquant à grandement endommager l'image pour retirer la marque ;
- la modulation de phase possède une meilleure robustesse au bruit.

Enfin, on voit maintenant apparaître de plus en plus de méthodes dans le domaine multi-résolutions. Cette évolution semble naturelle dans la mesure où elle suit celle des standards récents comme JPEG2000 et MPEG4.

Mais au-delà du domaine dans lesquels ils agissent, les schémas de tatouage se distinguent essentiellement par leur manière d'insérer la marque dans le médium.

2.3.2.3 Schémas additifs

Lors de l'insertion, le signal représentant la marque est ajouté à certaines composantes du médium. Pour y parvenir, il s'agit d'adapter la marque au médium, afin que le signal qu'elle représente ne soit ni trop faible (risques de non détectabilité et problèmes de robustesse), ni trop fort (effacement du signal initial, et donc trop grande dégradation de celui-ci).

Le tableau 2.4 présente le principe d'insertion par addition. La génération de la marque $\mathsf{w_k}$ se fait généralement par *étalement de spectre*. Cette technique est utilisée dans les systèmes de communication afin de transmettre un message au travers d'un canal bruité [PS94]. en tatouage, elle permet de retrouver la marque malgré le médium. Avec Le *direct-sequence spreading* (cf. tab. 2.5 page 42), la marque, signal basse fréquence et constituée de n bits, est suréchantillonnée, produisant un signal binaire de $\mathsf{n} \times \mathsf{r}$ bits. Un bruit large bande est généré aléatoirement à l'aide d'une clé k puis un OU-exclusif est effectué entre les deux signaux pour obtenir le signal $\mathsf{w_k}$ ajouté au médium.

La détection de la marque dans le stégo-médium est obtenue en analysant la corrélation entre le stégo-médium et la séquence $\mathsf{w_k}$. Le stégo-médium est considéré comme un bruit, relativement à l'information contenue dans le tatouage.

Des solutions [TRvS+93, vSTO94] furent élaborées dès 1993 dans le domaine

Insertion par addition de la marque au signal

1. extraire des coefficients du médium initial m
2. réordonner ces coefficients selon une permutation paramétrée par une clé k pour obtenir un vecteur $\mathsf{c_k(m)}$
3. générer une marque $\mathsf{w_k}$, dépendante ou non du médium initial
4. tatouer le médium en ajoutant la marque aux coefficients :
$$\mathsf{c_k(m_w) = c_k(m) + w_k}$$
5. réordonner les coefficients puis reconstruire le médium tatoué $\mathsf{m_w}$

TABLEAU 2.4 : Insertion dans un schéma de tatouage additif

spatial. I. Cox fut le premier à proposer l'emploi de ces techniques dans le domaine fréquentiel ([CKLS96]) en considérant la DCT appliquée intégralement à l'image. Depuis, de nombreux travaux ont contribué à améliorer l'efficacité, tant au niveau de l'insertion que de la détection, comme par exemple [BPBC97] ou [DDM98] qui ne nécessitent plus l'image initiale pour détecter la présence ou non de la marque.

2.3.2.4 Schémas substitutifs

La différence majeure entre les schémas additifs et substitutifs provient de la phase d'insertion. Dans ces derniers, au lieu d'ajouter un signal au médium, certaines composantes sont remplacées afin que le stégo-médium exhibe une propriété caractéristique. Lors de la phase de détection, si cette particularité est présente, le stégo-médium est considéré comme tatoué avec une marque donnée.

La substitution n'est pas systématique en fonction de la situation du paramètre. Par exemple, l'échange peut être conditionné par un critère qualitatif, comme nous l'avons vu en 2.2.1.2 avec le logiciel `outguess` en stéganographie.

Zhao et Koch [ZK98] furent des précurseurs avec ce type d'approches. Leur schéma repose sur l'utilisation d'une DCT 8×8 afin d'être plus résistant à la compression JPEG. Ils proposent de ne modifier que les coefficients moyens des blocs, en fonction du bit à insérer, de manière à créer une relation d'ordre artificielle entre ces coefficients. Dans le domaine spatial, citons également les travaux de P. Bas [BCD98] dont le principe est de remplacer des blocs d'une image par des blocs similaires ; la détection s'effectue alors en recherchant ces similarités.

Kundur *et al.* [KH98] proposent une méthode similaire à celle de Zhao et Koch, mais à l'aide d'ondelettes. L'insertion est réalisée en modifiant l'ordonnancement des trois sous-bandes horizontale, verticale et diagonale à une échelle donnée. Davoine [Dav00] compare ce schéma à une nouvelle technique qui partitionne l'union des détails à basse résolution en régions distinctes comportant le

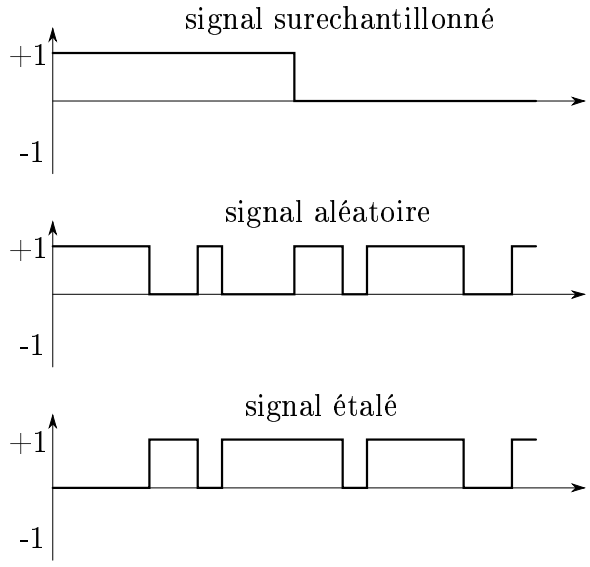


TABLEAU 2.5 : *Direct-sequence spreading*

même nombre de coefficients significatifs. En ne se limitant plus à des triplets, il est ainsi possible d'adapter la robustesse en fonction du nombre de coefficients considérés. Enfin, Manouri em et al. [MLL99] utilisent des paquets d'ondelettes. La méthode modifie la meilleure base extraite du paquet d'ondelettes en imposant une parité[17] donnée sur des sous-bases déterminées par une clé secrète.

2.4 Fingerprinting

L'objectif du fingerprinting est de reproduire ce qui se passe avec nos empreintes digitales : dès que nous touchons un objet, nous les y déposons.

Par rapport à la dissimulation d'information dans un médium, l'objectif du fingerprinting est de permettre l'identification de la provenance d'une copie illégale. En aucun cas ces schémas ne cherchent à prévenir la fabrication de la copie. On souhaite simplement être capable de retrouver la source de la copie.

Si une copie distincte d'un même médium est fournie à chaque utilisateur, ceux-ci peuvent examiner toutes les versions disponibles afin de découvrir les différences. Ils ont alors la possibilité de reconstruire un nouveau médium en prenant des extraits de chaque copie. La figure 2.6 illustre ceci dans un cas très simple de collusion : toutes les différences sont reportées dans la nouvelle image. Cet exemple montre bien que cette approche n'est pas satisfaisante pour les attaquants : il est possible de remonter à l'origine des deux copies.

Pour fonctionner correctement, un schéma de fingerprinting doit donc sa-

[17]Les vecteurs qui appartiennent à la meilleure base sont représentés par 1, les autres par 0, ce qui permet de les sommer à une échelle donnée pour obtenir la parité à cette échelle..

Insertion par substitution de la marque au signal

1. extraire des coefficients du médium initial m
2. réordonner ces coefficients selon une permutation paramétrée par une clé k pour obtenir un vecteur $\mathsf{c_k(m)}$
3. générer une marque $\mathsf{w_k}$, dépendante ou non du médium initial
4. tatouer le médium en remplaçant certains coefficients $\mathsf{c_k(m)}$ par ceux nouvellement générés par la marque :
$$\mathsf{c_k(m_w)} = (\text{condition}\,?\ \mathsf{w_k} : \mathsf{c_k(m)})$$
5. réordonner les coefficients puis reconstruire le médium tatoué $\mathsf{m_w}$

TABLEAU 2.6 : Insertion dans un schéma de tatouage par substitution

tisfaire les mêmes exigences qu'un schéma de tatouage, mais également être résistant aux attaques par collusions.

Des solutions ont été proposées dans [BS95] et [CEZ99] pour détecter les collusions à l'aide de la théorie des codes. L'empreinte est un mot d'un code. Lorsque plusieurs mots sont assemblés pour constituer une nouvelle empreinte invalide, il est alors possible de retrouver les mots ayant servis à sa fabrication.

Conclusion

Dans ce chapitre, nous avons présenté les objectifs de la dissimulation d'information. Selon les attentes, les contraintes d'imperceptibilité, de capacité et de robustesse varient. Cependant, comme ces besoins sont à l'encontre les uns des autres, un compromis est toujours nécessaire.

Par ailleurs, ce chapitre montre la diversité des approches proposées. La dissimulation d'information du moins dans sa version informatique, est une discipline récente. Elle connaît en effet un grand essor depuis la seconde moitié des années 1990.

Les recherches portent actuellement autant sur la mise au point d'algorithme d'insertion et de détection que de protocoles susceptibles de les encadrer afin d'en accroître la sécurité. Dans cette perspective, nous présentons dans la suite de ce mémoire notre étude sur des outils issus de la théorie fractale à des fins de tatouage. Ensuite, nous étudions différents protocoles, soit pour évaluer les méthodes de tatouage, soit ceux définis en cryptographie.

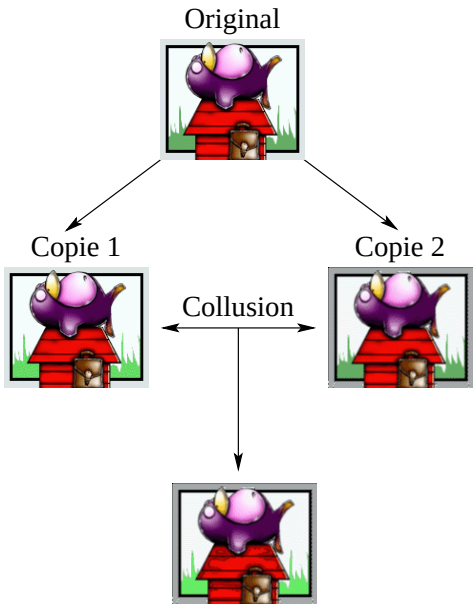


FIGURE 2.6: Collusion. Dans la copie 1, l'herbe est plus claire. Dans la copie 2, la cadre est gris. L'union de ces deux copies contient ces deux différences par rapport à l'image originale.

Première partie

Outils fractals pour le tatouage : IFS et spectres mutuels

Introduction

Il existe proportionnellement assez peu de méthodes de tatouage s'appuyant sur les fractales ([PJ96, BCD98, RD98]). Dans cette partie, nous nous intéressons à deux domaines issus de cette théorie : les *systèmes de fonctions itérées* (*Iterated Functions System* – IFS) et l'*analyse multifractale*. Les méthodes de tatouage se situent essentiellement soit dans le domaine spatial, soit dans le domaine fréquentiel ou multi-résolution. Les outils fractals constituent un intermédiaire entre ces différentes représentations.

Tout d'abord, le schéma de tatouage proposé par Puate et Jordan dans [PJ96] repose sur un algorithme de compression fractale ([Jac93, Fis95]) dont le but est d'exprimer l'image à compresser comme étant l'attracteur d'un système de fonctions itérées. La détermination de ce système est réalisée sous contraintes, l'image décompressée (*i.e.* l'attracteur de l'IFS) devenant par là-même l'image tatouée. Nous montrons cependant que ce schéma comporte de nombreuses lacunes.

Ce schéma de tatouage correspond à un problème de génération d'IFS sous contraintes. Les algorithmes évolutionnaires offrent une perspective intéressante pour ce type de problèmes d'optimisation, en particulier lorsqu'il est tout autant important d'explorer l'espace de recherche que de parvenir à un optimum. Deux problématiques se distinguent alors :

- d'une part, générer des attracteurs satisfaisant à des contraintes géométriques comme la densité de points ;
- d'autre part, retrouver un IFS à partir d'un attracteur donné.

Des IFS construits avec des fonctions affines ont déjà été proposés pour résoudre ces questions ([Vrs90, LL93]). Dans un premier temps, nous avons opté pour des IFS mixtes (non-linéaires) afin de relâcher les hypothèses faite lors de la compression fractale. La complexité de ces derniers ne donnant toutefois pas suffisamment de résultats, nous lui avons préféré une sous-classe, les IFS *polaires*, dont la convergence est plus facile à contrôler.

Les schémas évolutionnaires classiques ne sont néanmoins pas très bien adaptés à ce type de problèmes. Nous avons alors contribué à l'élaboration d'une nouvelle approche, dit *individuelle* ou *Parisienne* [CLRS00], qui se distingue par une représentation différente : un individu de la population ne constitue plus une solution à part entière mais une contribution à la solution générale. Nous montrons également en quoi ce changement s'avère pertinent pour certains problèmes, comme ceux de la génération sous contraintes d'IFS.

Ensuite, après avoir introduit l'analyse multifractale qui quantifie la régularité d'un signal, nous détaillons les expériences réalisées pour tenter de mettre en place un schéma de tatouage reposant sur l'emploi du spectre multifractal d'une image. Nous nous intéressons en particulier à la détermination de paramètres caractéristiques sur le spectre de Hausdorff qui pourraient faire figure de marque lorsque leur valeur est contrainte, comme cela se passe avec les schémas substitutifs.

Chapitre 3

Théorie des IFS, tatouage et algorithmes évolutionnaires

Après avoir introduit la théorie des IFS et son application à la compression fractale, nous présentons puis analysons la solution proposée par Puate et Jordan [PJ96]. L'idée centrale de cette méthode consiste à ajouter une contrainte supplémentaire pendant la génération du code IFS de l'image à tatouer. Nous avons alors étudié ce problème de génération contrainte sous deux angles :

- générer des attracteurs possédant certaines caractéristiques géométriques ;
- retrouver un IFS à partir d'un attracteur donné.

La plupart des études menées sur ces questions traitent d'IFS composés de fonctions affines. Nous avons voulu relâcher cette hypothèse et construire des IFS à l'aide de fonctions quelconques[1]. Cependant, le prix de cette liberté est une perte presque totale de contrôle sur les IFS : d'une part, il est très difficile d'estimer, au niveau informatique, si une fonction non-linéaire est contractante ou pas ; d'autre part, la densité de fonctions non-linéaires et contractantes est assez faible dans l'espace des fonctions.

Afin de contourner ces difficultés, nous avons étudié une classe intermédiaire d'IFS : les *IFS polaires*, dont le nom provient de l'expression en coordonnées polaires des fonctions qui les constitue. De plus, elles possèdent, de par leur structure, la propriété d'être *localement contractantes* vis-à-vis d'un point. Nous montrerons que la construction des IFS polaires nous rend une grande partie du contrôle perdu lors du passage aux IFS mixtes.

Pour manipuler ces divers types d'IFS, nous avons utilisé un algorithme évolutionnaire, algorithme d'optimisation stochastique où un ensemble de points (appelés *individus*) de l'espace de recherche parcourt celui-ci à la découverte d'optima (un rappel des notions élémentaires est fourni en annexe A page 181). Il nous a fallu pour cela modifier l'approche classique de ces algorithmes en proposant une *approche Parisienne*[2] ou *individuelle*. Nous présentons donc cette

[1] On parle d'*IFS mixtes* pour les distinguer.

[2] Dénommée ainsi par référence à l'approche «Pittsburgh» et l'approche «Michigan» des systèmes de classeurs

variante et les résultats obtenus sur les problèmes de générations d'IFS sous contraintes. Ces résultats sont également disponibles dans [CLRS99a, RLCS99, CLRS99b, CLRS00].

3.1 Introduction à la théorie des IFS

Les systèmes de fonctions itérées (*Iterated Functions System* - IFS) constituent un pan important des fractales et fournissent des outils adaptés à l'étude des ensembles fractals (voir [Hut81, BD85, Har85]).

Les recherches pratiquées dans le domaine des IFS visent essentiellement à résoudre le *problème inverse* (voir 3.1.3 page 52) pour des courbes 1D, des ensembles 2D ou des mesh 3D (par exemple, une image et les niveaux de gris de ses pixels). Une solution exacte n'est calculable que dans des circonstances particulières. Généralement, nous devons nous contenter d'une approximation, obtenue par des méthodes d'optimisation.

Les résultats de ces optimisations servent, entre autre, en traitement du signal (pour obtenir une représentation fonctionnelle adaptée à l'interpolation [LDL94]) ou en compression [Bar93, Jac93, Fis95].

3.1.1 Notations et définitions

Définition 3.1 *Soient (F, d) un espace métrique complet, et $(w_i)_{i=1,..,N}$ une collection de fonctions définies de F dans F.*

$\Omega = \{F, (w_i)_{i=1,..,N}\}$ *est appelé* IFS.

La notion principale dans la théorie des IFS est la contractance :

Définition 3.2 *Une fonction $w : F \to F$, d'un espace métrique (F, d) dans lui-même, est dite* contractante *s'il existe un réel positif $s < 1$ tel que :*

$$\forall (x, y) \in F^2, \quad d\big(w(x), w(y)\big) \leq s.d(x, y) \tag{3.1}$$

Le plus grand de ces réels s est appelé facteur de contractance *de w.*

Le théorème 3.1 présente un résultat sur la convergence d'une fonction contractante :

Théorème 3.1 (Théorème du point fixe) *Soient (F, d) un espace métrique complet, et $w : F \to F$ une fonction contractante, alors w possède un unique point fixe.*

Avant de présenter une propriété importante sur la convergence des IFS, nous rappelons ce que sont l'opérateur de Hutchinson et la distance de Hausdorff.

À l'aide d'un IFS $\Omega = \{F, (w_i)_{i=1,..,N}\}$, on définit l'*opérateur de Hutchinson* W sur l'espace des sous-ensembles de F :

$$\forall K \subset F,\ W(K) = \bigcup_{i \in \{0...N\}} w_i(K)$$

Définition 3.3 *La distance de Hausdorff entre deux ensembles A et B de (F, d) est définie par :*

$$d_H(A,B) = \max\left(\max_{x\in A}\left(\min_{y\in B} d(x,y)\right), \max_{y\in B}\left(\min_{x\in A} d(x,y)\right)\right)$$

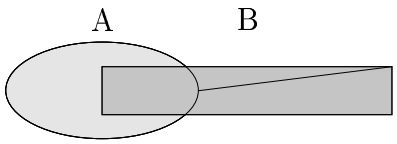


FIGURE 3.1 : Distance de Hausdorff $d_H(A, B)$

Si chaque fonction w_i qui compose l'IFS Ω est contractante (de facteur s_i), Ω est alors lui aussi contractant par rapport à la distance de Hausdorff. Son facteur de contractance est donné par :

$$s_\Omega = \sup_{i=1,..,N} \{s_i\}$$

Ceci nous permet de parvenir à la propriété importante sur la convergence des IFS :

Proposition 3.1 *Si un IFS $\Omega = \{F, (w_i)_{i=1,..,N}\}$ est contractant, alors il existe un unique ensemble $A \subset F$, appelé* attracteur, *tel que $W(A) = A$.*

L'unicité de l'attracteur découle du théorème du point fixe appliqué à W, qui est contractant relativement à la distance de Hausdorff :

3.1.2 Calcul de l'attracteur

Deux méthodes principales permettent de calculer l'attracteur de l'IFS par des approches différentes :

1. *algorithme déterministe* : à partir d'un sous-ensemble K_0 de F, construire la suite $\{K_n\}$ de sous-ensembles de F :

$$K_{n+1} = W(K_n) = \bigcup_{i=1}^{N} w_i(K_n)$$

Pour des valeurs de n suffisamment grandes, K_n tend vers l'attracteur de Ω.

2. *algorithme stochastique* (également appelée *toss-coin* : soit x_0 le point fixe d'une des fonctions w_i, construire la suite de points x_n, telle que $x_{n+1} = w_i(x_n)$ où i est choisi aléatoirement parmi $\{1..N\}$. L'ensemble $K = \bigcup_n \{x_n\}$ tend vers l'attracteur de Ω pour n assez grand.

En pratique, on utilise plus couramment la seconde approche car elle nécessite beaucoup moins de calculs. De plus, à condition de partir d'un point situé dans l'attracteur[3], le parcours ne sort alors jamais de l'attracteur. Au contraire, la première méthode impose d'estimer chaque w_i en tout point de K_n.

3.1.3 Problème inverse pour les IFS

On énonce le problème inverse pour les IFS en 2D de la manière suivante :

> Trouver un IFS contractant dont l'attracteur est exactement une forme donnée (*i.e.* une image binaire).

Toutefois, la contrainte d'exactitude est relâchée dans la majorité des cas car inapplicable à des images non fractales, transformant le problème en :

> Trouver un IFS contractant dont l'attracteur est aussi proche que possible d'une forme donnée (*i.e.* une image binaire), au sens d'une mesure inter-images prédéfinie.

Le théorème du collage fournit une réponse adaptée à cette version du problème :

Théorème 3.2 (théorème du collage ([BEHL86])) *Soient A l'attracteur de l'IFS $\Omega = \{F, (w_i)_{i=1,..,N}\}$, et λ son facteur de contractance. Alors :*

$$\forall K \subset F, \quad d_H\big(K, W(K)\big) < \varepsilon \quad \Longrightarrow \quad d_H(K, A) < \frac{\varepsilon}{1-\lambda},$$

Il s'agit alors de minimiser $d_H\big(K, W(K)\big)$ afin de s'approcher le plus possible de l'attracteur réel. Cependant, lorsque le facteur de contractance est proche de 1, la borne perd tout son intérêt, ce qui montre la nécessité d'une estimation fine de ce facteur. Lorsque celle-ci est précise, il est alors mieux de minimiser $\frac{1}{1-\lambda} d_H\big(I, \bigcup_{i=1}^{N} w_i(I)\big)$ pour que la majoration conserve son sens.

3.1.4 Application à la *compression fractale*

Si la fougère de Barnsley ou le triangle de Sierpinski sont construits à partir de répliques réduites de l'image elle-même, ce n'est toutefois pas le cas de la majorité des images. Dans le cas de la fougère (cf. fig. 3.2 page suivante), celle-ci est *auto-similaire*. Lorsqu'il s'agit d'images de paysages ou de visages, cette propriété n'est plus vérifiée. Par ailleurs, les IFS que nous avons présenté jusqu'à

[3] On débute généralement d'un des points fixes des fonctions w_i.

maintenant ne permettent que de coder des images binaires. Or nous souhaitons utiliser cet outil sur des images en niveaux de gris. Il est donc nécessaire d'adapter cet outil pour représenter une image.

$$
\begin{aligned}
w_0(x,y) &= \begin{pmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.075 \\ 0.18 \end{pmatrix} \\
w_1(x,y) &= \begin{pmatrix} 0.2 & -0.26 \\ 0.23 & 0.22 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.4 \\ 0.045 \end{pmatrix} \\
w_2(x,y) &= \begin{pmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.575 \\ -0.086 \end{pmatrix} \\
w_3(x,y) &= \begin{pmatrix} 0 & 0 \\ 0 & 0.16 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.5 \\ 0 \end{pmatrix}
\end{aligned}
$$

FIGURE 3.2 : La fougère de Barnsley et son code IFS

Au lieu de rechercher une similarité de l'image entière dans elle-même, la compression fractale utilise le fait que des morceaux d'une même image sont similaires, à une transformation affine près. La figure 3.3 page suivante montre ces similarités, présentes à différentes échelles, entre l'épaule de Lena, son chapeau et le reflet de ce dernier dans le miroir. Mais la recherche de similarités ne s'effectue pas uniquement sur l'aspect géométrique : le niveau de gris de la région est aussi pris en compte.

Pour parvenir à ce résultat, les PIFS (*Partitioned Iterated Functions System*) ne considèrent plus l'image dans son intégralité mais en tant qu'une partition de blocs. Dans leur construction, les PIFS ne sont pas limités à un type donné de transformations. Toutefois, en pratique, seules des fonctions affines sont employées. Ainsi, pour un pixel positionné en (x, y) et d'intensité z, on note :

$$
w_i \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix} \tag{3.2}
$$

La restriction v_i donnée par $v_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix}$ gère la partie géométrique de la transformation, celle qui permet de déplacer et d'ajuster la taille des blocs entre eux. Les coefficients s_i et o_i jouent sur les niveaux de gris. Afin que la fonction w_i soit contractante, il faut et il suffit que $s_i < 1$.

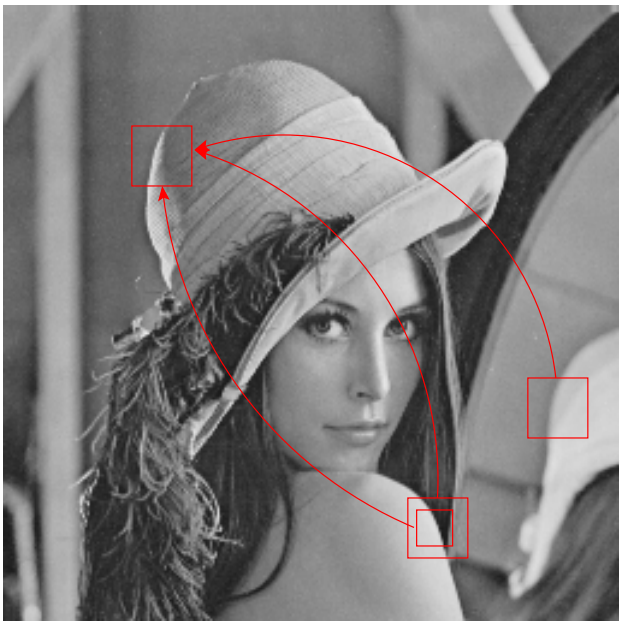


FIGURE 3.3: Régions similaires dans Lena *via* une transformation affine (géométrie et niveaux de gris)

L'image à compresser est partitionnée en régions de petite taille afin de limiter les effets visuels de la compression. Ces blocs sont appelés *blocs destinations* (ou *ranges*). On construit également une bibliothèque de blocs, plus gros, et caractéristiques de l'image : les *blocs sources* (ou *domaines*).

Pour chaque bloc destination $R_i = (y_1, \ldots, y_n)$, l'algorithme de compression recherche le «meilleur» antécédent parmi tous les domaines. On applique tout d'abord la transformation géométrique v_i au domaine candidat pour obtenir $D_i = (x_1, \ldots, x_n)$. On détermine alors les coefficients s et o qui transforment les niveaux de gris de D_i en ceux de R_i. Le meilleur antécédent est celui qui minimise l'erreur quadratique lors de cette étape :

$$err = \sum_{i=1}^{n} (s\ y_i + o - x_i)^2$$

Cette valeur minimale est atteinte lorsque les dérivées partielles par rapport à s et o sont nulles, ce qui donne comme résultat pour s et o :

$$s = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \qquad \text{et} \qquad o = \frac{1}{n}\left(\sum_{i=1}^{n} y_i - s\sum_{i=1}^{n} x_i\right)$$

L'ensemble des fonctions w_i construites correspond au *code IFS* de l'image.

Dans un second temps, pour reconstruire l'image, on applique les transformations w_i contractantes à partir d'une image initiale quelconque. Après plusieurs itérations, comme le stipule la théorie des IFS, la suite converge vers son point fixe : l'image compressée.

Le tableau 3.1 résume les différentes étapes qui interviennent dans la création de l'image compressée. La figure 3.4 page suivante illustre le procédé de reconstruction.

Algorithme de compression fractale

1. Détermination du *code IFS* W de l'image :
   - partitioner l'image I en n *blocs destination* R_i :

$$I = \bigcup_{i=1}^{n} R_i$$

     L'ensemble des ranges est noté R.
   - construire un ensemble D composé de *blocs sources*.
   - pour chaque bloc R_i :
     - déterminer l'élément de D, et la transformation w_i associée, qui correspond «au mieux» à R_i ;
     - sauvegarder w_i.
2. Calcul de l'attracteur correspondant :

$$lim_{n \to +\infty} W^n(I_0) = \tilde{I}$$

TABLEAU 3.1 : Algorithme de compression fractale

3.2 Tatouage par IFS contraints

Les premières solutions de tatouage ont proposé d'utiliser les bits de poids faible des signaux ([Tur89, vSTO94]), mais la moindre compression effaçait alors la marque. I. Cox a montré dans [CKLS97] qu'il fallait donc utiliser les composantes contenant le plus d'informations (bits de poids forts, basses fréquences ... selon les domaines) du signal pour y dissimuler la marque afin d'obtenir une solution résistant à une compression quelconque.

Suite à ce constat, les méthodes de tatouage proposées se sont beaucoup appuyées sur les algorithmes de compression. En particulier, quelques unes se sont intéressées aux possibilités offertes par la compression fractale ([PJ96, BCD98, RD98]). Nous étudions ici la solution proposée par Puate et Jordan [PJ96]. Nous présentons tout d'abord des problèmes non résolus qui apparaissent à la lecture de l'article.

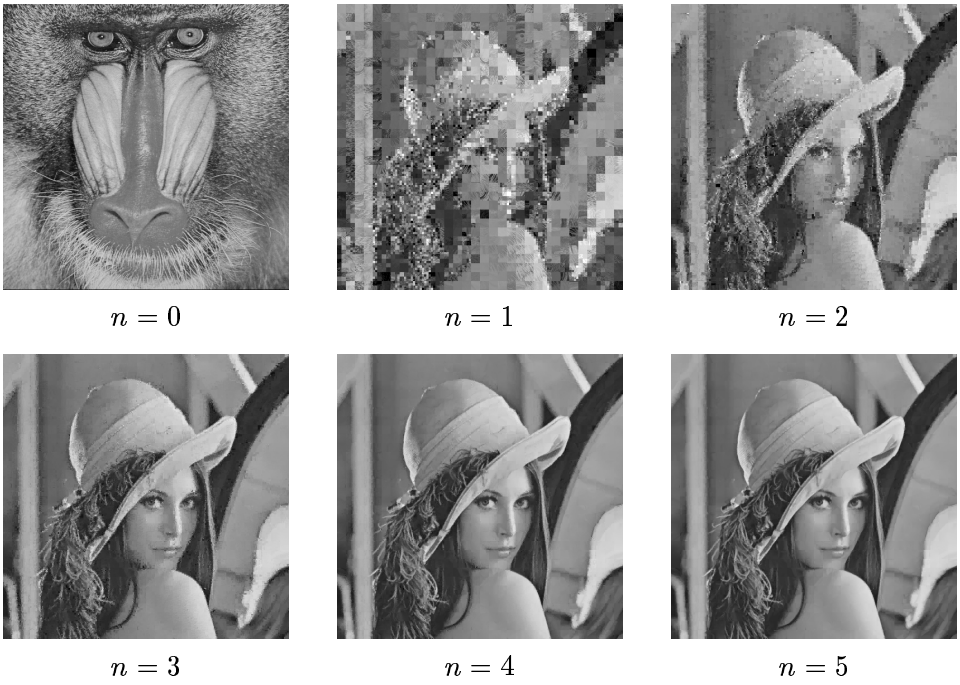


FIGURE 3.4: Reconstruction de l'image de Lena à partir du code IFS correspondant en se servant du mandrille comme image initiale.

Avant de continuer, précisons que les algorithmes de compression, dont est issue cette méthode, optimisent soit le temps de calcul (essentiellement le nombre de comparaisons domaine / range), soit la qualité de l'image obtenue (*i.e.* minimiser l'erreur entre l'image initiale et l'image compressée). En tatouage, seules quelques applications spécifiques ont réellement besoin d'être rapides. Le facteur le plus important est donc la qualité de l'image obtenue. De plus, nous pouvons nous permettre de nous affranchir des contraintes liées à la compression puisque nous ne cherchons nullement à diminuer l'espace disque occupé par l'image.

3.2.1 Présentation de la méthode

La méthode proposée repose sur un algorithme de compression fractale. Nous avons vu (cf. 3.1.4 page 52) que le principe en est de partitionner l'image à compresser en blocs destination pour rechercher dans une bibliothèque de domaines celui qui correspond le mieux à chaque range.

Puate et Jordan proposent de contraindre cette recherche pour insérer une marque de 32 bits. Elle est souvent effectuée dans un voisinage du bloc destination, appelé LSR (*Local Searching Region*). Cette région LSR est divisée en deux parties appelées LSR_A et LSR_B (voir 3.5 page suivante). Une clé secrète k permet d'initialiser un générateur aléatoire qui désigne les blocs destination qui contiendront un bit de la marque $s = (s_0, \ldots, s_{31})$: pour chaque bloc considéré, selon la valeur du bit s_i à dissimuler, l'un ou l'autre des sous-domaines (res-

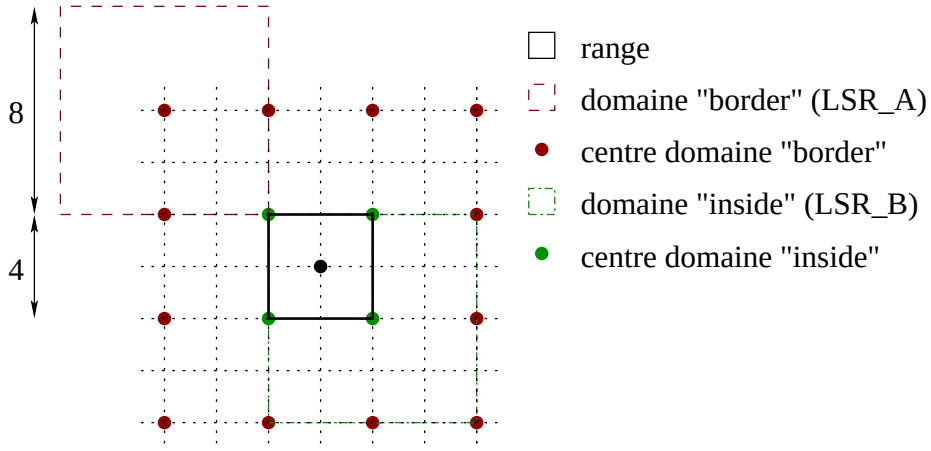


FIGURE 3.5 : Découpage de la LSR en région A et B selon le bit à insérer.

pectivement appelé LSR_A et LSR_B) est utilisé pour rechercher l'antécédent. Afin de renforcer le marquage, chaque bit du filigrane est inséré plusieurs fois dans l'image : si on note r cette redondance, le processus d'insertion requière donc $32*U$ blocs. Les blocs restant sont codés normalement, en recherchant leur antécédent dans LSR. L'image marquée correspond finalement à l'attracteur de l'IFS ainsi construit.

Pour vérifier la présence du tatouage, le générateur aléatoire est à nouveau initialisé à l'aide de la clé secrète k, et les domaines correspondant aux U premiers blocs destination sont déterminés. Selon qu'ils appartiennent à la région LSR_A ou LSR_B, la valeur du bit s'_0 est connue. Les U ranges suivant donnent la valeur du bit suivant, et ainsi de suite jusqu'à reconstruire le tatouage $s' = (s'_0, \ldots, s'_{31})$. Celui-ci est alors comparé à la marque initiale s.

Le tableau 3.2 page suivante détaille les fonctions d'insertion et de détection.

3.2.2 Problèmes

Nous ne détaillons pas ici les problèmes de robustesse liés au partitionnement de l'image. En revanche, nous avons relevé deux problèmes importants dans l'approche présentée. Tout d'abord, lors de l'insertion d'une marque dans une image, celle-ci doit subir le moins de modifications possibles. Or, la définition de la LSR, en particulier pour les ranges contenant un bit de la marque, ne laisse le choix que parmi très peu d'antécédents, ce qui altère d'autant la qualité de l'image résultante.

Par ailleurs, la détection de la marque dans des zones uniformes n'est pas très fiable. En effet, des domaines satisfaisants existent alors à la fois dans les régions LSR_A et LSR_B qui correspondent au bloc destination désiré.

Insertion de la marque

1. Génération d'une marque binaire $s = (s_0, \ldots, s_{31})$
2. Détermination d'une clé secrète k qui initialise un générateur aléatoire
3. Pour insérer le bit s_i
   (a) Sélection aléatoire de U blocs destination à l'aide du générateur
   (b) Pour chacun des U blocs
   - Si $s_i = 0$, rechercher un domaine dans LSR_B
   - Si $s_i = 1$, rechercher un domaine dans LSR_A
4. Pour les ranges qui n'ont pas encore été codé, rechercher leur antécédent dans $LSR = LSR_A \bigcup LSR_B$
5. Génération de l'attracteur de l'IFS qui constitue l'image tatouée

Vérification de la marque

1. Initialisation du générateur aléatoire avec la clé secrète k
2. Pour déterminer le bit s'_i ($i \in \{0, \ldots, 31\}$)
   (a) Sélection aléatoire de U blocs destination $R_j, j \in \{1, \ldots U\}$ à l'aide du générateur
   (b) Pour chacun, recherche de son antécédent D dans LSR
   - si $D \in LSR_B$ alors $s'_{ij} = 0$
   - si $D \in LSR_A$ alors $s'_{ij} = 1$
   (c) s'_i est obtenu par vote selon les s'_{ij}
3. Comparaison de s et s' pour déterminer si l'image est tatouée

TABLEAU 3.2 : Schéma de tatouage par IFS contraints de Puate et Jordan

3.2.2.1 Antécédents trop rares

La mise en correspondance des ranges avec les domaines influence aussi la qualité de l'image obtenue. On recherche, pour un range donné le meilleur antécédent possible dans l'ensemble des domaines. Or, tel que construit ici, l'ensemble des domaines ne contient que peu d'éléments (voire un seul pour les coins de l'image), ce qui conduit à une dégradation de l'image décompressée.

La figure 3.6 illustre ceci dans le cas extrême du coin inférieur droit de l'image. Seul un domaine est valide sur la frontière si le bit à coder vaut 1. Dans le cas où le bit vaut 0, l'intérieur ne contient également qu'un seul domaine (en

vert) puisqu'on sélectionne les domaines de 4 en 4.

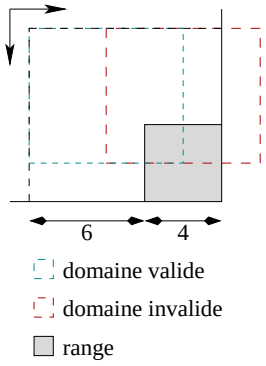


FIGURE 3.6 : Domaine pour un range en coin avec n=4

3.2.2.2 Indécidabilité dans les zones uniformes

Le découpage employé pour rechercher des domaines pose des problèmes dans les zones uniformes de l'image lors de la détection de la marque. En effet, dans une telle région, des domaines identiques existent à la fois dans LSR_A et LSR_B. On recherche le meilleur antécédent dans ces deux ensembles. Ensuite, selon que le meilleur antécédent appartient à un ensemble ou à l'autre, on obtient la valeur du bit. Or, lorsque deux blocs de contenu identique se retrouvent dans les deux ensembles et qu'ils correspondent au meilleur antécédent, on ne parvient pas à déterminer la valeur du bit.

Pour palier cette difficulté, les auteurs utilisent de la redondance. Chaque bit étant inscrit plusieurs fois dans l'image, même si certaines zones ne permettent pas de lever l'ambiguïté, d'autres le feront.

Toutefois, l'image 3.7 est un cas pathologique qui laisse la détection complètement perdue.

FIGURE 3.7: l'uniformité des régions pose des problèmes pour distinguer les ensembles LSR_A et LSR_B

Certaines méthodes de recherche des domaines, comme celle du quadtree, reposent sur l'hypothèse que les régions voisines ont une probabilité plus élevée de correspondre à un range lorsqu'on en compare le niveau de gris moyen. Ce-

pendant, ce critère n'est pas optimal comme le montre l'image 3.3 page 54. Des méthodes fondées sur des dictionnaires permettent d'élargir cette recherche.

D'un point de vue tatouage, pour lever une éventuelle indécidabilité, la méthode dissimule plusieurs fois les 32 bits de la marque. Cette solution suffit amplement pour remédier à ce problème ... mais le coût en est élevé vis-à-vis de la qualité de l'image.

3.2.3 Attaques

Outre la qualité de l'image tatouée, la robustesse de la méthode semble faible. En effet, elle repose sur un partitionnement de l'image. Un outil comme `StirMark` permet de modifier de manière imperceptible ce découpage. Par ailleurs, une autre attaque consiste à appliquer de nouveau un algorithme de compression fractale, mais qui utilise une LSR différente de celle présentée dans l'article. Ainsi, lors de la phase de détection, les correspondances entre les domaines et blocs destination étant bouleversées, la recherche d'antécédent ne conduit plus à la détection de la marque espérée. Cette approche ne permet aucun contrôle sur la «nouvelle marque» ainsi générée.

La réalisation d'une attaque plus précise est néanmoins possible, au prix d'une détérioration supplémentaire de l'image. L'algorithme étant public, un attaquant connaît le partitionnement utilisé par le tatoueur, même s'il ignore les blocs qui contiennent l'information.

Si l'image contient en tout n blocs, dont p servent à dissimuler un bit, on peut alors estimer la difficulté de l'attaque en se demandant quelle est la probabilité de modifier au moins k blocs parmi les p marqués lorsqu'on altère exactement l blocs dans l'image.

Tout d'abord, on estime la probabilité de modifier exactement i blocs marqués. Parmi les p blocs marqués, il existe alors C_i^p combinaisons de i éléments. Puisqu'au total on sélectionne l blocs, il reste encore à en choisir $l-i$ parmi les $n-p$ blocs vierges, soit C_{l-i}^{n-p} combinaisons. Ainsi, il existe, au total, $C_i^p\ C_{l-i}^{n-p}$ manières de sélectionner exactement i blocs marqués en tirant au sort l blocs parmi les n. La probabilité associée est donc :

$$P(\text{exactement } i) = \frac{C_i^p\ C_{l-i}^{n-p}}{C_l^n}$$

Modifier au moins k blocs marqués signifie qu'on en modifie soit k, soit $k+1$ ..., soit p. Ainsi, la probabilité recherchée est donnée par :

$$p(\text{au moins } k) = \frac{1}{C_l^n} \sum_{i=k}^{p} C_i^p\ C_{l-i}^{n-p}$$

Cependant, sauf cas particuliers[4], l'attaquant ne peut pas forger la marque de son choix car il ne sait pas quels sont les emplacements utilisés pour tatouer l'image.

[4]Les marques ne contenant que des bits identiques

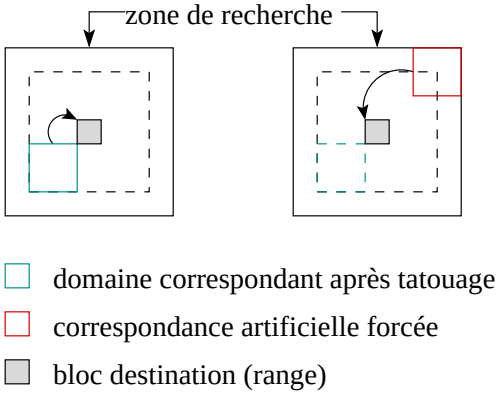


FIGURE 3.8 : Modifier la marque en changeant les correspondances

3.2.4 Conclusion

Malgré ces défauts, cette méthode propose une voie intéressante. Utiliser deux ensembles distincts pour la recherche de domaines reste pertinent, à condition de construire ces ensembles de sorte que leur intersection soit bien vide. Afin d'éviter une recherche trop localisée, on peut diviser une bibliothèque de domaines en deux par rapport à un seuil sur la valeur d'un paramètre donné. Par exemple, considérer la valeur moyenne de luminance des domaines permet de fabriquer deux ensembles disjoints.

Nous avons néanmoins vu en 3.2.2.2 page 59 un cas extrême présentant une ambiguïté. Pour éviter cette situation d'indécidabilité, il faut s'assurer que les deux sous-ensembles utilisés constituent bien une partition de l'ensemble de recherche.

Toutefois, la qualité de l'image obtenue dès qu'une méthode utilise des IFS dépend fortement de la construction du code IFS de l'image initiale. Nous avons vu (cf. sec. 3.1.3) la définition du problème inverse pour les IFS. Pour le résoudre, les algorithmes de compression fractale reposent sur une version simplifiée, où des hypothèses fortes sont émises :

1. les fonctions constituant l'IFS sont localisées dans des domaines et des ranges (ces derniers constituant une partition de l'image) ;
2. les fonctions sont supposées affines.

Pour chaque range, une optimisation donne le meilleur domaine correspondant au sens des moindres carrés. Lorsque la taille des ranges est trop importante, l'image résultante comporte de nombreux défauts (voir fig. 3.9). La qualité obtenue dépend fortement du choix fait pour la partition : des ranges de petite taille dissimuleront les imperfections visuelles, au prix d'une recherche plus coûteuse et d'un taux de compression moins élevé.

Ainsi, la mise au point d'une méthode plus efficace pour résoudre le problème inverse pour les IFS permettrait d'améliorer la qualité de l'approximation réalisée pour transformer un domaine en un range, et par la même, diminuer le

FIGURE 3.9: Lena (image initiale en 512×512) compressée puis décompressée avec une taille de ranges contrainte à 32 pixels à gauche contre 2 ou 4 à droite.

nombre nécessaire de domaines et de ranges.

Une solution, fondée sur les algorithmes évolutionnaires, a déjà été présentée dans le cas de fonctions affines ([Vrs90, LL93]). Nous avons étendu la classe de fonctions utilisée en considérant des fonctions quelconques. Pour y parvenir, nous avons dû mettre au point une nouvelle variante pour un algorithme évolutionnaire afin de l'adapter au problème et d'en améliorer les performances.

3.3 Introduction aux IFS mixtes et polaires

Les méthodes de compression fractales reposent sur l'utilisation d'IFS particuliers où les fonctions sont affines. Dans cette partie, nous considérons d'autres classes d'IFS pour lesquelles nous relâchons cette contrainte. En compression, l'utilisation d'IFS affines nécessite de construire une partition constituée de nombreux ranges et domaines afin d'obtenir une qualité visuelle acceptable. L'usage de d'IFS plus généraux permet d'obtenir une plus grande variété de formes ([LLC+95]).

Nous précisons tout d'abord le terme d'*IFS mixte*, pour lequel les fonctions sont quelconques, ainsi qu'une représentation informatique adaptée aux AE, et à la programmation génétique en particulier. Cependant, ces IFS sont difficilement contrôlables. Nous introduisons ensuite la notion de *contractance locale vis-à-vis d'un point* et une nouvelle classe d'IFS, dits *polaires*, construits de manière à vérifier cette propriété. Enfin, nous concluons avec le problème de l'estimation numérique de la contractance d'une fonction.

3.3.1 IFS Mixtes

Lorsqu'on parle d'IFS dans des applications, les fonctions w_i qui le composent sont le plus souvent supposées affines. Dès lors que les w_i ne sont plus restreintes à cette classe de fonctions mais laissées libres, nous avons choisi de parler d'*IFS mixtes* pour bien rendre clair le fait que nous ne nous restreignons plus les fonctions à être affines.

Outre la facilité calculatoire des IFS affines, leur représentation informatique offre l'avantage d'être simple et de taille fixe comme le montre l'équation 3.2 page 53.

La programmation génétique manipule usuellement des arbres (voir annexe A page 181). Cette structure de données est particulièrement adaptée à la représentation de fonctions : les w_i sont alors construites à partir de constantes prises dans l'intervalle $[0, 1]$, d'un ensemble de variables (x et y) et d'un ensemble de fonctions élémentaires.

Éléments	Ensemble de définition
Constantes	$[0, 1]$
Variables	$\{x, y\}$
Fonctions	$\{+, -, \times, div(x,y) = \frac{x}{0.0001+\lvert y\rvert}, \cos, \sin,$ $root(x) = \sqrt{\lvert x\rvert}, loga(x) = \log(1 + \lvert x\rvert)\}$

TABLEAU 3.3 : Éléments intervenant dans la construction des fonctions

S'il est facile de calculer directement à partir des paramètres le facteur de contractance lorsque les w_i sont affines, ce n'est plus le cas avec les IFS mixtes (voir sec. 3.3.4 page 67). De plus, assez peu de fonctions ainsi construites sont naturellement contractantes. Pour remédier à ces problèmes, nous nous sommes restreints à une classe d'IFS plus facilement contrôlable : les *IFS polaires*. Les fonctions qui les composent sont construites de manière à être *localement contractante vis-à-vis d'un point*.

3.3.2 Contractance locale

Pour obtenir des fonctions affines contractantes à partir de l'équation (3.2), les valeurs propres de la matrice $\begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix}$ doivent être inférieures à 1. Toutefois, pour une fonction quelconque, il n'existe pas de propriété facilement vérifiable calculatoirement. Celle-ci ne peut donc avoir lieu qu'*a posteriori*, en testant l'hypothèse de contractance sur un ensemble de points (voir 3.3.4 page 67).

Le but des IFS polaires est d'offrir un intermédiaire entre contraintes structurelles très fortes (cas des IFS affines) et liberté importante (cas des IFS mixtes).

Cette classe d'IFS est construite à partir de fonctions particulières qui sont toutes *localement contractantes vis-à-vis d'un point*. Nous définissons cette notion avant d'introduire les IFS polaires :

Définition 3.4 *Une fonction $w : F \to F$ d'un espace métrique complet (F, d) dans lui-même est dite* localement contractante relativement au point P *s'il existe un nombre réel positif $s < 1$ tel que :*

$$\forall M \in F, \quad d\big(P, w(M)\big) \leq s.d(P, M) \tag{3.3}$$

Le plus grand réel s tel que la propriété précédente est vérifiée est appelé facteur de contractance local de w vis-à-vis de P.

Étudions les propriétés induites pour ces fonctions.

Proposition 3.2 *Si une fonction w est localement contractante relativement à un point P, alors P est l'unique point fixe de w.*

La conséquence immédiate est que w ne peut être localement contractante que par rapport à un unique point.

Démonstration Soit w une fonction localement contractante vis-à-vis d'un point P.

Montrons tout d'abord que P est un point fixe de w. Si on applique (3.3) au point P, nous avons :

$$d(P, w(P)) \leq s.d(P, P) = 0$$

Donc, $d(P, w(P)) = 0$ et $w(P) = P$ puisque d est une distance.

Montrons maintenant que P est unique. Supposons donc que w soit localement contractante relativement à deux points P_0 et P_1. D'après l'équation (3.3), on a :

$$d(P_0, w(P_1)) = d(P_0, P_1) \leq s.d(P_0, P_1)$$

Et donc $P_0 = P_1$.

On constate donc qu'une fonction localement contractante possède des propriétés comparables à celle d'une fonction contractante. Il est donc naturel de se demander quel(s) lien(s) existe(nt) entre ces propriétés :

Proposition 3.3 *Si une fonction w est contractante, alors elle est localement contractante vis-à-vis de son point fixe.*

La réciproque n'est toutefois pas vraie. On peut construire une fonction localement contractante admettant un unique point fixe, mais qui n'est pas contractante. Un contre-exemple est présenté dans [CLRS00].

3.3.3 IFS polaires

Nous nous intéressons aux IFS dans le cas des images, et nous nous restreignons au domaine défini par le carré $[0, 1] \times [0, 1]$.

Comme nous le détaillons en 3.3.4 page 67, vérifier la contractance d'une fonction non-linéaire est coûteux en temps de calcul. De plus, assez peu de

ces fonctions sont naturellement contractantes. Ainsi, plutôt que de s'appuyer sur ces fonctions, nous proposons une méthode de construction de fonctions localement contractantes respectivement à un point P.

La définition (3.4) de contractance locale vis-à-vis d'un point P s'accorde naturellement avec le système de coordonnées polaires. En effet, dans un tel système, un point M quelconque a pour coordonnées (ρ, θ) par rapport à P, où ρ représente la distance entre ces deux points. L'équation (3.3) de la définition correspond alors uniquement à une contrainte sur la fonction qui contrôle l'évolution de ρ, indépendamment de θ, comme le prouve la propriété suivante :

Proposition 3.4 *Soient (F, d) un espace métrique complet et w une fonction définie de F dans F. Soit, pour tout point M de F, la suite $w^n(M)$ définie par*

$$w \quad : \quad \mathbb{N} \rightarrow \mathbb{R}^2$$
$$n \mapsto w^n(M) = \begin{cases} w^0(M) = M \\ w^n(M) = \begin{pmatrix} \rho(n) \\ \theta(n) \end{pmatrix} \text{ où } \rho(n) = d(P, w^n(M)) \end{cases}$$

La fonction w est localement contractante relativement au point P si et seulement si la suite $\rho(n)$ est strictement décroissante.

Démonstration Si w est localement contractante vis-à-vis d'un point P alors, d'après l'équation (3.3), $\exists s \in [0, 1[$ tel que

$$\forall M \in F, d(P, w^{n+1}(M)) \leq s.d(P, w^n(M))$$

Or, par définition des coordonnées polaires, on a $d(P, w^n(M)) = \rho(n)$. Donc, $\forall n \in \mathbb{N}, \rho(n+1) \leq s.\rho(n)$ et la suite $\rho(n)$ est bien strictement décroissante puisque $s \in [0, 1[$.

Supposons maintenant que la suite $\rho(n)$ soit strictement décroissante, c'est-à-dire que $\forall n \in \mathbb{N}, \rho(n+1) < \rho(n)$.

Cette inégalité est équivalente à

$$\forall M \in F, d(P, w^{n+1}(M)) < d(P, w^n(M))$$

Et donc l'équation (3.3) est vérifiée.

Étant donnée cette contrainte sur $\rho(n)$, nous pouvons exprimer la fonction w de la manière suivante :

$$w : \begin{array}{ll} F \rightarrow & F \\ M \mapsto & w(M) = \begin{pmatrix} \rho f(\rho, \theta) \\ g(\rho, \theta) \end{pmatrix} \end{array}$$

où f et g sont deux fonctions avec pour seule contrainte que $f : \mathbb{R}^2 \rightarrow [0, 1]$ soit toujours inférieure à 1 afin de garantir la contractance locale de w respectivement à un point P.

Le choix de la fonction f résulte d'une comparaison présentée dans le tableau 3.4 : la fonction $\tanh(x)$ est celle qui donne le taux le plus élevé de fonctions contractantes. Ces tests ont été réalisés sur un échantillon de 10000 fonctions générées aléatoirement sur un carré 64×64 à l'aide du «test du carré» (cf. section 3.3.4 page ci-contre).

Fonction	**% de fonctions contractantes**	**Contractance moyenne**
$\frac{1+\lvert\sin(x)\rvert}{2}$	11.47%	0.425621
$\lvert\sin(x)\rvert$	13.3%	0.350461
$\frac{1}{1+\lvert x\rvert}$	14.52%	0.374461
$\frac{\tanh(x*10^{-1})+1}{2}$	22.48%	0.35302
$\frac{\tanh(x*10^{-3})+1}{2}$	38.62%	0.35881
$\frac{\tanh(x*10^{-5})+1}{2}$	40.65%	0.35082

TABLEAU 3.4: Résultats des essais pour le choix de la fonction utilisée dans la composante ρ des IFS polaires

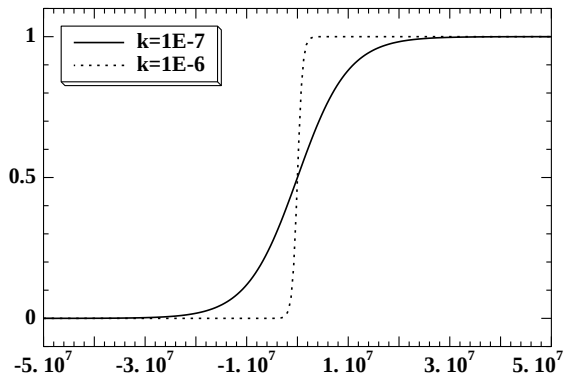


FIGURE 3.10 : La fonction $\frac{\tanh(k.x)+1}{2}$ pour différentes valeurs de k

En principe, toutes les fonctions à valeurs dans $[0,1[$ conviennent. Cependant, les expériences présentées dans le tableau 3.4 nous ont fait choisir la forme suivante :

$$\begin{array}{rccl} w: & F & \to & F \\ & M & \mapsto & w(M) = \begin{pmatrix} \rho\frac{\tanh(k*F(\rho,\theta))+1}{2} \\ g(\rho,\theta) \end{pmatrix} \end{array} \tag{3.4}$$

où $F(\rho,\theta)$ et $G(\rho,\theta)$ sont maintenant des fonctions quelconques à valeurs dans $\mathbb{R}$. La fonction $x \mapsto \frac{\tanh(k.x)+1}{2}$ permet un passage continu de $\mathbb{R}$ à l'intervalle $[0,1]$, comme le montre la figure 3.10.

Enfin, un autre paramètre intéressant des IFS polaires vient du contrôle sur la position des points fixes. Tout comme dans le cas des IFS mixtes et affines, il est possible de translater la fonction sans nuire à la contractance. En revanche, il n'est plus besoin d'estimer la position du point fixe puisqu'elle apparaît explicitement dans l'expression de la fonction. Chacune des fonctions w_i n'est pas localisée spatialement et peut dès lors se situer n'importe où.

3.3.4 Tests de contractance

Nous avons vu que les fonctions localement contractantes relativement à un point ne sont pas systématiquement contractantes. On peut se demander, dans ces conditions, quel est l'intérêt de cette propriété. Nous avons expérimentalement vérifié que la contrainte structurelle simple des fonctions localement contractantes vis-à-vis d'un point permettait d'avoir un espace de recherche contenant plus de fonctions contractantes.

Dans cette partie, nous présentons des expériences réalisées dans le but de trouver quels sont les tests de contractance qui seront utilisés dans notre algorithme évolutionnaire. Il doit être à la fois simple, peu coûteux en temps de calcul et le plus sélectif possible.

Comme nous ne possédons pas de propriété facilement utilisable pour vérifier si une fonction est contractante ou non, nous devons tester numériquement l'équation (3.1) sur un ensemble de points.

Le problème qui se pose est donc de trouver une méthode numérique qui permet de vérifier la contractance d'une fonction et, si elle l'est, d'estimer son facteur de contractance. Travailler sur des images impose une discrétisation du carré $[0,1]\times[0,1]$ et les résultats obtenus dépendent alors de la résolution choisie. Bien évidemment, plus celle-ci est fine, plus le nombre de points est important, ce qui autorise une meilleure estimation ... mais au prix d'un temps de calcul grandement accru.

Nous générons aléatoirement des fonctions non-linéaires à partir des éléments définis dans le tableau 3.3 page 63. Dans le cas des IFS mixtes, elles sont utilisées tel quel. En revanche, pour les IFS polaires, elle sont encapsulées selon les termes de l'équation 3.4 page précédente.

Le principe général des tests est de considérer une paire de points (M, M') et leurs images $(w(M), w(M'))$ pour vérifier que

$$d(w(M), w(M')) < d(M, M')$$

Chacun des quatre tests défini dans le carré $[0,1] \times [0,1]$ discrétisé correspond à une manière différente de sélectionner les paires (on considère des images de taille $n \times m$) :

1. *test complet* :
   un point donné est associé à chaque autre point de l'image pour constituer une paire. On fait cela pour tous les points de l'image. À une résolution donnée, ce test fournit les meilleurs résultats car toutes les paires sont testées. Cependant, sa complexité est trop élevée pour qu'il soit retenu

(cf. 3.5 page ci-contre). Il nous sert donc de valeur de référence pour évaluer les performances des autres tests.

2. *test twist* :
   ce test part de deux points situés dans des coins opposés de l'image, chaque point parcourant symétriquement une moitié de l'image, ligne par ligne jusqu'à atteindre le centre de l'image (cf. fig. 3.11).

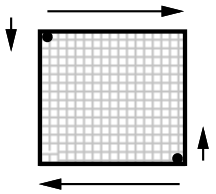


FIGURE 3.11 : Test twist

3. *test du carré* :
   chaque point et son voisin immédiat forment la paire utilisée (cf. fig. 3.12). Toute l'image est ainsi parcourue.

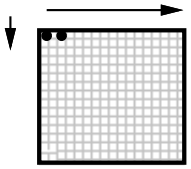


FIGURE 3.12 : Test du carré

4. *test aléatoire* :
   nm paires de points sont sélectionnées aléatoirement, uniformément dans $[0,1]^2$.

La complexité de ces tests change grandement de l'un à l'autre. Dans la perspective d'un AE, il faut en trouver un qui ne prenne pas trop de temps de calcul. Le tableau 3.5 page suivante présente la complexité de chacun d'eux. Comme prévu, les trois autres possédant le même ordre de grandeur.

Les résultats sont présentés dans les tableaux 3.6 et 3.7 page 70. Pour des raisons obscures, le test twist est inefficace[5]. En revanche, le test du carré donne des résultats bien plus proches de ceux obtenus par le test complet. La précision est même meilleure pour les IFS polaires que pour les mixtes.

L'autre résultat intéressant est la confirmation de notre hypothèse : la densité de fonctions contractantes est effectivement plus élevée parmi les fonctions localement contractantes par rapport à un point.

[5]Une explication pourrait être que la contractance a plus de chance de ne pas être vérifiée pour des points proches que pour des points éloignés ou bien, plus prosaïquement, un problème d'arrondi dans les calculs lié à la proximité des points fausserait le test.

	# de w_i calculés	# de comparaisons
complet	nm	$\frac{(nm-1)(nm-2)}{2}$
twist	nm	$\frac{nm}{2}$
carré	nm	nm
aléatoire	$2nm$	nm

TABLEAU 3.5: Complexité des différents tests (pour une image de taille $n \times m$)

n	# of w_i computed	all-pixels	twist	square	random
16	256	10.26	24.46	21.64	14.67
32	1024	10.46	24.16	21.80	13.73
64	4096	9.42	23.96	21.62	13.25
128	16384	-	24.03	22.08	13.39
256	65536	-	23.94	21.19	13.04
512	262144	-	23.91	21.18	-

TABLEAU 3.6: Proportion de fonctions contractantes (en %) obtenues à chaque test pour des IFS mixtes

Ainsi, avec le test du carré, nous disposons d'un outil performant, particulièrement adapté aux IFS polaires, afin de tester nos fonctions dans le cadre d'un algorithme évolutionnaire.

3.4 Applications : génération et problème inverse

La méthode présentée dans [PJ96] est similaire à la génération d'un IFS sous contraintes, tout comme la compression fractale. Cependant, à la différence de cette dernière, une hypothèse supplémentaire intervient : l'ensemble des domaines est lui-même partitionné. L'inconvénient majeur de cet ajout réside dans la dégradation de la qualité de l'image décompressée.

Partant de ce constat, nous nous sommes intéressés au problème de la génération d'IFS sous contraintes selon deux angles différents :

1. *problème contraint géométriquement* : étant donné un ensemble de contraintes, générer un attracteur qui y correspond «au mieux[6]» (voir 3.4.1 page suivante) ;
2. *problème classique* : trouver un IFS dont l'attracteur est donné (voir 3.1.3 page 52 pour une définition précise).

[6]Cette notion sera définie par la suite selon une métrique appropriée au problème

n	# of w_i computed	all-pixels	twist	square	random
16	256	62.48	90.56	64.94	74.29
32	1024	58.82	88.91	60.72	69.28
64	4096	56.12	88.08	57.91	64.30
128	16384	-	87.39	56.44	62.14
256	65536	-	87.24	55.26	60.00
512	262144	-	87.03	53.89	-

TABLEAU 3.7: Proportion de fonctions contractantes (en %) obtenues à chaque test pour des IFS polaires

Lorsque le problème est «simplement» contraint géométriquement, l'attracteur vers lequel doit converger l'IFS n'est pas donné. Seules des contraintes géométriques sur l'attracteur sont imposées, comme une densité de points par exemple.

Les schémas de tatouage par substitution utilisent ce même genre de procédé afin de remplacer des extraits du signal par d'autres, vérifiant une propriété désirée ... mais ils doivent aussi être le plus proche possible du signal original. Ils utilisent donc conjointement les deux problèmes évoqués ci-dessus. Nous avons donc voulu séparer ces problèmes afin de mieux appréhender leur spécificité.

Nous utilisons les IFS précédemment introduits dans le cas de génération sous contraintes géométriques à l'aide d'algorithmes évolutionnaires. Constatant expérimentalement le coût élevé du modèle classique, nous introduisons alors l'approche Parisienne comme alternative, ce qui nous permet de montrer son intérêt dans le cas du problème inverse pour les IFS.

3.4.1 Génération interactive d'IFS

À la différence des AG classiques, nous avons ici décidé d'étudier le comportement de convergence lorsqu'un facteur externe intervient dans l'évolution. Le but de l'AG est de produire des attracteurs qui satisfont certaines contraintes, mais également de présenter des images qui plaisent à l'utilisateur. Ce paramètre, difficilement quantifiable, est fourni interactivement entre chaque génération : l'utilisateur donne une note à l'attracteur, influençant sa fitness mais perturbant d'autant l'évolution.

En effet, dans son processus d'évolution, l'algorithme a pu construire un IFS répondant à toutes les contraintes imposées, sans qu'il soit toutefois au goût de l'utilisateur. La difficulté du problème se situe alors dans l'équilibre à obtenir entre les performances calculées lors de l'évolution et les choix de l'utilisateur, ceux-ci pouvant aller à l'encontre de l'évolution naturelle de l'algorithme.

3.4.1.1 Codage et paramètres de l'algorithme

Le codage des fonctions suit celui présenté dans le tableau 3.3 page 63. Nous avons utilisé aussi bien des IFS mixtes et polaires. Dans cette approche, un individu est composé d'un ensemble de fonctions que l'évolution doit conduire à devenir un IFS.

La fonction de fitness se décompose en deux parties :

1. la *fitness interne* : elle évalue les individus selon des critères mathématiques (facteur de contractance, localisation du point fixe...). Si toutes les fonctions sont contractantes et possèdent un point fixe dans le carré unité, nous calculons ensuite son attracteur pour le comparer à celui décrit par les contraintes de l'utilisateur.

   Dans un premier temps, on cherche donc à conserver seulement les fonctions qui :

   (a) possèdent un facteur de contractance k dans $[0,1[$:

   $$fit_{fc(w_i)}(k) = \begin{cases} \frac{1}{1+k} & \text{si } k \geq 1 \\ 0.5 & \text{sinon} \end{cases}$$

   La fitness de l'individu $fit_{fc}(\Omega)$ concernant le facteur de contractance est alors donnée par le facteur de contractance moyen des fonctions qui le composent.

   (b) ont un point fixe P dans $[0,1]^2$ (on note C le centre de ce carré) :

   $$fit_{pf(w_i)}(P) = \begin{cases} \frac{1}{0.5+d_2(P,C)} & \text{si } P \notin [0,1]^2 \\ 1 & \text{sinon} \end{cases}$$

   La fitness de l'individu $fit_{pf}(\Omega)$ concernant la position des points fixes est alors donnée par le rapport entre le nombre de points fixes dans la zone désirée et le nombre de fonctions qui composent l'IFS[7].

   Une fois que l'ensemble des fonctions de notre individu sont toutes considérées comme contractantes et que leur point fixe se situe dans le carré $[0,1]^2$, il est alors possible d'estimer l'attracteur de cet IFS. On en mesure la densité de points $d(\Omega)$ par rapport à la résolution de l'image (d_u correspond à la densité souhaitée par l'utilisateur et σ_u sa variance) :

   $$fit_d(d(\Omega)) = \min(1, \frac{\exp\left(\frac{(d(\Omega)-d_u)^2}{\sigma_u}\right)}{\exp\left(-\sigma_u\right)})$$

   La fitness interne est alors obtenue par une moyenne pondérée de ces trois composantes :

   $$fit_{int} = \alpha(fit_{fc}(\Omega) + fit_{pf}(\Omega)) + (1-\alpha)fit_d(d(\Omega))$$

[7] Le nombre de fonctions présentes dans l'IFS entre en compte ici afin de ne pas défavoriser les IFS en contenant beaucoup, et pour lesquels il est plus difficile d'avoir toutes les fonctions contractantes et convergentes dans $[0,1]$.

2. une *fitness externe* attribuée par l'utilisateur, à qui sont présentées les six meilleures images de la population, puis qui les note selon ses propres goûts. Trois valeurs sont possibles : laid, indifférent, beau.

La fitness globale d'un attracteur est obtenue en sommant ces deux fitness. Les autres paramètres sont donnés dans le tableau 3.8.

fitness interne	$\alpha = 0.2$
Probabilités de mutation	
constante → constante	0.15 selon $\mathcal{N}(.,0.2)$
variable → constante	0.02, selon $\mathcal{U}[-1,1]$
constante → variable	0.06
variable → variable	0.08
fonction → fonction (même arité)	0.08
points fixes (polaire uniquement)	0.03 selon une loi uniforme sur un disque centré sur le point fixe et de rayon 0.2
Probabilités de croisement	
p_{cross}	0.95 pour les arbres (mixtes et polaires) et les points fixes (polaires)
Sharing	
σ (Goldberg)	0.4
Remplacement de population	
pourcentage	50% avec superposition des populations

TABLEAU 3.8 : Paramètres de l'AE pour la génération interactive d'IFS

Signalons que ces travaux ont été poursuivis au sein du projet FRACTALES par Jonathan Chapuis qui a construit un logiciel utilisant à la fois des IFS affines, polaires et mixtes. Il a également ajouté des contraintes, comme la dimension de boîtes de l'image et une gestion plus dynamique de la palette de couleurs.

3.4.1.2 Résultats

Pour présenter rapidement des formes à l'utilisateur, nous initialisons la population à partir de :

1. fonctions générées aléatoirement ;
2. fonctions issues d'un fichier de sauvegarde et déjà reconnues comme contractantes.

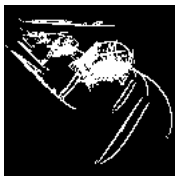
Cependant, lorsqu'un IFS contractant apparaît dans la population, celle-ci tend vers cet individu. La population devient alors très uniforme malgré le sharing. En effet, obtenir un IFS mixte contractant est assez difficile car il faut que

toutes les fonctions qui le composent soient elles-mêmes contractantes. Or, assez peu de fonctions vérifient naturellement cette propriété. Les autres individus de la population convergent alors vers celui qui est contractant.

Les figures 3.13 et 3.14 donnent des exemples de résultats obtenus pour des IFS mixtes, dans lesquels la fitness utilisateur n'intervient pas.

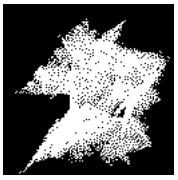
densité : 0.24
10 générations
6 fonctions

densité : 0.2
9 générations
7 fonctions

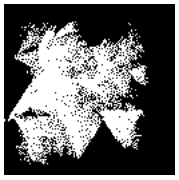


densité : 0.24
10 générations
6 fonctions

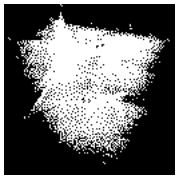
FIGURE 3.13: PG classique pour IFS mixtes : attracteur dans une image 128×128. L'évolution de la population (taille : 20) cesse dès que la densité dépasse 0.2.



densité : 0.41
53 générations
12 fonctions



densité : 0.41
54 générations
10 fonctions



densité : 0.41
91 générations
19 fonctions

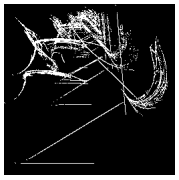
FIGURE 3.14: PG classique pour IFS mixtes : attracteur dans une image 128×128. L'évolution de la population (taille : 30) cesse dès que la densité dépasse 0.4.

L'apport des IFS polaires est très sensible pour résoudre ce problème puisque, proportionnellement, les fonctions contractantes y sont plus nombreuses que parmi les IFS mixtes. La diversité est ainsi mieux préservée. En effet, la population ne converge pas vers le premier individu contractant qui apparaît car il n'est pas unique.

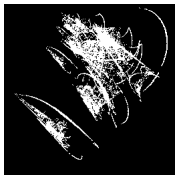
Cependant, qu'il s'agisse d'IFS mixtes ou polaires, la population contient encore de nombreux individus qui, à des degrés divers, ne nous conviennent pas :

- l'IFS comprend une fonction non contractante ;
- un des points fixes de l'IFS se situe en dehors du carré unité (ce qui est bien moins gênant, et donc moins pénalisé au niveau de la fitness).

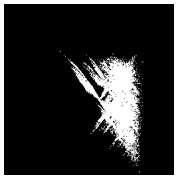
Même si un individu comporte de nombreuses fonctions satisfaisantes, une seule suffit à le faire considérer comme «mauvais», et le condamne donc vis-à-vis de



densité : 0.11
8 générations
12 fonctions



densité : 0.12
10 générations
12 fonctions



densité : 0.1
48 générations
8 fonctions

FIGURE 3.15: PG classique pour IFS polaires : attracteur dans une image 256×256. L'évolution de la population (taille : 30) cesse dès que la densité dépasse 0.1.

l'évolution.

Afin d'éviter de perdre ce patrimoine génétique, nous avons développé une variante pour les algorithmes évolutionnaires : l'approche Parisienne.

3.4.2 Approche Parisienne des algorithmes évolutionnaires

Lorsqu'il est utilisé naturellement, un AE produit beaucoup de déchets pour parvenir à une solution : le moindre défaut chez un individu le condamne, à plus ou moins court terme, à disparaître de la population. Son patrimoine génétique est alors intégralement perdu, même s'il contenait des «morceaux» intéressants.

L'exemple des systèmes de classeurs concerne des problèmes analogues. Ces systèmes proposent de modéliser le comportement par un ensemble de règles du type : si situation alors action. L'approche Pittsburgh fait évoluer une population d'ensemble de règles ([DeJ75, Smi83]). à l'inverse, l'approche Michigan ([Hol75, Hol86, WG89]) assimile un individu à une règle et fabrique sa base à partir des meilleures solutions de la population.

L'approche Parisienne applique le même principe aux algorithmes évolutionnaires. Plutôt que de laisser un individu représenter une solution, il n'en constitue qu'un morceau. La solution au problème est alors construite à partir d'un sous-ensemble de la population. Chaque individu possède sa propre fitness, dite *locale*. La qualité de la solution est mesurée grâce à sa *fitness globale*.

On distingue alors différents modèles, selon le type de fitness qu'il est possible de calculer :

- chaque individu possède sa propre fitness et on espère que le comportement global émergeant sera proche de celui attendu pour que la population résolve le problème ;
- seule la fitness globale est calculable, auquel cas la fitness locale n'est quantifiable qu'en étudiant ce que la présence ou l'absence de l'individu apporte ou enlève à la fitness globale (on parle alors de *fitness marginale*).

Un autre intérêt de cette approche vient de la taille réduite du génome de la population. En effet, comme le «gaspillage» est bien moindre, les individus

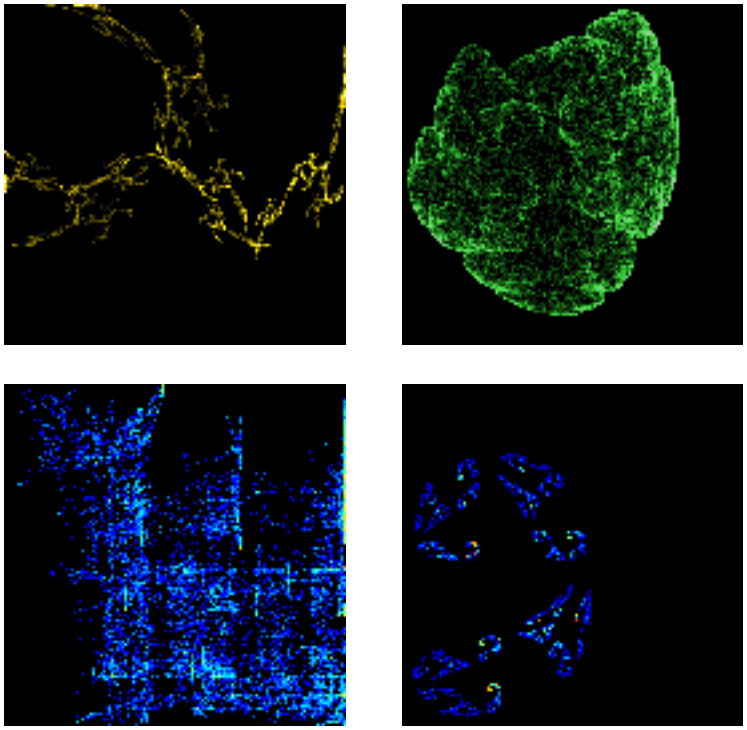


FIGURE 3.16 : Exemples d'IFS mixtes

contribuent tous génétiquement à l'élaboration de la solution. C'est pourquoi, comme nous le verrons par la suite, il est capital de préserver la diversité génétique à l'aide d'une technique de *sharing*, même si celui-ci ne sert pas uniquement dans cette optique.

3.4.3 Approche Parisienne et problème inverse pour les IFS

L'approche Parisienne n'est pas adaptée à tous les problèmes. Cependant, les IFS constituent un cadre adéquat, et en particulier le problème inverse.

3.4.3.1 Codage

Dans l'approche classique présentée figure A.1 page 182, lorsqu'un individu code un IFS, il est nécessaire que chaque fonction qui le compose soit contractante et possède un point fixe dans le carré $[0,1]^2$, sans quoi il obtient un mauvais score. Pour éviter cette perte, un individu de la population correspond maintenant non plus une solution (un IFS contractant) mais à un «bout de solution» (une fonction), la solution étant constituée à partir d'un sous-ensemble issu de la population.

Chaque individu représente donc une fonction (affine, mixte, ou polaire). On espère ainsi éliminer rapidement les fonctions non contractantes sans pour

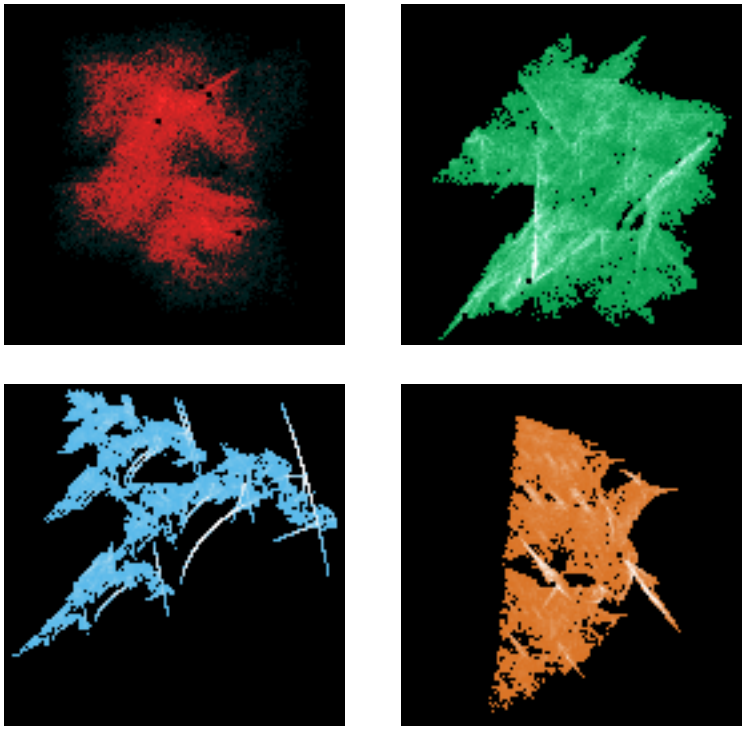


FIGURE 3.17 : Exemples d'IFS polaires

autant pénaliser la diversité de la population. à l'issue d'une génération, des individus sont sélectionnés dans la population pour construire un IFS contractant.

Dans le cas des polaires, nous avons vu que nous avons un accès direct à la position du point fixe. Celle-ci fait partie du génome pour ces fonctions, ce qui nous a permis de construire des opérateurs spécifiques afin d'assurer que les points fixes restent à l'intérieur de l'attracteur. Les fonctions sont quant à elles codées selon l'expression donnée dans l'équation :

$$\begin{array}{rlll} w: & F \rightarrow & F & \\ & M \mapsto & w(M) = & \begin{pmatrix} \rho^{\frac{\tanh(k*f(\rho,\theta))+1}{2}} \\ g(\rho,\theta) \end{pmatrix} \end{array}$$

où les fonctions f et g sont représentées sous forme d'arbres.

3.4.3.2 Fitness locale

Il nous faut maintenant déterminer une mesure pertinente pour la contribution d'un individu à l'attracteur. Soit A l'attracteur dont on cherche à déterminer le code IFS, alors $A = \cup_{i=1...N} w_i(A)$. De cette formule, nous pouvons déduire comment prendre en compte l'apport d'une fonction w_i relativement à l'attracteur A afin de mesurer la fitness locale de cette fonction :

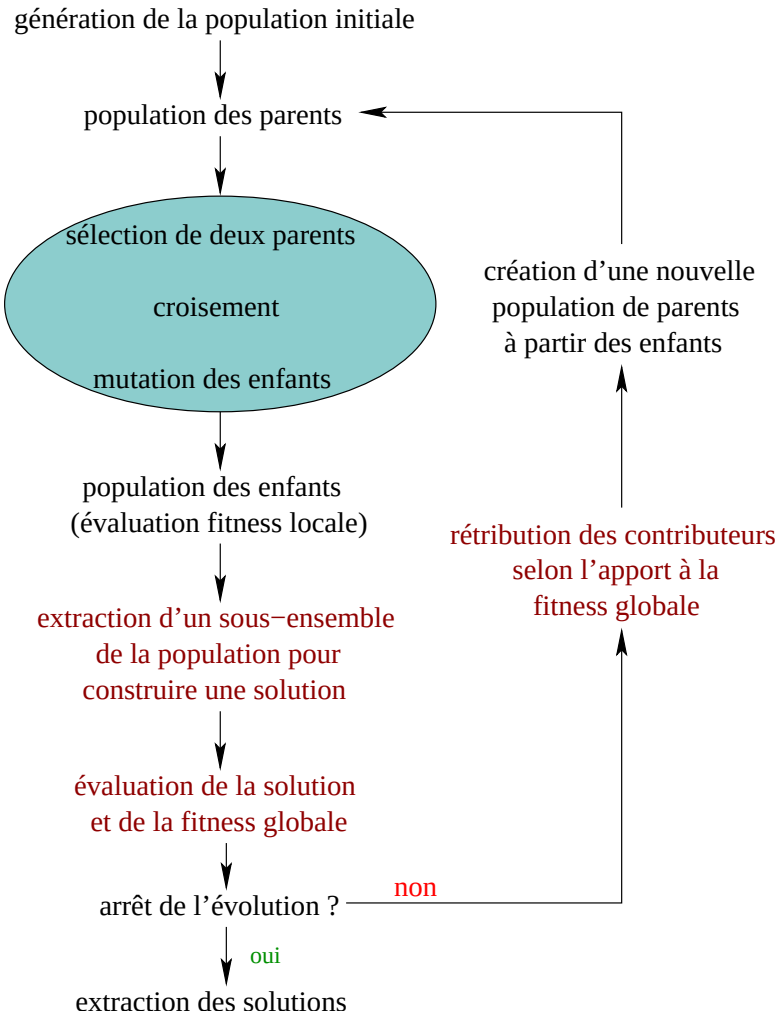


FIGURE 3.18 : Approche Parisienne pour un AE

1. l'image de A par w_i doit se trouver dans A ($w_i(A) \subset A$) : ceci a pour but «d'encourager» les fonctions qui tombent à l'intérieur de A.
   Soit $\#[A]$ le nombre de pixels de $A \subset F$. On définit la fonction $\mathcal{F}_1(w_i)$:

$$\mathcal{F}_1(w_i) = \frac{\#[w_i(A) \bigcap A]}{\#[w_i(A) \bigcap A] + \#[w_i(A) \backslash A]} \tag{3.5}$$

   $\mathcal{F}_1(w_i)$ est maximale (et vaut alors 1) si $w_i(A) \subset A$.

2. l'image de A par w_i doit maximiser l'aire occupée dans A : toutes les fonctions dont l'image est dans A ne sont pas égales, et l'AE favorise celles qui touchent le plus grand nombre de pixels. Ainsi, des fonctions qui ne comptabilisent que peu de points, mais sont précises, sont pénalisées.

Soit la fonction $\mathcal{F}_2$ qui estime cette maximisation :

$$\mathcal{F}_2(w_i) = \frac{\#[w_i(A) \bigcap A]}{\#[A]} \tag{3.6}$$

$\mathcal{F}_2(w_i)$ est maximale (et vaut alors 1) $A \subset w_i(A)$.

Cette fitness repose sur l'interprétation du théorème du collage (3.2) : l'algorithme cherche à construire un ensemble de w_i tels que $A = \bigcup w_i(A)$. Néanmoins, parmi toutes ces unions, nous cherchons à favoriser celles qui comportent le moins de fonctions possible, afin d'obtenir un collage «économique». Il s'agit donc de minimiser les recouvrements $w_i(A) \bigcap w_j(A), \forall (i,j)$, ce qui est fait à l'aide de la fonction de sharing.

Plusieurs actions sont effectuées lors de l'évaluation d'une fonction w_i :

- test de contractance pour déterminer le facteur de contractance et la position du point fixe de la fonction ;
- calcul de $w_i(A)$;

Si la fonction n'est pas contractante, les deux fitness $\mathcal{F}_1$ et $\mathcal{F}_2$ sont mises à 0.

La fitness locale est obtenue à partir de $\mathcal{F}_1$ et $\mathcal{F}_2$, mais également à l'aide de l'estimation du facteur de contractance k_i de la fonction w_i. En effet, $\mathcal{F}_2$ pousse les w_i à remplir l'attracteur, et favorise donc la solution triviale $w_i = Id$. Nous utilisons donc :

$$\mathcal{F}_{loc}(w_i) = \mathcal{F}_1(w_i) + \mathcal{F}_2(w_i) + (1 - s_i) \tag{3.7}$$

Le terme $1 - s_i$ favorise alors les fonctions avec un petit facteur de contractance.

3.4.3.3 Sharing et construction de la solution

Nous avons utilisé le *dynamic niching* [MS95] présenté en A.4 page 186 dans le but de rassembler dans une même niche des individus qui contribuent de manière analogue à la reconstruction de l'attracteur. Il est donc nécessaire de disposer d'un moyen pour estimer la distance entre deux fonctions, et être ainsi capable de les comparer.

Une première approche serait de considérer une image à une résolution grossière (32×32 ou 64×64) et d'en calculer la transformée par w_i et w_j. Ensuite, une distance classique sur les images nous donnerait une estimation de la ressemblance de nos fonctions. Toutefois, le calcul de ces images est coûteux. De plus, nous n'avons pas besoin ici d'une telle précision.

Plutôt que d'utiliser complètement le carré $[0,1]^2$ à une résolution donnée, nous l'échantillonons régulièrement et comparons l'image de ces points. Nous avons utilisé les points situés de 0.1 en 0.1 dans le carré, ce qui nous fournit un quadrillage de celui-ci. Notre but est d'obtenir un aperçu très grossier de l'image du carré par la fonction. Pour chaque point du quadrillage, on calcule son image par les fonctions w_i et w_j puis on les compare au sens de la distance Euclidienne :

$$d(w_i, w_j) = \sum_{k=1}^{N} d_2(w_i(P_k), w_j(P_k))$$

Avec cette distance, il est maintenant possible d'identifier des niches qui rassemblent les individus, comme expliqué dans le tableau A.1 page 187.

L'IFS est construit en prenant la tête de chaque niche. On espère ainsi récupérer des individus qui ne recouvriront pas tous les mêmes zones de l'attracteur.

Par ailleurs, cette technique permet de favoriser la diversité génétique. En effet, tous les individus d'une niche, même s'ils ont un patrimoine génétique différent, sont très proches dans leur contribution à la solution. Pour éviter qu'une niche ne devienne dominante au point d'écraser les autres, la fitness locale de chaque individu qui la compose, à l'exception de la tête, est divisée par le nombre d'individus présents dans la niche, comme nous l'avons présenté en A.4

3.4.3.4 Fitness globale et individus

Nous disposons maintenant d'un IFS contractant dont nous pouvons calculer l'attracteur afin d'en mesurer la correspondance à la cible.

Pour estimer l'attracteur, une *patience* accélère le toss-coin (cf. 3.1.2 page 51) en ajustant la durée (*i.e.* le nombre d'itérations) pendant laquelle on est prêt à attendre avant d'atteindre un nouveau pixel de l'image. Ainsi, une fois ce délai écoulé, si nous n'avons atteint aucun nouveau pixel, nous considérons que notre estimation ne peut s'améliorer et l'algorithme s'arrête. En revanche, la patience est remise à zéro dès qu'on découvre un nouveau point.

Ainsi muni de notre IFS Ω et de son attracteur A_Ω, nous pouvons le comparer à la cible A. Un premier critère pertinent sur la qualité de notre attracteur est donné par la proportion de points dans la cible et à l'extérieur :

$$\begin{aligned} In_\Omega &= \frac{\#[A_\Omega \bigcap A]}{\#[A]} && \text{proportion de points de } A_\Omega \text{ à l'intérieur de } A \\ Out_\Omega &= \#[A_\Omega \backslash A] && \text{proportion de points de } A_\Omega \text{ à l'extérieur de } A \end{aligned}$$

À une génération n donnée, la fitness globale considère uniquement les changements intervenues depuis la génération précédente. Afin de minimiser le nombre de fonctions qui interviennent dans l'IFS, on ajoute un terme qui prend en compte ce nombre. Comme la fitness globale de deux générations successives est comparée, il est possible de savoir si l'ajout ou le retrait de fonctions améliore ou dégrade l'attracteur. On obtient alors l'expression suivante :

$$\begin{aligned} F_{glob}(n) &= [In_{\Omega(n)} - In_{\Omega(n-1)}] - [Out_{\Omega(n)} - Out_{\Omega(n-1)}] \\ &\quad +\alpha[\text{Nb\_functions}(\Omega(n)) - \text{Nb\_functions}(\Omega(n-1))] \end{aligned}$$

La fitness globale est ajoutée de différentes manières selon le rôle tenu par l'individu dans l'élaboration de la solution :

- si w_i apparaît à cette génération dans l'IFS $\Omega(n)$ (*i.e.* il ne participait pas à $\Omega(n-1)$), alors :

$$Fitness(w_i) = F_{loc}(w_i) + F_{glob}(n)$$

– si w_i était déjà présent dans l'IFS $\Omega(n-1)$, alors :

$$Fitness(w_i) = F_{loc}(w_i) + \frac{F_{glob}(n) + F_{glob}(n-1)}{2} \times \frac{1}{[age(w_i)]^2}$$

où $age(w_i)$ représente le nombre de générations pendant lequel w_i appartient à l'IFS.

– si w_i vient juste de quitter l'IFS, alors :

$$Fitness(w_i) = F_{loc}(w_i) - F_{glob}(n)$$

– si w_i n'appartient pas à l'IFS, alors :

$$Fitness(w_i) = F_{loc}(w_i) + \frac{F_{glob}(n-1)}{2}$$

Ainsi, la fitness globale est distribuée sur tous les individus de la population en considérant son passé et ses relations avec l'IFS actuel. De cette manière si un excellent individu quitte l'IFS à la génération n, et que sans lui, l'attracteur $\Omega(n)$ est très mauvais, le terme F_{glob} étant négatif, il voit sa fitness augmenter d'autant. À l'inverse, s'il n'apporte pas grand chose à l'IFS (ou qu'il l'empêche de croître), F_{glob} jouera le rôle d'une pénalité, compromettant les chances de l'individu de revenir dans l'IFS.

Cette fitness globale fournit également le critère d'arrêt de l'algorithme. Il se termine lorsque F_{glob} atteint un seuil donné, qui correspond à un pourcentage de remplissage avec un taux minimal de points à l'extérieur de la cible.

3.4.3.5 Tous les pixels ne sont pas égaux !

Malgré le sharing, certaines régions de la cible sont rarement explorées par les fonctions. Pour encourager celles-ci à s'y rendre, nous avons alloué une pondération aux pixels. Lors du calcul de l'attracteur, nous connaissons, grâce au toss-coin, le nombre de fonctions qui «passent» sur un pixel. La pondération est proportionnelle à cette quantité. Si on la note x, alors le pixel p a pour poids :

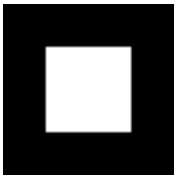
$$v(p) = \begin{cases} 1 & \text{si } x = 0 \text{ ou } x = 1 \\ \frac{1}{x} & \text{sinon} \end{cases}$$

Les équations de fitness locale 3.5 page 77 et 3.6 page 78 considèrent tous les pixels de manière identiques. Avec cette approche, elles ne se contentent plus de dénombrer les pixels présents ou non dans la cible, mais tiennent aussi compte de leur valeur :

$$\#[P] = \sum_{p \in P} v(p) \quad \text{où P est une partie de l'image.}$$

3.4.3.6 Résultats

Résultats pour l'approche Parisienne et les IFS polaires Les figures 3.19, 3.20, 3.21 et 3.22 présentent les résultats obtenus à l'aide de l'approche Parisienne pour le problème inverse pour les IFS. Les paramètres de l'algorithme sont donnés dans le tableau 3.9 page 85



Carré 64×64

Résultat

49 générations
14 fonctions
Population de 60 individus
pixels à l'intérieur : 85% (gris clair)
pixels à l'extérieur : 0% (gris foncé)

FIGURE 3.19 : Approche Parisienne pour un IFS polaire : carré

Trèfle 64×64

Résultat

60 générations
13 fonctions
Population de 100 individus
pixels à l'intérieur : 70% (gris clair)
pixels à l'extérieur : 2.2% (gris foncé)

FIGURE 3.20 : Approche Parisienne pour un IFS polaire : trèfle

Comparaison des résultats La comparaison entre les différents approches étudiées n'est pas très simple. En effet, deux paramètres importants changent : la nature des fonctions et celle de l'AE. L'approche Parisienne modifie énormément le comportement évolutionnaire par rapport à un AE classique. Les performances des ces algorithmes sont habituellement comparées en terme de nombre d'évaluation réalisée, mais ce critère ne représente pas la même chose avec notre approche. Néanmoins, on constate que la modélisation par IFS polaires fournit un espace de recherche certes restreint par rapport aux IFS mixtes, mais également mieux adapté à l'algorithme d'optimisation.

Le critère de comparaison le plus raisonnable est le nombre d'évaluation des arbres codant les fonctions. Pour l'approche Parisienne, on en dénombre environ[8] $(\textit{Taille de la Population} \times \frac{\textit{Nombre de Générations}}{2} + \textit{Nombre d'individus sélectionnés pour l'évaluation globale})$. Pour l'approche classique, on effectue à

[8]Le «/2» provient de ce que seulement 50% de la population est remplacée.

Dauphin 80×80

Résultat

400 générations
13 fonctions
Population de 100 individus
pixels à l'intérieur : 60% (gris clair)
pixels à l'extérieur : 0.36% (gris foncé)

FIGURE 3.21 : Approche Parisienne pour un IFS polaire : dauphin

Bague 128×128

Résultat

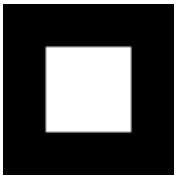
87 générations
31 fonctions
Population de 200 individus
pixels à l'intérieur : 84.52% (gris clair)
pixels à l'extérieur : 5.03% (gris foncé)

FIGURE 3.22 : Approche Parisienne pour un IFS polaire : bague

peu près *(Taille de la Population* $\times$ *Nombre de Générations)* calculs de fonctions. L'évaluation d'une fonction, dans les deux cas, prend approximativement le même nombre d'opérations, c'est-à-dire de l'ordre de grandeur de l'image.

La figure 3.23 présentent les résultats obtenus par un algorithme génétique avec des IFS affines codés par 4 fonctions (soit 24 paramètres réels). La population de 20 individus conduit à ce résultat après 10000 générations, ce qui donne 200000 évaluations. La figure 3.24 retrace l'évolution de la meilleure solution lorsque l'algorithme tourne sur une approximation de plus en plus précise de la cible. À chaque nouvelle génération, les meilleurs individus de la populations sont placés dans la nouvelle population initiale et la cible change pour s'approcher de plus en plus de celle désirée. Cette stratégie nécessite moins de générations que la précédente pour parvenir à des résultats similaires (environ 45000 évaluations).

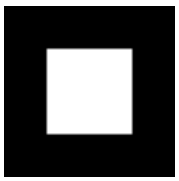
La figure 3.25 donne les résultats pour la Programmation Génétique classique utilisée dans le cadre d'IFS mixtes (tirés de [LLC+95]). Ils ont nécessité environ




Carré 64 × 64 Résultat

10000 générations
4 fonctions
Population de 20 individus
pixels à l'intérieur : 88.4% (gris clair)
pixels à l'extérieur : 17% (gris foncé)

FIGURE 3.23 : AG classique pour IFS affines : carré




Cible finale : carré 64 × 64	Itération 1 : 500 gén. 5 indiv. dedans : 31% dehors : 9%	Itération 2 : 1500 gén. 10 indiv. dedans : 79% dehors : 10%	Itération 2 : 2500 gén. 15 indiv. dedans : 88% dehors : 7%

FIGURE 3.24: AG classique et codage de l'individu de taille variable pour IFS affines avec schéma itératif : carré

45000 évaluations (30 individus pendant 1500 générations).

Enfin, pour terminer cette comparaison, l'approche Parisienne sur des IFS polaires ne demande que 1519 évaluations pour parvenir au résultat de la figure 3.19 page 81 : 60 individus $\times \frac{49 \text{ générations}}{2} + 49$ évaluations globales.

3.5 Conclusion

Si les résultats obtenus sont excellents d'un point de vue évolutionnaire, ils n'en restent pas moins insuffisants pour être utilisés en tatouage. Les contraintes imposées par l'algorithme de P&J restent trop fortes par rapport à celles définies dans ces problèmes.

Toutefois, nous avons introduit deux nouveaux concepts dans le cadre du problème inverse pour les IFS :

– par rapport aux fractales, les IFS polaires constituent un modèle intéressant qui facilite la manipulation des IFS non-linéaires. De plus, la pro-

Carré 64×64 gen. 10 gen. 100 gen. 300 gen. 1500

1500 générations
Population de 30 individus

FIGURE 3.25 : GP classique pour IFS mixtes : carré

portion élevée de fonctions contractantes parmi l'ensemble des fonctions localement contractantes vis-à-vis d'un point combinée avec la simplicité de la transformation des deux arbres[9] en une fonction localement contractante respectivement à un point améliorent grandement les performances de l'algorithme de recherche ;

- du côté évolutionnaire, l'approche Parisienne fournit un environnement parfaitement adapté : l'évolution se produit sur des fonctions simples, puis un sous-ensemble de la population est extrait afin de construire la solution au problème. Cette variante nécessite une grande précision dans l'élaboration des fitness locales et globales, ainsi que dans leur combinaison utilisée lors de l'étape de sélection.

[9]Qui représentent une fonction quelconque

Opérateurs binaires	$+$, -, *, $\div$	
Opérateurs unaires	**neg, 1/., cos, sin, th^{-1}, psqrt, plog**	
Taille de population	voir les résultats section 3.4.3.6	
Opérateurs	**Probabilités de mutation**	
	constante $\rightarrow$ constante	0.15 selon une loi Gaussienne de variance 0.1
	variable $\rightarrow$ constante	0.05 choisit aléatoirement dans $[-1, 1]$
	constante $\rightarrow$ variable	0.08
	variable $\rightarrow$ variable	0.08
	fonction $\rightarrow$ fonction (même arité)	0.08
	points fixes :	0.1 diminuant linéairement au cours des générations.
	Probabilités de croisement	
	PCROSS	0.95 pour les arbres pas de croisement pour les points fixes
Sélection	**Ranking**	
	pression sélective	1.35
Remplacement	**Stratégie de remplacement**	
	% de remplacement	50% population superposée
Critère d'arrêt	**Relativement à F_{glob}**	
		voir section 3.4.3.4
Codage	**description des IFS polaires**	
		voir section 3.3.3
Divers	**Sharing dynamique**	
	σ	0.05
	nb. max de clusters	0.5 of POP_SIZE

TABLEAU 3.9: Paramètres pour l'approche Parisienne appliquée au problème inverse pour les IFS

Chapitre 4

Analyse multifractale

Dans ce chapitre, nous essayons d'envisager comment l'analyse multifractale pourrait être employée dans un schéma de tatouage. En particulier, les *spectres mutuels* permettent de comparer des mesures à la fois localement et globalement. Ainsi, nous commençons par rappeler les principales notions liées à l'analyse multifractale. Puis, à partir d'un schéma de tatouage théorique, nous présentons les premières expériences réalisées afin de tester les possibilités offertes par notre approche.

4.1 Introduction à la théorie multifractale

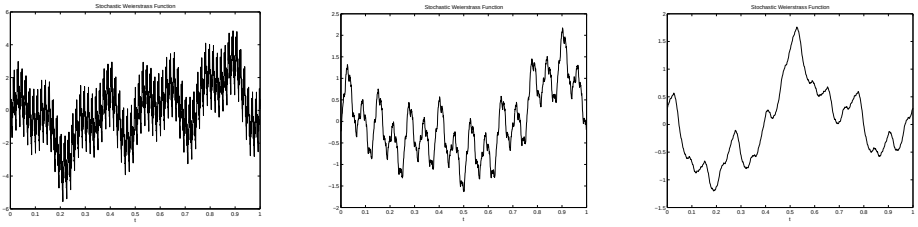
La particularité des spectres multifractals est d'analyser et de répertorier l'irrégularité locale des points d'une image vis-à-vis d'une mesure de référence. Suite à un certain nombre d'études menées au projet FRACTALES ([CL96], [CLT98], [Can98]), nous savons qu'il est possible de modifier l'irrégularité locale de certains ensembles de points afin de changer l'apparence des spectres multifractals associés à l'image.

Dans cette partie, nous introduisons tout d'abord la notion de régularité à l'aide de l'exposant de Hölder ponctuel. Nous regardons ensuite comment mesurer cette régularité dans le cas d'une image, puis nous présentons enfin le spectre de Hausdorff.

4.1.1 Exposant de Hölder ponctuel

Il existe de nombreuses façons de mesurer l'irrégularité locale d'un signal. L'idée est ici de définir un exposant α tel qu'une fonction f se «comporte» comme $|x - x_0|^\alpha$ au voisinage de x_0. C'est une sorte de généralisation de la dérivée ; quand f est C^1, on sait que $|f(x) - f(x_0)|$ est, au premier ordre, proportionnel à $|x - x_0|$. Quand f n'est pas dérivable, on cherche si on peut écrire $|f(x) - f(x_0)| \sim c|x - x_0|^\alpha$. Plus généralement, on regarde si il existe c_1 et c_2, $0 < c_1 \leq c_2$ tels que :

$$\forall x, c_1|x - x_0|^\alpha \leq |f(x) - f(x_0)| \leq c_2|x - x_0|^\alpha$$



$\alpha = 0.1$ $\alpha = 0.5$ $\alpha = 0.9$

FIGURE 4.1 : Courbes de différentes régularités

La figure 4.1 montre des signaux de différentes régularités.

Dans le cas où f est dérivable, mais où une dérivée de f d'ordre supérieur n'existe pas, il faut retrancher à f son polynôme de Taylor pour voir apparaître la singularité et se retrouver dans la situation précédente.

L'exposant ponctuel de Hölder de f, α, en x_0 est alors défini par :

- $\forall \gamma < \alpha, \lim_{h \to 0} \frac{|f(x_0+h)-P(h)|}{|h|^\gamma} = 0,$
- si $\alpha < +\infty, \forall \gamma > \alpha, \limsup_{h \to 0} \frac{|f(x_0+h)-P(h)|}{|h|^\gamma} = +\infty$

où P est un polynôme de degré inférieur ou égal à la partie entière de α. On dit alors que $f \in C^\alpha_{x_0}$.

Enfin, une définition équivalente est, quand $0 < \alpha < 1$:

$$\exists c, \rho_0 > 0, \forall \rho < \rho_0 \quad \sup_{x,y \in B(x_0,\rho)} |f(x) - f(y)| \leq c\ \rho^\alpha$$

L'estimation de $\alpha(x)$ se fait usuellement par régression linéaire sur le nuage de points de coordonnées $(\log|x - x_0|, log|f(x) - f(x_0)|)$ dont la pente nous fournit l'exposant α. Notons qu'il existe d'autres méthodes d'estimation, comme celles fondées sur les bases d'ondelettes ([Mey92]).

La figure 4.2 illustre le calcul du coefficient de Hölder. On remarque dans les expressions précédentes que le terme $|h|$ (respectivement $|x - x_0|$) représente en fait la mesure de Lebesgue de l'intervalle $[x, x + h]$ (respectivement $|f(x) - f(x_0)|$). L'utilisation d'une mesure différente, provenant par exemple d'une autre image, permet d'analyser l'irrégularité d'une image à l'aide d'un spectre dit *mutuel*. L'estimation du coefficient α résulte alors de la régression faite sur le nuage de points $(\log|f(x) - f(x_0)|, \log|g(x) - g(x_0)|)$, où g définit la *mesure de référence*. L'image des coefficients de Hölder de la figure 4.3 a été calculée à l'aide d'une mesure de référence générée par une loi uniforme ; celle de la figure 4.4 page 91 l'a été par une mesure de référence binômiale.

4.1.2 Dimensions fractales

Comme nous l'avons évoqué en introduction, il existe différents moyens pour évaluer la régularité d'un ensemble ou d'une fonction. Pour cela, les dimensions

FIGURE 4.2: Image des coefficients de Hölder de Lena par rapport à la mesure de Lebesgue

fractales sont nombreuses et nous en présentons ici deux classiques. De plus amples informations sont disponibles dans [Fal90] et [Tri95].

Nous commençons par la *dimension de Hausdorff* qui est la plus générale mais se révèle, dans la majorité des cas, inutilisable en pratique. Pour les applications, on lui préfère souvent la *dimension de boîtes* que nous aborderons ensuite.

4.1.2.1 Mesure et dimension de Hausdorff

La dimension de Hausdorff repose sur la mesure de Hausdorff. Dans un espace Euclidien E de dimension D, on définit pour chaque élément $\Omega \subset E$ son diamètre, noté $|\Omega|$, comme la borne supérieure, éventuellement infinie, des distances entre deux points de Ω. Un ϵ-recouvrement se définit alors par :

Définition 4.1 *Si $\{U_i\}$ est un ensemble dénombrable d'ensembles tels que $\forall i, |U_i| < \epsilon$ et $\Omega \subset \bigcup_{i=1}^{\infty} U_i$, alors $\{U_i\}$ est un ϵ-recouvrement de Ω.*

Remarquons que si $\epsilon_1 \geq \epsilon_2$, alors l'ensemble des ϵ_1-recouvrements est contenu dans celui des ϵ_2-recouvrements.

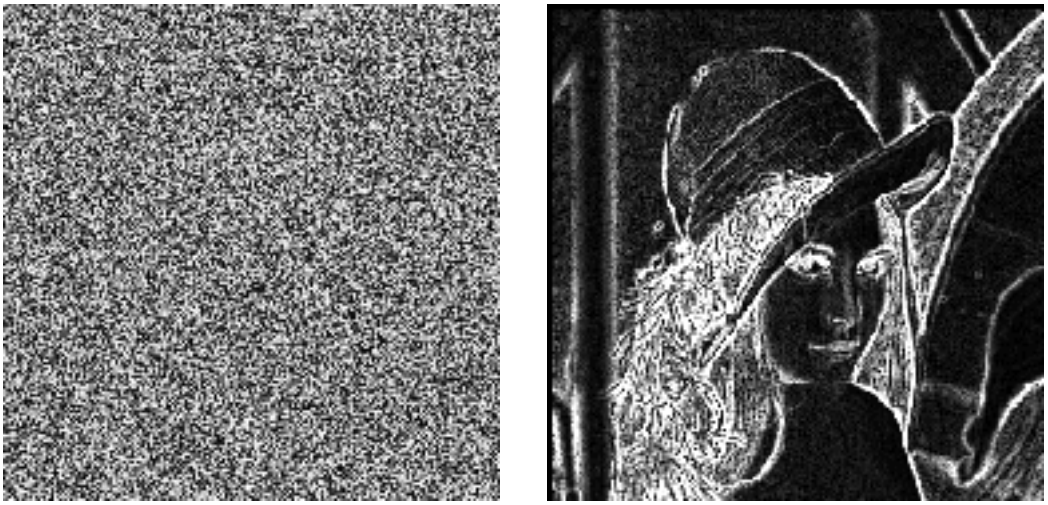


FIGURE 4.3: Image des coefficients de Hölder de Lena par rapport à une mesure générée aléatoirement

Étant donné un nombre réel $s \geq 0$, définissons pour tout Ω la quantité suivante :

$$\mathcal{H}^s_\epsilon(\Omega) = \left\{ \sum_{i=1}^{\infty} |U_i|^s \ ; \ |U_i| \ \epsilon\text{-recouvrement de } \Omega \right\} \tag{4.1}$$

Quand ϵ décroît, nous calculons une borne inférieure de quantités positives sur une suite imbriquée décroissante de recouvrements, donc $\mathcal{H}^s_\epsilon(\Omega)$ croît et admet par conséquent une limite, éventuellement infinie, appelée *mesure*[1] *de Hausdorff de dimension s de l'ensemble* Ω :

$$\mathcal{H}^s(\Omega) = \lim_{\epsilon \to 0} \mathcal{H}^s_\epsilon(\Omega)$$

Considérons à présent deux réels s et t tels que $0 < s < t$, et un ϵ-recouvrement $\{U_i\}$ de Ω, on a alors :

$$\sum_i |U_i|^t \leq \epsilon^{t-s} \sum_i |U_i|^s$$

d'où

$$\mathcal{H}^t_\epsilon(\Omega) \leq \epsilon^{t-s} \mathcal{H}^s_\epsilon(\Omega)$$

ce qui assure que, si $\mathcal{H}^s(\Omega) < \infty$ pour $s \in \mathbb{R}^+$, alors, en faisant tendre ϵ vers 0 dans la dernière équation, $\mathcal{H}^t(\Omega) = 0$ pour tout $t > s$.

Ainsi, il existe une valeur de coupure $D_H(\Omega)$ telle que, si $s < D_H(\Omega)$, alors la mesure de Hausdorff de dimension s est infinie, et inversement nulle pour $s > D_H(\Omega)$. On appelle *dimension de Hausdorff* cette valeur de coupure.

[1] Une démonstration rigoureuse est disponible dans [Fal90].

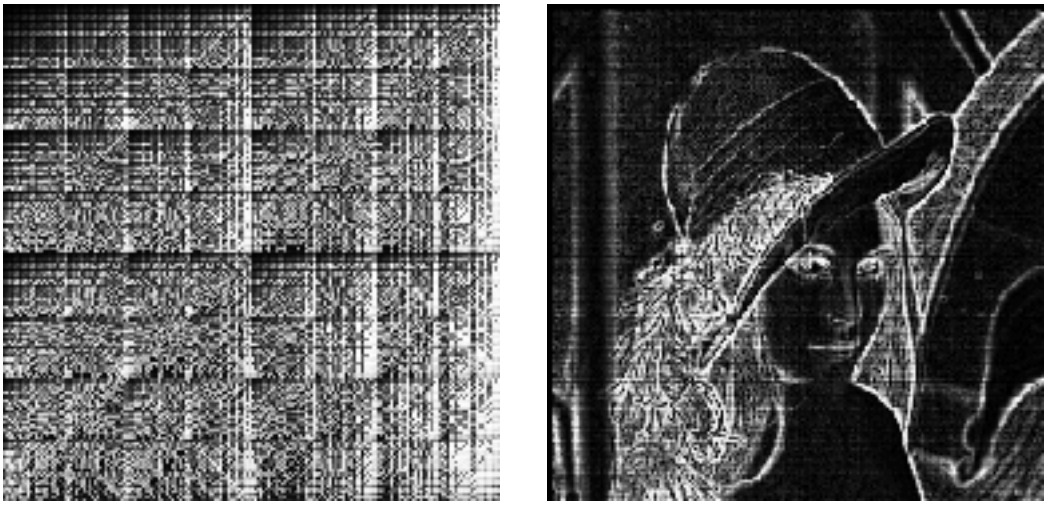


FIGURE 4.4: Image des coefficients de Hölder de Lena par rapport à une mesure binômiale

Définition 4.2 *La dimension de Hausdorff d'un ensemble* Ω, $D_H(\Omega)$, *est définie par :*

$$\begin{aligned} D_H(\Omega) &= \inf\{s \in \mathbb{R}, \mathcal{H}^s(\Omega) = 0\} \\ &= \sup\{s \in \mathbb{R}, \mathcal{H}^s(\Omega) = \infty\} \end{aligned}$$

Notons qu'en général nous n'avons pas d'information sur la valeur de la mesure de Hausdorff pour la valeur de coupure.

4.1.2.2 Dimension de boîtes

La dimension de Hausdorff est l'outil le mieux adapté pour décrire des objets fractals, mais elle présente l'inconvénient d'être difficile à estimer. On lui préfère en pratique la dimension de boîtes, non équivalente à la dimension de Hausdorff mais de signification intuitive voisine. Sa définition permet d'écrire naturellement des algorithmes d'évaluation :

Définition 4.3 *Soient* Ω *un ensemble borné de* E, $\epsilon > 0$ *et* $N_\epsilon(\Omega)$ *le plus petit nombre d'ensembles de diamètre au plus* ϵ *recouvrant* Ω. *On définit alors les dimensions de boîtes inférieures et supérieures de* Ω *par :*

$$\begin{aligned} D_B^-(\Omega) &= \lim_{\epsilon \to 0} \inf \frac{-\log N_\epsilon(\Omega)}{\log \epsilon} \\ D_B^+(\Omega) &= \lim_{\epsilon \to 0} \sup \frac{-\log N_\epsilon(\Omega)}{\log \epsilon} \end{aligned}$$

Si ces deux nombres coïncident, on appelle leur valeur commune dimension de boîtes de Ω, *notée* $D_B(\Omega)$.

Il est strictement équivalent de remplacer $N_\epsilon(\Omega)$ par le nombre de cubes intersectants Ω dans un pavage de l'espace par des cubes d'arêtes ϵ, ainsi que par toutes les définitions suivantes (voir démonstrations dans [Fal90]) :

- le plus petit nombre de boules fermées de diamètre ϵ recouvrant Ω ;
- le plus petit nombre de cubes d'arête ϵ recouvrant Ω ;
- le nombre de cubes, dans un pavage de l'espace par des cubes d'arête ϵ, intersectants Ω ;
- le plus petit nombre d'ensembles de diamètre au plus ϵ recouvrant Ω ;
- le plus grand nombre de boules disjointes de diamètre ϵ de centre dans Ω.

Selon le choix adopté pour $N_\epsilon(\Omega)$, les algorithmes de calcul de la dimension de boîtes diffèrent. Dans les applications, la définition où $N_\epsilon(\Omega)$ représente le nombre de cubes intersectants Ω est la plus pratique car elle dispense de la recherche d'un extremum.

Notons enfin la relation existant entre dimension de Hausdorff et dimension de boîtes :

$$D_H(\Omega) \leq D_B^-(\Omega) \leq D_B^+(\Omega)$$

4.1.3 Images et analyse multifractale

La géométrie fractale fournit un outil puissant pour l'analyse des ensembles. Dans le cas particulier du traitement d'images, une image dite *binaire*, c'est-à-dire constituée de point noirs et blancs (ou encore de 0 et de 1), définit donc un ensemble dans l'image : selon la valeur de la fonction indicatrice $\mathbb{1}_E$, un point P de l'image appartient à l'ensemble E si et seulement si $\mathbb{1}_E(P) = 1$.

Cette notion d'ensemble dans une image ne tient pas compte de l'information apportée par chaque point, comme son intensité lumineuse $I(x, y)$. Une approche classique pour tenir compte de cette information consiste alors à définir un espace à trois dimensions formé des points de coordonnées (x, y, z) où $z = I(x, y)$. Néanmoins, la transformation d'un problème bidimensionnel en un autre, tridimensionnel, accroît également la complexité algorithmique.

L'analyse multifractale est une notion portant sur les mesures, au même titre que la géométrie fractale se réfère aux ensembles. Aussi considérons nous l'intensité lumineuse comme une mesure portée par l'image afin de tenir compte de l'information locale. Ainsi, on calcule le coefficient de Hölder en chaque point (x, y) de l'image, en considérant que la valeur en ce point est $z = I(x, y)$. Il est alors possible de regrouper les points de l'image en ensembles contruits avec les points dont les coefficients de Hölder sont proches.

Dans un premier temps, on construit donc des ensembles qui possèdent un comportement local homogène puis on étudie ces ensembles d'un point de vue global. Le terme *multifractal* s'explique par cette dualité :

- le support de la mesure est partitionné en des régions possédant un comportement local identique (*i.e.* un exposant de Hölder voisin) ;
- chaque ensemble ainsi constitué est analysé dans sa globalité à l'aide d'une dimension fractale (celle de Hausdorff en théorie et celle de boîtes en pratique).

Estimation du spectre de Hausdorff

1. Estimation du coefficient de Hölder en tout point ;
2. Discrétisation en N parties égales de l'intervalle de variation de α, aboutissant ainsi à N ensembles iso-α ;
3. Calcul de la dimension de chaque ensemble iso-α, notée $f_H(\alpha)$;
4. Tracé de la courbe $(\alpha,\ f_H(\alpha))$.

TABLEAU 4.1 : Algorithme d'estimation du spectre de Hausdorff

4.1.4 Spectres multifractals de mesures

Trois types de spectres multifractals sont usuellement définis : celui de Hausdorff, celui de grandes déviations et celui de Legendre. L'étude présentée dans ce mémoire est fondée sur le spectre de Hausdorff

À chaque exposant α, on associe un ensemble E_α :

$$E_\alpha = \{t/\alpha(t) = \alpha\}$$

dont la dimension de Hausdorff f_H est :

$$f_H(\alpha) = \dim_H E_\alpha$$

La fonction f_H est appelé *spectre de Hausdorff du signal*, il fournit une caractérisation géométrique des singularités du signal. Son évaluation est décrite dans le tableau 4.1.

C'est ainsi qu'à l'aide de l'image des coefficients de Hölder (fig. 4.2), on détermine le spectre multifractal de l'image de Lena (fig. 4.5).

Le principal inconvénient de cette estimation vient de la discrétisation en N intervalles. Il faut en effet choisir une taille appropriée car si elle est trop grande, tous les points sont groupés dans peu d'ensembles, et inversement si elle est trop faible, les iso-α contiennent trop peu de points pour une estimation correcte de la dimension.

Nous avons pensé à une alternative en discrétisant le domaine des valeurs de α à l'aide d'intervalles de masse constante, et non plus de taille constante (voir [EM92]). La discrétisation se fait sur le nombre de points contenus dans chaque intervalle. Cela permet d'obtenir des intervalles de tailles variables. Cette variante présente toutefois un problème similaire à la précédente. En effet, définir les ensembles avec trop peu de points ne conduit pas à une estimation pertinente. Au contraire, si trop de points appartiennent aux ensembles, on ne dispose pas de suffisamment d'ensembles pour estimer le spectre.

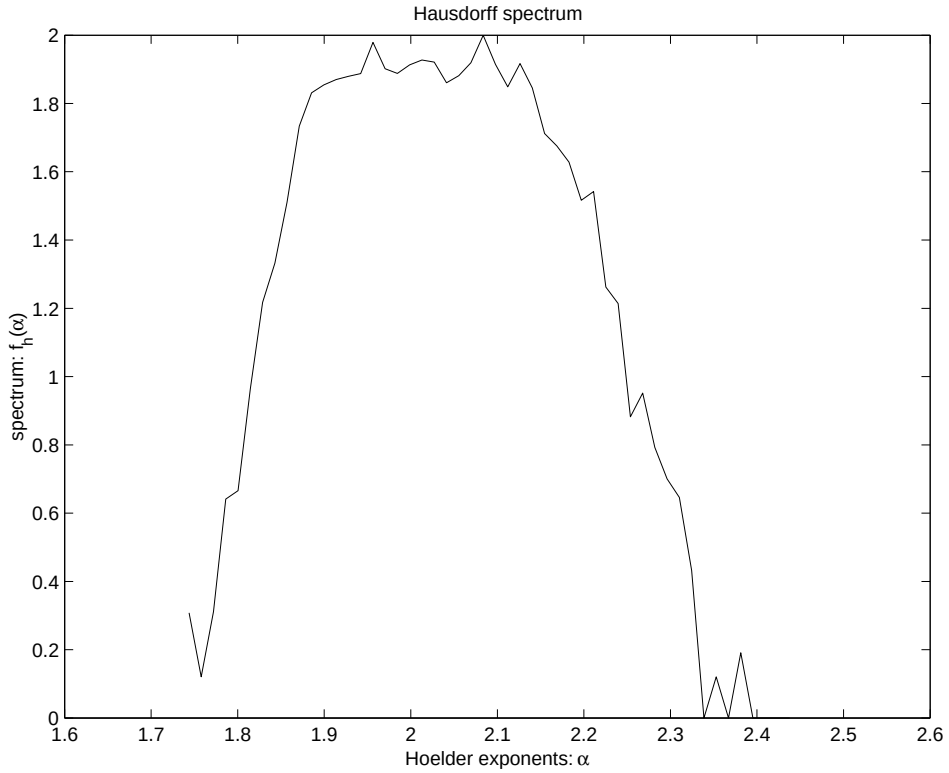


FIGURE 4.5: Spectre de Hausdorff de Lena par rapport à la mesure de Lebesgue

4.2 Approche envisagée

Des travaux récents menés par J. Lévy Véhel et E. Lutton ([LL01]) ont montré qu'il était possible de modifier la régularité d'un signal sans pour autant altérer ce signal. Dans cette section, nous présentons la première ébauche d'un schéma de tatouage fondé sur l'emploi de spectres mutuels puis les résultats d'expériences destinées à analyser la validité de nos hypothèses.

4.2.1 Principe général d'un schéma substitutif

En se servant d'une mesure de référence (éventuellement générée à l'aide d'une clé secrète), on détermine un spectre multifractal de l'image qu'on souhaite marquer. Ensuite, grâce à la clé secrète, on modifie les exposants de Hölder de façon à ce que le spectre ait une caractéristique voulue. La détection se fait en utilisant la clé secrète pour s'assurer que le spectre de l'image marquée, relativement à la même mesure de référence, contient toujours la caractéristique insérée.

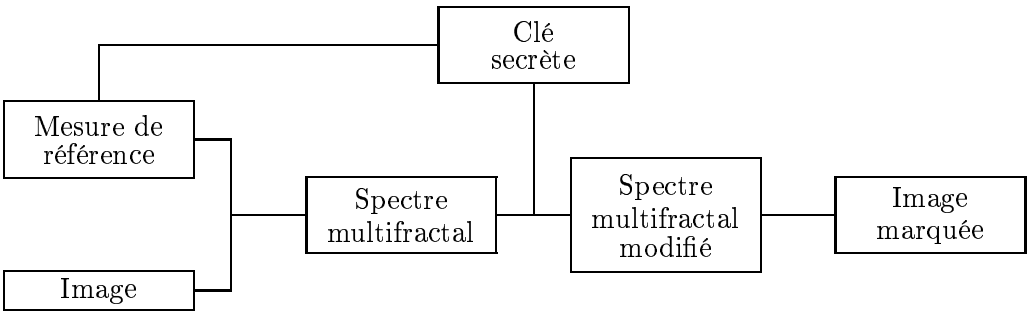


FIGURE 4.6 : Principe d'insertion fondé sur les spectres multifractals

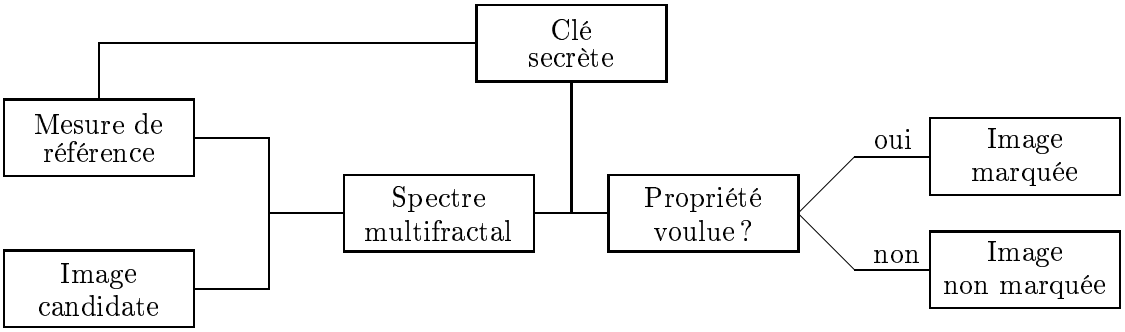


FIGURE 4.7 : Principe de détection fondé sur les spectres multifractals

4.2.2 Intérêts et problèmes

Cette approche présente de nombreux intérêts :
- la marque est théoriquement répartie spatialement dans toute l'image ;
- choix des gammes de fréquences : en sélectionnant une région du spectre à modifier, on désigne en fait plusieurs ensembles iso-α, ce qui permet donc de choisir les fréquences sur lesquelles on agit (basses fréquences / hautes fréquences, i.e. zones régulières / irrégulières) ;
- la méthode fondée sur les spectres mutuels offre une grande souplesse (choix de la clé et de la mesure de référence).

Toutefois, certains problèmes ne sont pas encore résolus :
- il faut trouver une modification du spectre qui soit à la fois robuste (*i.e.* résistante aux transformations éventuellement subies par l'image) et discriminante (pour résister au marquage multiple) ;
- il n'y a pas de bijection entre le spectre et l'image. On doit recalculer une image et s'assurer ensuite que son spectre possède bien la caractéristique voulue. Or, on ne sait pas, à l'exception de cas très particuliers, déterminer cette image (translation de spectre pour du débruitage [LG97], ajout d'un pic sur le spectre [CL96]).

4.2.3 Expériences et résultats

Avant de nous lancer plus avant dans l'élaboration de cette méthode, nous avons entrepris quelques expériences afin de tester certaines hypothèses. Nous présentons ici la démarche qui nous a permis d'évaluer les possibilités offertes par l'utilisation de spectres mutuels dans des schémas de tatouage.

Comme cela a été expliqué précédemment (cf. paragraphe 4.2.2), le changement dans le spectre doit résister aux éventuelles modifications de l'image mais doit également permettre de différencier des marques distinctes.

Des tests ont été mis en place pour déterminer les caractéristiques «intéressantes» du spectre. Par «intéressantes», on entend des attributs qui varient assez peu entre une stégo-image donnée, et ses versions bruitées, filtrées ..., *i.e.* soumises aux multiples attaques envisageables (voir chapitre II page 109).

En revanche, cette variance doit être assez élevée pour des mesures de référence différentes, ceci pour permettre de distinguer diverses marques éventuellement présentes. Cette propriété garantit la discrimination entre plusieurs marques.

Cette première idée semble offrir la sécurité nécessaire pour qu'une méthode de marquage soit efficace, c'est-à-dire un bon compromis entre la robustesse aux déformations et la discrimination entre plusieurs marques.

4.2.3.1 Protocole expérimental

Les principaux paramètres étudiés sur les spectres multifractals mutuels sont (cf. fig. 4.8 page suivante) :

- les exposants de Hölder extrêmes α_{min} et α_{max} du spectre ;
- le coefficient de Hölder $\alpha_{f_{max}}$ de la dimension maximale $f_{max}(\alpha)$;
- pour un ratio r (0.5 sur la figure 4.8 page ci-contre) donné de $f_{max}(\alpha)$, on mesure les coefficients de Hölder $\alpha_{(min,r)}$ et $\alpha_{(max,r)}$, ainsi que leur différence.

Chaque expérience réalisée repose sur les mêmes dix images (cf. fig. 4.9 page 98), de taille 512×512 et codées en niveaux de gris de 0 à 255.

Pour chaque test ci-après, les paramètres intéressants ont été mesurés sur l'image initiale et sur des versions modifiées de celle-ci afin d'évaluer la robustesse :

- ajout d'une perturbation sur chaque pixel d'une amplitude aléatoire comprise dans $[-25, +25]$ (bruit) ;
- chaque pixel possède une probabilité de 0.07 de voir sa valeur passer à 255 (impulsion) ;
- lissage par filtre Gaussien ($\sigma = 1$).

Dans les tableaux de résultats, nous notons :

- σ_r la variance obtenue pour un paramètre dans les différentes «versions» d'une image pour une mesure de référence identique à chaque estimation : elle doit être petite pour traduire la robustesse du paramètre aux modifications éventuellement subies par l'image ;

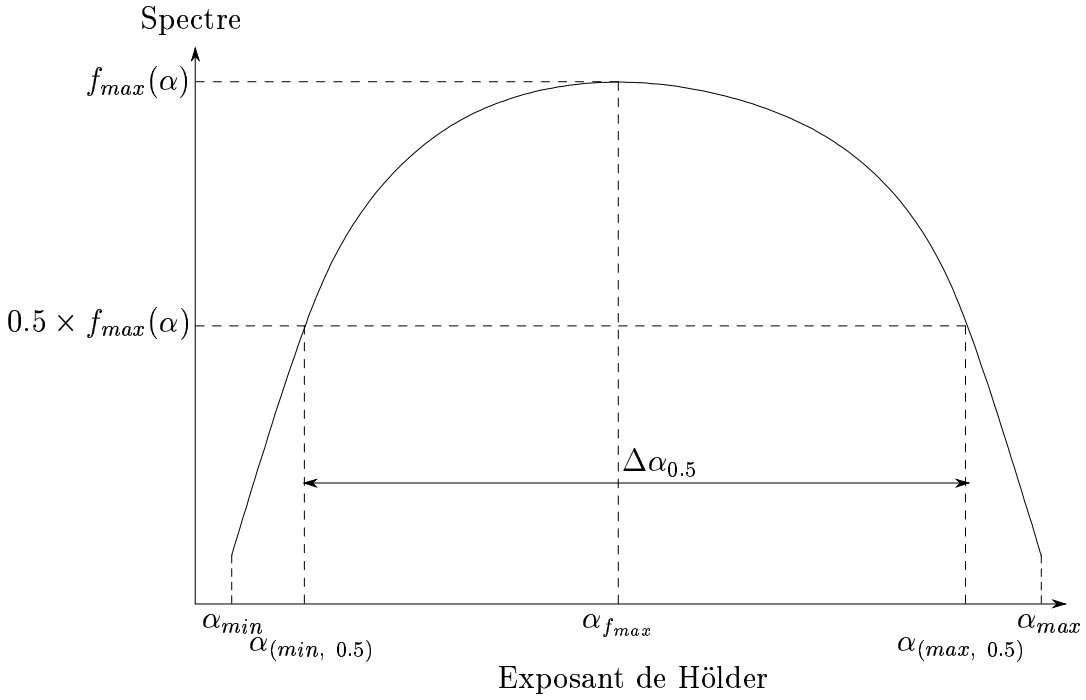


FIGURE 4.8 : Paramètres étudiés sur un spectre de Hausdorff

– σ_m la variance obtenue pour un paramètre dans une même image mais avec une mesure de référence différente à chaque estimation : elle doit être grande afin de permettre la distinction entre différentes mesures de références.

4.2.3.2 Mesures de référence aléatoires

Pour la première série de tests, nous avons calculé le spectre de chaque «version» de l'image relativement à dix mesures aléatoires uniformes sur $[0, 1]$ (Fig. 4.3). Les résultats sont présentés dans le tableau 4.2 page 100. Nous constatons que la variance σ_r varie très peu pour les différentes images, ce qui traduit une bonne robustesse aux attaques ou perturbations. Cependant, il semble que la variance σ_m ne permet pas de discriminer les différentes mesures employées.

Nous nous sommes alors demandés si ce phénomène ne provenait pas des mesures de référence. En effet, un spectre multifractal tient compte des informations locales mais aussi de l'aspect global des mesures utilisées pour estimer les coefficients de Hölder. Or, toutes les mesures de référence sont statistiquement identiques. Nous avons ensuite testé des mesures de référence générées selon des lois différentes.


Couple


École


Géographe (Vermeer)


Goldhill


Mandrille


Lena


Poupée


Savanne


Théière


L'homme blessé
(Courbet)

FIGURE 4.9 : Images 512 × 512 utilisées pour les expériences

4.2.3.3 Mesures de référence statistiquement différentes

Suite à ce constat, nous avons entrepris une nouvelle série de tests afin d'étudier le comportement du spectre face à des mesures générées par des lois de nature différente :

1. loi uniforme $\mathcal{U}([0,\ 1])$;
2. loi uniforme sur $\mathcal{U}([\frac{1}{2},\ 1])$;
3. loi uniforme sur $\mathcal{U}([0,\ 1])$ ensuite seuillée à $\frac{1}{2}$, notée $\mathcal{U}([0,\ 1] \geq \frac{1}{2})$;
4. loi normale $\mathcal{N}(\frac{1}{2},\ \frac{1}{2})$;
5. loi normale $\mathcal{N}(0.5,\ 0.2)$;
6. loi normale $\mathcal{N}(0.2,\ 0.2)$;
7. loi normale $\mathcal{N}(0.8,\ 0.2)$.

La tableau 4.3 page 101 présente les résultats obtenus. Pour chaque image, nous avons testé les sept mesures de référence précédentes dix fois sur chaque version bruitée de l'image. On constate que le changement de loi ne permet toujours pas de distinguer celle-ci avec les différents paramètres sélectionnés. En effet, si la variance pour la robustesse σ_r reste faible, la variance σ_m devant permettre de distinguer les mesures est toujours trop basse.

4.3 Conclusion et perspectives

Les résultats précédents démontrent que, si les mesures de référence ont peu de rapport avec l'image, on ne peut pas distinguer de propriétés suffisamment caractéristiques sur le spectre multifractal. Ainsi, la méthode imaginée ne semble pas suffisamment discriminante pour permettre de distinguer des marques différentes.

Une autre direction à explorer, toujours fondée sur l'utilisation des mesures de référence, pourrait venir des techniques de détection de changements dans une image. Par exemple, pour découvrir ce qui a changé entre deux photos satellite successives, on peut étudier l'image des coefficients de Hölder de la seconde photo en se servant de la première comme mesure de référence. Les deux images étant relativement inchangée, seules les régions différentes sont révélées comme zones irrégulières. On voit alors apparaître un pic sur le spectre de Hausdorff (cf. [CL96]). Cependant, une méthode de tatouage s'appuyant sur un tel procédé nécessite l'utilisation de l'image initiale dans la fonction de vérification.

		Couple	Poupée	École	Géographe	Goldhill	Courbet	Lena	Mandrille	Savanne	Théière
α_{min}	σ_r	$1.316e-05$	$2.612e-05$	$8.922e-06$	$3.692e-05$	$2.890e-05$	$2.600e-05$	$1.162e-05$	$1.660e-05$	$2.982e-05$	$6.792e-05$
	σ_m	$7.892e-06$	$4.748e-06$	$2.966e-03$	$6.122e-05$	$1.609e-05$	$1.226e-04$	$1.355e-06$	$4.462e-06$	$1.436e-05$	$1.616e-05$
α_{max}	σ_r	$5.718e-04$	$2.309e-04$	$9.196e-04$	$2.617e-05$	$2.287e-05$	$7.407e-05$	$1.431e-05$	$1.558e-04$	$1.452e-05$	$8.535e-05$
	σ_m	$1.275e-02$	$2.126e-03$	$3.316e-01$	$2.770e-04$	$3.460e-04$	$5.622e-03$	$2.636e-05$	$5.826e-04$	$2.346e-04$	$1.946e-04$
$\alpha_{f_{max}}$	σ_r	$8.902e-05$	$2.098e-04$	$7.948e-04$	$6.285e-05$	$2.059e-05$	$4.138e-05$	$1.682e-05$	$4.387e-05$	$9.916e-04$	$1.713e-04$
	σ_m	$8.223e-05$	$1.796e-04$	$1.455e-02$	$8.615e-05$	$1.458e-05$	$5.864e-05$	$7.799e-06$	$3.593e-05$	$1.641e-03$	$1.235e-04$
$\alpha_{(min,0.5)}$	σ_r	$9.181e-07$	$1.232e-06$	$6.284e-06$	$2.811e-06$	$9.902e-07$	$7.184e-07$	$1.731e-07$	$2.562e-07$	$1.051e-06$	$3.465e-06$
	σ_m	$1.431e-05$	$9.256e-06$	$3.614e-02$	$7.398e-05$	$1.912e-05$	$1.484e-04$	$1.577e-06$	$4.178e-06$	$1.524e-05$	$1.979e-05$
$\alpha_{(max,0.5)}$	σ_r	$3.097e-05$	$2.430e-05$	$3.746e-05$	$1.555e-06$	$8.939e-06$	$6.233e-06$	$1.170e-06$	$2.429e-06$	$2.613e-06$	$3.703e-06$
	σ_m	$3.140e-04$	$1.611e-04$	$1.782e-02$	$1.170e-04$	$2.485e-04$	$1.792e-03$	$1.366e-05$	$4.406e-05$	$1.240e-04$	$8.456e-05$
$\Delta\alpha_{0.5}$	σ_r	$3.543e-05$	$2.749e-05$	$7.025e-05$	$4.901e-06$	$1.165e-05$	$7.865e-06$	$1.409e-06$	$3.029e-06$	$3.368e-06$	$6.061e-06$
	σ_m	$4.586e-04$	$2.436e-04$	$1.046e-01$	$3.172e-04$	$4.038e-04$	$2.910e-03$	$2.369e-05$	$7.457e-05$	$2.242e-04$	$1.823e-04$
$\alpha_{(min,0.7)}$	σ_r	$7.369e-07$	$5.535e-07$	$1.640e-05$	$5.373e-06$	$9.212e-07$	$3.700e-06$	$3.215e-07$	$3.590e-07$	$3.254e-07$	$3.349e-06$
	σ_m	$1.410e-05$	$8.802e-06$	$9.972e-03$	$6.631e-05$	$1.741e-05$	$1.614e-04$	$1.695e-06$	$4.680e-06$	$1.435e-05$	$2.144e-05$
$\alpha_{(max,0.7)}$	σ_r	$1.276e-05$	$1.496e-05$	$8.469e-06$	$8.376e-07$	$4.560e-06$	$6.319e-06$	$1.120e-06$	$2.883e-06$	$5.425e-06$	$3.731e-06$
	σ_m	$1.192e-04$	$1.795e-05$	$1.169e-02$	$9.995e-05$	$3.002e-05$	$3.776e-04$	$1.871e-05$	$9.964e-06$	$3.470e-05$	$3.013e-05$
$\Delta\alpha_{0.7}$	σ_r	$1.425e-05$	$1.522e-05$	$3.968e-05$	$6.823e-06$	$6.738e-06$	$1.044e-05$	$2.050e-06$	$3.882e-06$	$5.212e-06$	$9.468e-06$
	σ_m	$2.117e-04$	$3.734e-05$	$4.294e-02$	$3.195e-04$	$9.128e-05$	$1.025e-03$	$3.127e-05$	$2.622e-05$	$9.161e-05$	$1.006e-04$
$\alpha_{(min,0.9)}$	σ_r	$3.672e-06$	$2.987e-06$	$2.123e-05$	$5.564e-06$	$5.671e-06$	$3.188e-06$	$5.318e-07$	$6.411e-07$	$4.178e-06$	$4.484e-05$
	σ_m	$1.080e-05$	$9.002e-06$	$3.795e-04$	$1.712e-05$	$3.886e-06$	$8.456e-05$	$1.981e-06$	$4.328e-06$	$9.214e-06$	$4.284e-05$
$\alpha_{(max,0.9)}$	σ_r	$4.692e-06$	$5.228e-06$	$7.809e-06$	$3.083e-06$	$1.521e-05$	$7.781e-06$	$3.759e-06$	$3.056e-06$	$1.591e-06$	$3.516e-06$
	σ_m	$2.213e-05$	$5.320e-06$	$8.202e-03$	$7.136e-05$	$7.237e-06$	$5.408e-05$	$1.965e-06$	$2.465e-06$	$2.208e-05$	$2.178e-05$
$\Delta\alpha_{0.9}$	σ_r	$1.360e-05$	$1.050e-05$	$4.594e-05$	$1.222e-05$	$2.673e-05$	$1.666e-05$	$6.306e-06$	$5.014e-06$	$7.331e-06$	$6.231e-05$
	σ_m	$5.606e-05$	$2.266e-05$	$1.192e-02$	$8.397e-05$	$1.544e-05$	$8.863e-05$	$3.566e-06$	$1.038e-05$	$5.781e-05$	$1.059e-04$

TABLEAU 4.2 : Résultats obtenus avec des mesures de référence générées selon une loi uniforme

		Couple	Poupée	École	Géographe	Goldhill	Courbet	Lena	Mandrille	Savanne	Théière
α_{min}	σ_r	$8.595e-06$	$6.933e-06$	$3.058e-03$	$6.870e-05$	$1.798e-05$	$1.357e-04$	$2.128e-06$	$3.543e-06$	$1.693e-05$	$2.009e-05$
	σ_m	$1.737e-04$	$3.150e-04$	$1.152e-04$	$1.968e-04$	$2.997e-04$	$1.832e-04$	$3.119e-04$	$2.620e-04$	$1.647e-04$	$2.737e-04$
α_{max}	σ_r	$1.178e-02$	$1.850e-03$	$3.247e+01$	$2.875e-04$	$3.578e-04$	$5.884e-03$	$3.717e-05$	$7.878e-04$	$2.115e-04$	$1.692e-04$
	σ_m	$3.783e-04$	$1.354e-04$	$2.603e-03$	$7.368e-05$	$8.698e-05$	$6.142e-05$	$1.087e-04$	$1.827e-04$	$1.214e-04$	$6.396e-05$
$\alpha_{f_{max}}$	σ_r	$4.370e-05$	$9.631e-06$	$6.052e-03$	$6.989e-05$	$1.146e-05$	$3.354e-05$	$6.948e-06$	$3.113e-05$	$1.839e-03$	$1.630e-04$
	σ_m	$5.433e-05$	$1.137e-05$	$3.629e-03$	$4.329e-05$	$1.489e-05$	$3.153e-05$	$1.266e-05$	$3.916e-05$	$8.481e-04$	$2.765e-04$
$\alpha_{(min,0.5)}$	σ_r	$5.393e-03$	$2.461e-03$	$2.998e-02$	$3.061e-04$	$1.206e-03$	$3.116e-03$	$2.164e-04$	$1.717e-03$	$1.261e-03$	$8.265e-04$
	σ_m	$1.397e-02$	$9.936e-03$	$6.021e-03$	$8.842e-04$	$1.956e-02$	$4.008e-03$	$4.270e-04$	$2.493e-03$	$3.473e-03$	$2.710e-02$
$\alpha_{(max,0.5)}$	σ_r	$3.513e-04$	$1.660e-04$	$2.237e-02$	$1.322e-04$	$2.922e-04$	$1.992e-03$	$1.261e-05$	$4.532e-05$	$1.289e-04$	$6.481e-05$
	σ_m	$1.733e-05$	$2.710e-05$	$6.429e-06$	$1.165e-05$	$8.660e-06$	$5.332e-06$	$1.365e-05$	$7.152e-06$	$1.131e-05$	$1.144e-05$
$\Delta\alpha_{0.5}$	σ_r	$7.123e-03$	$3.062e-03$	$9.176e-02$	$6.863e-04$	$2.061e-03$	$8.462e-03$	$2.480e-04$	$2.085e-03$	$1.680e-03$	$9.642e-04$
	σ_m	$1.394e-02$	$9.529e-03$	$5.927e-03$	$8.545e-04$	$1.914e-02$	$3.947e-03$	$4.675e-04$	$2.451e-03$	$3.392e-03$	$2.637e-02$
$\alpha_{(min,0.7)}$	σ_r	$9.643e-05$	$8.456e-06$	$8.629e-03$	$1.397e-04$	$3.480e-04$	$1.670e-04$	$1.767e-06$	$4.798e-06$	$1.750e-05$	$5.951e-04$
	σ_m	$7.309e-05$	$2.380e-05$	$4.092e-04$	$1.141e-05$	$7.874e-04$	$6.053e-06$	$2.117e-05$	$1.431e-05$	$5.173e-06$	$3.660e-03$
$\alpha_{(max,0.7)}$	σ_r	$1.057e-04$	$3.489e-05$	$1.429e-02$	$1.245e-04$	$3.172e-05$	$4.264e-04$	$2.154e-05$	$2.022e-05$	$2.367e-05$	$2.586e-05$
	σ_m	$6.357e-06$	$1.028e-04$	$1.166e-05$	$1.910e-06$	$1.375e-05$	$1.920e-05$	$3.085e-06$	$1.740e-05$	$4.005e-05$	$1.663e-05$
$\Delta\alpha_{0.7}$	σ_r	$3.342e-04$	$5.348e-05$	$4.463e-02$	$5.208e-04$	$5.261e-04$	$1.109e-03$	$3.541e-05$	$3.887e-05$	$7.007e-05$	$6.553e-04$
	σ_m	$9.707e-05$	$2.157e-04$	$3.880e-04$	$1.679e-05$	$7.064e-04$	$1.896e-05$	$3.914e-05$	$5.811e-05$	$6.696e-05$	$3.411e-03$
$\alpha_{(min,0.9)}$	σ_r	$1.089e-05$	$3.095e-04$	$4.786e-04$	$1.993e-05$	$6.064e-06$	$9.576e-05$	$1.731e-06$	$4.947e-06$	$9.838e-06$	$1.363e-04$
	σ_m	$4.059e-06$	$2.438e-04$	$1.808e-05$	$1.225e-05$	$2.386e-05$	$5.663e-06$	$2.552e-06$	$1.219e-06$	$1.093e-05$	$1.776e-04$
$\alpha_{(max,0.7)}$	σ_r	$1.551e-05$	$4.754e-06$	$1.072e-02$	$7.030e-05$	$2.103e-05$	$6.636e-05$	$3.070e-06$	$3.704e-06$	$2.235e-05$	$1.516e-05$
	σ_m	$5.387e-06$	$3.107e-05$	$1.612e-05$	$3.261e-06$	$4.243e-05$	$5.026e-06$	$1.682e-05$	$4.191e-06$	$7.752e-06$	$1.070e-05$
$\Delta\alpha_{0.9}$	σ_r	$4.793e-05$	$3.444e-04$	$1.562e-02$	$6.963e-05$	$4.320e-05$	$1.155e-04$	$3.909e-06$	$1.487e-05$	$5.977e-05$	$1.611e-04$
	σ_m	$1.747e-05$	$3.376e-04$	$2.265e-05$	$2.490e-05$	$1.236e-04$	$1.798e-05$	$3.123e-05$	$4.178e-06$	$3.382e-05$	$2.630e-04$

TABLEAU 4.3 : Résultats obtenus avec des mesures de référence de lois différentes

Conclusion et perspectives

Dans cette partie, nous avons essayé d'utiliser deux outils issus de la théorie fractale dans le cadre du tatouage d'images.

Tout d'abord, nous nous sommes intéressés à une solution fondée sur la génération sous contraintes d'IFS, et au problème inverse pour les IFS. Si des solutions à ce problème existent dans des cas particuliers, sa résolution s'avère bien délicate dès que certaines contraintes sont relâchées. Pour cela, nous avons introduits les IFS mixtes, construits à partir de fonctions quelconques. À l'aide d'un algorithme évolutionnaire (AE), nous parvenons à générer des attracteurs qui vérifient des contraintes géométriques imposées.

Toutefois, peu de fonctions sont naturellement contractantes et l'AE utilise beaucoup de ressources pour parvenir à ce résultat. Afin remédier à cette situation, nous avons envisagé deux modifications importantes :

1. nous avons construits des *IFS polaires* à partir de fonctions *localement contractantes vis-à-vis d'un point* car ces fonctions sont naturellement plus souvent contractantes que des fonctions quelconques ;
2. nous avons changé le comportement classique de l'AE, avec l'*approche Parisienne*, pour qu'il emploie au mieux les ressources dont il dispose en construisant la solution à partir d'un sous-ensemble de la population.

Ces deux améliorations se sont révélées particulièrement efficaces dans la résolution du problème inverse, surtout par rapport aux méthodes antérieures fondées sur les IFS mixtes. Néanmoins, les performances obtenues ne permettent pas encore de s'affranchir des contraintes imposées dans le cadre de la compression fractale.

Dans un second temps, nous avons voulu élaborer une solution de tatouage fondée sur l'utilisation des spectres multifractals mutuels. Nous souhaitions dissimuler la marque à l'aide de la mesure mutuelle afin de donner au spectre une forme caractéristique. Nos expériences préalables nous ont toutefois révélé que, bien que la robustesse de la méthode semble probable, cette approche souffre d'une incapacité à distinguer des mesures différentes à l'aide du spectre de Hausdorff.

Nous avons signalé qu'il n'existait pas de bijection entre le spectre et l'image. Cela signifie que le problème de calculer le spectre lorsqu'on dispose de l'image et de la mesure de référence est facile. En revanche, on ne sait pas retrouver l'image ou la mesure de référence à partir du spectre et de l'une de ces images

dans la plupart des cas. Cette situation est similaire à celle des *fonctions à sens uniques* utilisées en cryptographie. La recherche d'une trappe pourrait alors permettre l'élaboration d'un schéma asymétrique, comme celui reposant sur le calcul spectre de Fourier présenté par T. Furon et P. Duhamel dans [FD00]. Néanmoins, il semble nécessaire de résoudre préalablement les problèmes de différenciation.

La mise au point du protocole expérimental qui nous a permis de tester nos hypothèses sur les spectres mutuels nous a semblé intéressante à étendre. En effet, la plupart des solutions de tatouage proposées fournissent une analyse assez réduite de leurs performances. Souvent, seules quelques images et transformations sont expérimentées. Le premier chapitre de la seconde partie de ce mémoire traite ainsi de la question de l'évaluation des méthodes de tatouage.

Deuxième partie

Protocoles : évaluation, cryptographie et dissimulation d'information

Introduction

Cette partie est consacrée à deux questions qui apparaissent souvent dans les documents liés à la dissimulation d'information :

1. l'évaluation de l'efficacité d'une méthode, en particulier pour le tatouage ;
2. les rapports qui existent entre ce domaine et la cryptographie.

Les méthodes de tatouage sont habituellement testées séparément, selon des approches et des critères dissemblables, qui rendent les comparaisons entre ces différentes solutions pour le moins hasardeuses. La seule solution permettant d'obtenir une estimation objective est de laisser un tiers la pratiquer, à condition que les outils employés soient eux-mêmes publics.

Notre outil d'évaluation présenté en chapitre II page 109 se décompose en plusieurs modules :

- des fonctions bas niveau, applicables directement sur les média, comme des transformations géométriques ou des fonctions de filtrage ;
- des tests, qui prennent un médium marqué et l'altèrent grâce aux transformations précédentes ;
- une base de données pour stocker les résultats ;
- un protocole de soumission des bibliothèques de tatouage.

Notre logiciel s'intéresse uniquement à un schéma donné mais ne le considère pas dans le cadre d'une application. C'est dans cette perspective que nous nous sommes interrogés sur les liens existant réellement entre la dissimulation d'information et la cryptographie puisque cette discipline propose à la fois des algorithmes et les protocoles pour les utiliser de manière sécurisée.

La plupart des travaux traitant de dissimulation d'information soulignent l'existence de ce lien. Néanmoins, personne n'a encore approfondi cette assertion. Nous nous proposons donc d'explorer cette voie dans le chapitre 5 page 129 (un rappel des principales notions de la cryptographie est fourni en annexe B page 189).

La *cryptographie* concerne la sécurité des informations. Il s'agit d'un ensemble de techniques qui poursuivent différents objectifs :

- le *chiffrement* cherche à assurer la *confidentialité*, transformant les données de manière incompréhensible pour qui n'est pas autorisé à y accéder ;
- l'*authentification* est liée à l'identification des entités, et donne les moyens de garantir que la personne avec laquelle un échange a lieu est bien celle qu'elle prétend être ;

– la *signature* procure à la fois l'*intégrité des données*, qui implique que les informations n'ont pas été modifiées, et la *non répudiation*, qui interdit à l'auteur d'un message de nier *a posteriori* en être l'émetteur.

Les solutions proposées reposent sur des propriétés qui, lorsqu'elles sont réunies, fournissent un certain niveau de sécurité. Si l'une d'elle vient à manquer, les risques auxquels cette approche est exposée sont connus, c'est-à-dire qu'il est possible d'évaluer précisément l'apport d'une propriété à un schéma. Pour cela, des modèles détaillés d'attaques existent, décrivant les conditions dans lesquelles elles sont réalisables.

La cryptographie fournit des techniques, et des protocoles indiquant comment les appliquer. Par exemple, dans le cas du chiffrement, celui-ci seul ne suffit pas à garantir la sécurité du système. S'il s'agit de chiffrement à clé secrète, l'échange de la clé du chiffreur au déchiffreur représente le point faible : la clé risque d'être interceptée. Pour le chiffrement à clé publique, il est nécessaire de garantir que la clé publique utilisée est bien celle de la personne à qui on souhaite envoyer un message. Pour résoudre ces problèmes, les cryptographes s'attachent autant à proposer des algorithmes qu'à étudier les moyens de les mettre en pratique.

Au contraire, la dissimulation d'information, qui n'en est encore qu'à ses débuts, se préoccupe essentiellement des schémas en eux-mêmes. La mise en place du schéma de dissimulation d'information, dans son intégralité, est souvent laissée de côté : les solutions proposées s'intéressent principalement à la manière d'insérer un signal dans un autre, puis à sa détection.

Pour chaque solution soumise en cryptographie, on dispose d'une évaluation plus ou moins précise du niveau de sécurité. Il existe des arguments qui permettent de classer la difficulté des problèmes sur lesquels reposent les algorithmes de chiffrement. Ainsi, même si on ne connaît pas toujours la complexité exacte d'une attaque contre un tel algorithme, on dispose néanmoins d'un ordre de grandeur suffisamment précis pour apprécier la fiabilité de la solution.

En dissimulation d'information, et particulièrement en tatouage, la difficulté se situe plutôt dans l'autre sens : il existe de nombreux schémas dont la sécurité n'est évaluée qu'expérimentalement sur quelques signaux. Rares sont les solutions proposées qui étudient l'impact de la clé (taille de l'espace des clés, collisions...) ou les attaques protocolaires envisageables.

Dans le chapitre 5, nous reprenons les principaux concepts cryptographiques. Nous analysons en quoi ils correspondent à la problématique de la dissimulation d'information, si ils sont directement applicables dans l'algorithme ou plutôt rattachés au protocole qui l'entoure. Ce travail de prospection montre les voies à explorer afin d'accroître la sécurité des schémas proposés pour la dissimulation d'information en adaptant les solutions efficaces fournies par la cryptographie.

Enfin, dans un dernier chapitre (cf. ch. 6 page 159) plus concret, nous recherchons un canal stéganographique dans un endroit assez peu exploré : les protocoles réseau. Une rapide analyse montre que les protocoles des couches réseau et transport du modèle OSI ne conviennent pas. Nous montrons alors l'existence d'un tel canal dans le protocole SSH et en décrivons l'emploi.

chapterÉvaluation des méthodes de tatouage

4.4 Introduction

Le domaine du tatouage numérique connaît un essor important depuis quelques années et le besoin de solutions efficaces se fait d'autant plus sentir que les outils numériques deviennent de plus en plus accessibles. Les techniques de tatouage se multiplient, mais aucune ne parvient encore à s'imposer. En effet, les utilisateurs potentiels de schémas de tatouage ne savent à quel algorithme se fier car il n'existe pas de programme permettant une évaluation fine bien que les premières tentatives apparurent dès 1998 [PAK98]. Ce manque de référence provoque une grande confusion qui empêche tous les acteurs concernés (ayants droit de ces média, fabricants de matériels ou éditeurs de logiciels) de sélectionner une solution appropriée à leurs besoins : fonder une politique de protection à long terme sur des schémas peu testés ne ressemble pas à une idée pertinente.

Partant de ce constat, en collaboration avec Matthieu Brunet[2] et Caroline Fontaine[3], nous nous sommes lancés dans la mise au point d'un outil d'évaluation automatique. Nous avons contacté Fabien Petitcolas[4] en Janvier 2001, car il avait réalisé le logiciel de tests `stirmark`, afin de mettre au point le protocole expérimental et la structure générale du programme. Une fois le modèle générique, indépendant du médium considéré, mis au point, nous avons commencé à l'adapter aux images. Par la suite, Martin Steinebach[5] et Jana Dittman, de l'Université de Darmstadt en Allemagne, nous ont rejoints pour traiter la partie audio à partir du travail déjà réalisé pour les images ([PSR+01, SPR+01, RPF01]).

Deux autres projets similaires existent. Tout d'abord, le projet Européen Certimark[5], lancé en Mai 2001 sous la direction de C. Rollin (Société des Auteurs et Compositeurs Dramatiques - SACD), regroupe de nombreuses équipes (SACD, INA, Philips, Thalès Communication, EPFL, EURECOM, Université catholique de Louvain, NetImage ...). Ce projet travaille principalement dans deux directions :

1. élaboration et développement d'un outil d'évaluation pour le tatouage ;
   - outil d'évaluation complet pour les images et les vidéos ;
   - création de nouvelles attaques, élaboration de nouveaux tests pour tous types de paramètres, définition d'une nouvelle mesure qualitative pour les images.
2. recherche sur des algorithmes de tatouage :
   - élaboration de méthodes de tatouage en profitant des compétences variées de tous les participants ;
   - évaluation des techniques les plus efficaces pour une exploitation commerciale future

[2] de l'Institut National de Recherche en Informatique et Automatique (INRIA)
[3] du Laboratoire d'Informatique Fondamentale de Lille (LIFL)
[4] de Microsoft Research, U.K.
[5] http://www.certimark.org/

On trouve également le projet `checkmark`[6] initialisé par S. Pereira et supervisé par T. Pun ([PVM+01, VPP+01]). Ce programme contient de nombreux tests (compression par ondelettes, attaque par recopie par exemple) et s'appuie sur une métrique plus performante que le PSNR : la métrique de Watson ([MEC98]). Celle-ci prend mieux en compte la luminance et le contraste d'une image que le PSNR. Ce logiciel est développé pour `Matlab 6` et les résultats fournis en XML.

Ces projets sont complémentaires, en particulier par le grand nombre de tests différents qu'ils proposent. Par ailleurs, ils offrent la possibilité de travailler dans des environnements différents, tant par le langage utilisé (`Matlab` ou `C++`) que par le support des média (images uniquement, ou bien également des fichiers audios ou vidéos).

Nous présentons tout d'abord les conditions générales indispensables à l'utilisation d'un logiciel d'évaluation. Ensuite, nous introduisons l'architecture globale du système. Enfin, une dernière partie est consacrée à l'évaluation elle-même, détaillant les niveaux d'assurances de différents critères ainsi que de nouveaux tests.

4.5 Pré-requis pour un outil d'évaluation

Le tatouage numérique reste un champ d'investigations où les évaluations précises sont encore peu répandues. D'une part, peu d'organismes ont déjà fourni un cahier des charges complet, détaillant précisément les spécifications attendues qui permettraient de valider, ou au moins de tester, les méthodes proposées ([Int97], [Eur00]). D'autre part, les équipes de recherche ou les sociétés ne publient que très rarement les résultats des tests intensifs menés sur leurs solutions [Bra98].

De plus en plus d'attaques sont recensées contre les schémas de tatouage ([PAK98, LvD98, SG00, KVH00, HSG00]). Cela illustre bien la nécessité d'accroître les performances des algorithmes de tatouage de sorte que les nouveaux standards multimédia puissent s'appliquer, entre autre, à la protection des droits d'auteur.

Les tests réalisés dans diverses publications ne présentent que des résultats partiels, obtenus à partir d'un ensemble restreint de média et d'attaques, en appliquant un protocole expérimental personnel. Ce point, déjà présenté dans [KP99b], démontre l'impossibilité de comparer les solutions élaborées les unes par rapport aux autres, à moins de reprogrammer soi-même les algorithmes. Cependant, cela conduirait très certainement à des implantations informatiques différentes, probablement moins performantes que celles des auteurs initiaux puisque les détails des algorithmes, très importants pour une comparaison impartiale, sont rarement publiés. Ce manque d'étalonnage laisse à penser que le besoin d'une référence commune dans l'évaluation des schémas de tatouage est urgent.

[6] http://watermarking.unige.ch/Checkmark/

Cette référence commune et précise doit suffire aux chercheurs et industriels qui fourniront alors un tableau de résultats reflétant les performances du schéma qu'ils proposent. Les utilisateurs finaux pourront vérifier alors si leurs attentes sont satisfaites. De leur côté, les chercheurs verront les conséquences d'un changement dans une méthode, et seront à même d'en évaluer la pertinence. Ce protocole expérimental facilitera les développements initiaux d'une solution en identifiant rapidement ses forces et faiblesses. Enfin, les industriels connaîtront le degré de confiance à accorder à un schéma de tatouage.

Nous n'abordons ici que l'évaluation des méthodes de tatouage elles-mêmes. Cependant, le cadre d'utilisation de ces algorithmes dépasse la simple volonté de dissimuler une marque dans un médium. Les systèmes qui les emploient poursuivent des objectifs différents (commerce d'images, de vidéo ou de fichiers audios par exemple) et s'appuient aussi sur d'autres techniques, bien souvent liées à la cryptographie. Dans ce contexte, un protocole, une configuration ou encore un générateur aléatoire peuvent tout autant se révéler vulnérables.

4.5.1 Nécessité d'un tiers de confiance

Les performances d'une méthode de tatouage doivent être évaluées et rendues publiques afin que les utilisateurs sachent quelle approche employer. Plusieurs solutions sont envisageables pour la mise en œuvre de ces tests :

- faire confiance à qui fournit la méthode, ainsi qu'aux résultats qu'elle affiche ;
- réaliser soi-même les tests pour vérifier que la méthode répond à ses propres attentes ;
- laisser un tiers indépendant évaluer la méthode.

Seule la dernière démarche permet d'obtenir un résultat objectif, à la condition impérative que la méthodologie et les outils employés soient eux-mêmes connus de tous. Ainsi, les sources du programme seront accessibles à tous, de même que divers documents, détaillant le protocole expérimental. Cette approche permettra à chacun de reproduire aisément les tests réalisés sur la plateforme publique.

La mise en œuvre d'un tel système soulève de nombreux problèmes. Ainsi, le concepteur d'un algorithme de tatouage doit-il envoyer son programme sous forme de sources ou d'exécutable ? Ou bien l'évaluation est-elle accomplie à distance par le biais de preuves interactives fondées sur des échanges entre le testeur et le concepteur ?

Sur ce dernier modèle, on peut imaginer l'approche suivante, inspirée des preuves à divulgation nulle de connaissance (*zero-knowledge proof* - [Kah96, Sch96, MvOV99] - également présenté dans ce mémoire en B.4.1 page 198). Ces protocoles mettent en scènes un vérificateur V et un prouveur P. Le but de P est de convaincre V qu'il connaît un secret sans révéler la moindre information sur celui-ci. En appliquant ce modèle à l'évaluation de schémas de tatouage, P cherche à convaincre V de la résistance de son algorithme à la transformation f :

1. V envoie à P un médium m ;

2. P marque m, produisant le médium $\tilde{\mathsf{m}}$, qu'il renvoie à V ;
3. V renvoie à P le médium $\mathsf{m}' = \mathsf{f}(\mathsf{m}_\mathsf{b})$, où $\mathsf{b} \in \{0, 1\}$, généré aléatoirement, est tel que $\mathsf{m}_0 = \mathsf{m}$ et $\mathsf{m}_1 = \tilde{\mathsf{m}}$;
4. P annonce s'il réussit ou non à retrouver la marque.

Si P triche, il a une chance sur deux de se faire prendre et une chance sur deux de tromper le vérificateur. Donc, au bout de n essais, la probabilité qu'il triche sans se faire prendre est $1/2^\mathsf{n}$. Malheureusement, si ce protocole fonctionne parfaitement dans les cas d'authentification, ce n'est plus vrai pour l'évaluation. D'une part, V dispose alors du médium initial et de sa version marquée, ce qui peut lui permettre de construire des attaques en choisissant les média appropriés. D'autre part, la majorité des transformations f sont facilement inversibles, ou au moins compensables, même si P ne connaît pas f. En effet, P dispose du médium original m, du médium marqué $\tilde{\mathsf{m}}$ et d'un médium à tester m'. Il lui suffit simplement de comparer m' à m et $\tilde{\mathsf{m}}$ pour obtenir une bonne idée de f^{-1}, et ensuite rechercher la marque dans $\mathsf{f}^{-1}(\mathsf{m}')$. Ceci démontre, outre l'inadaptation de ce protocole expérimental, que le vérificateur doit au moins disposer du mécanisme d'insertion ou de détection/extraction.

4.5.2 Conditions requises

Une autre difficulté qui se dresse dans l'élaboration d'un outil d'évaluation automatique des méthodes de tatouage provient de la diversité des algorithmes et des média. En effet, une même catégorie de média contient des éléments très différents les uns des autres, aussi bien au niveau du sens (un concert de musique religieuse ou de hard rock) que de la représentation (une même note jouée par des divers instruments ne possède pas le même timbre). Par ailleurs, certains algorithmes sont dédiés à un type d'images ou de fichiers sons précis alors que d'autres se veulent plus généraux. La composition de notre base de média doit donc être variée.

Ainsi, chaque méthode est testée sur un sous-ensemble généré aléatoirement de média. Dans un second temps, l'utilisateur pourra également spécifier une ou plusieurs catégories de média particuliers (images médicales, satellites, de synthèse, et autres), soit au vu des résultats obtenus, soit parce que l'algorithme est dédié à ce type précis de média. De cette constatation, nous avons conclu que notre outil devait être simple et modulaire.

4.5.2.1 Simplicité

Pour être largement accepté, ce service d'évaluation s'appuie sur une interface simple, compatible avec les bibliothèques de marquage déjà existantes. L'utilisateur fournit trois fonctions : la première pour insérer une marque, la deuxième pour la détecter et une dernière permettant de donner des informations sur la méthode utilisée (cf. section 4.6.1). Comme nous l'avons déjà évoqué, les schémas de tatouage ne répondent pas tous aux mêmes besoins, et nous pro-

posons donc des profils d'évaluation adaptés (ensembles de média et de tests). Ces objectifs sont résumés dans la figure 4.10.

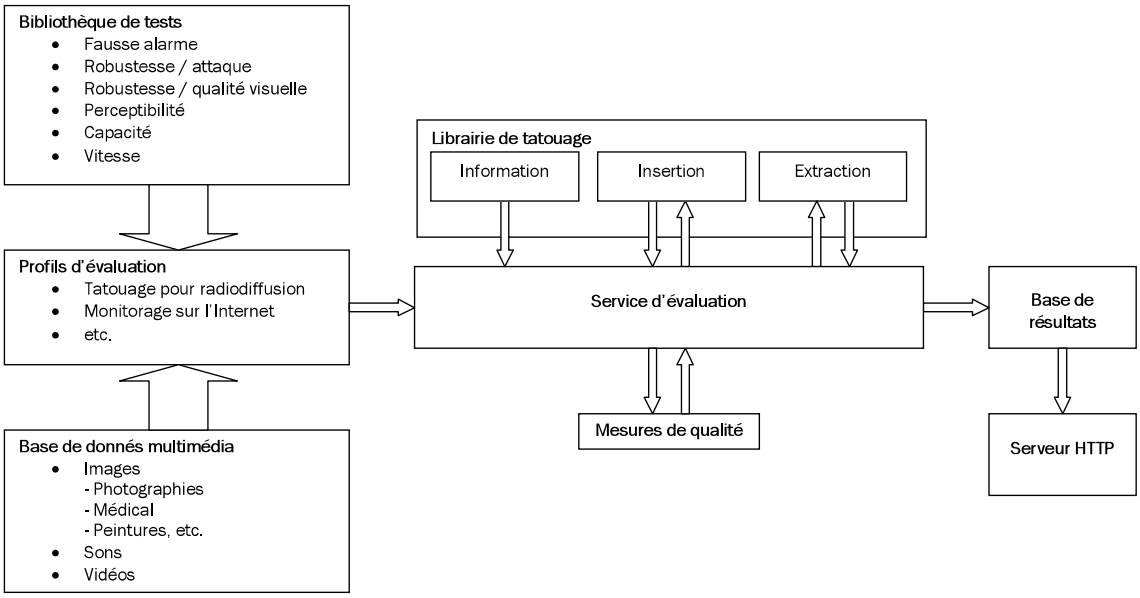


FIGURE 4.10 : Fonctionnement général de `StirMark Benchmark 4`

Le service lui-même repose sur un simple modèle client-serveur : le client envoie une bibliothèque compilée respectant l'interface, puis spécifie le profil d'évaluation souhaité ; un automate déclenche la batterie de tests sélectionnés, et dès qu'ils sont terminés, les résultats sont retournés au client et peuvent être rendus publics, à la demande du client.

4.5.2.2 Modularité

Pour parvenir à un mécanisme pertinent d'évaluation, la première tâche consiste à identifier les différents objectifs poursuivis par les schémas de tatouage :

- *identification de signaux audiovisuels* : la marque transporte un numéro d'identification unique, type ISBN, servant de clé pour une base de données. Ceci permet d'associer à un contenu différentes informations, comme une licence. Cependant, dans certaines occasions, il vaut mieux dissimuler directement les données dans le médium plutôt que sur une base de données centrale afin d'éviter une connexion à un serveur distant ;
- *preuve de propriété* : la marque dissimulée permet à une autorité de connaître l'ayant droit ou le créateur d'un médium ;
- *audit* : la marque contient une information utilisée pour identifier les acteurs en présence lors d'une transaction autour du médium (*i.e.* le distributeur et l'utilisateur final). Cette trace montre le transfert entre les parties.

Les marques qui permettent d'identifier les utilisateurs sont couramment appelées *empreintes* ;

- *contrôle de copies* : la marque dénombre les copies effectuées, ce qui permet d'en contrôler la quantité effectuée. De telles protections sont employées pour prévenir la copie de Digital Versatile Disks (DVD) [BCL+99] ou de fichiers audio (par Sony avec le format ATRAC3) ;
- *contrôle de l'utilisation du médium* : le médium comporte en guise de marque une sorte de numéro de licence. En parallèle, un automate vérifie sur la toile que les utilisateurs sont bien valides ;
- *preuve d'altération* : certaines marques permettent de détecter les modifications subies ultérieurement par le médium ;
- *avertissement aux utilisateurs* : ce type de marque prévient l'utilisateur d'un médium de son copyright. Par exemple, lorsqu'une personne cherche à sauvegarder le médium protégé, un message d'avertissement est alors affiché.

De nombreuses autres utilisations illustrent encore la diversité des attentes. Dans ces conditions, il est indispensable que l'outil d'évaluation soit compatible avec ces applications. Cela passe par la définition de profils, répondant à chaque objectif recherché. Cette tâche délicate nécessite l'agrément de la communauté scientifique. Les profils sont proposés à titre indicatif et peuvent être affinés soit *a priori*, soit *a posteriori* au vu des résultats obtenus lors d'un précédent test.

Tester l'intégralité de tels systèmes dépasse largement le cadre de notre outil : nous ne nous intéressons ici qu'à l'aspect tatouage. De ce fait, les principaux critères à évaluer sont la perceptibilité, la fiabilité (robustesse et fausses alarmes), la capacité ou encore la rapidité. De plus amples détails sont présentés au paragraphe 4.7, tant sur le sens de ces termes que sur les solutions mises en œuvre pour les quantifier. Chaque paramètre est plus ou moins corrélé aux autres. Des bibliothèques spécifiques dédiées à leur évaluation reposent sur des tests *ad hoc*, comme ceux décrits dans [KP99b].

4.6 Architecture de `StirMark Benchmark 4`

4.6.1 Interface

Pour se servir de `StirMark Benchmark 4`, un utilisateur soumet sa méthode de tatouage sous forme de bibliothèque compilée, soit pour Windows, soit pour Linux, exportant trois fonctions.

La première, `GetSchemeInfo()`, fournit des informations sur la méthode, comme sa catégorie (privé, aveugle, asymétrique), son type (I ou II), son auteur, sa version, sa date de mise en disponibilité, une description.

Les deux autres fonctions sont `Embed()` et sa complémentaire `Extract()`. Elles utilisent un ensemble de paramètres, certains obligatoires et d'autres optionnels selon le type de tatouage. Ils comprennent le médium original, la marque, la clé d'insertion, la *force* d'incrustation, la tolérance maximale autorisée, etc. Cette approche permet de conserver une compatibilité entre les différentes mé-

force	0.1	0.5	1	10	30	50	70	80	90	100
PSNR	80	66.02	60	40	30.46	27.96	23.1	21.94	20.92	20

TABLEAU 4.4 : Correspondance entre force et PSNR

thodes de tatouage.

La *force* représente un compromis entre l'imperceptibilité, la capacité et la fiabilité procurées par l'algorithme. Pour le moment, elle est comparable au PSNR. Dans le cas d'une image I comportant N pixels et de sa version tatouée $\tilde{I}$, on a :

$$\begin{aligned} MSE &= \frac{1}{N}\sum_{i,j} |I(i,j) - \tilde{I}(i,j)|^2 \\ PSNR &= 20\log_{10}\frac{b}{\sqrt{MSE}} \end{aligned}$$

où b est la valeur maximale d'un pixel.

La force vérifie les propriétés suivantes :

- il s'agit d'un réel (`float`) compris entre 0 et 100 ;
- plus la force est élevée, plus la qualité de l'image en sortie se dégrade, mais, si tout va bien, la robustesse s'accroît d'autant ;
- une force nulle correspond à une absence de marque (PSNR tendant vers ∞) ;
- une force de 100 implique un médium marqué avec un PSNR proche de 20dB ;
- la distribution de la force doit être harmonieuse. Le PSNR espéré devrait être proportionnel à $20\log_{10}(\frac{1000}{\text{force}})$[7], autrement dit la force est proportionnelle à l'énergie du bruit. Le tableau 4.4 fournit la correspondance entre force et PSNR.

4.6.2 Profils

Le support de différentes applications de tatouage est obtenu grâce à une initialisation dépendant du profil spécifié ([Pet00]). Un profil est construit autour d'un ensemble de tests, avec les paramètres appropriés. Le tableau 4.5 illustre ce mécanisme pour un schéma applicable à la diffusion radiophonique, et un autre correspondant à une vérification pour des images.

Dans un profil, l'utilisateur définit d'abord les tests auxquels son schéma doit être soumis. Ensuite, pour chaque test prévu, les paramètres sont précisés. Des paramètres par défaut sont offerts dans les classes de bases mais il est très facile pour le programmeur d'un nouveau test de spécifier les siens. Par exemple,

[7] Dans [BSC01], les auteurs fournissent une estimation de la déformation subie par l'image après l'insertion de la marque dans un schéma utilisant une DCT : $PSNR = 20\log_{10} 255 - 10\log_{10}(\sigma_I^2 - \sigma_\mu^2) - 20\log_{10}\alpha$, où σ_I^2 représente la variance de la luminance de l'image I, σ_μ^2 la variance de la moyenne de la luminance, et α la *force* de l'inscrustation.

Marquage audio aveugle	**Marquage d'image**
[Test list]	[Test list]
Test 1=Mean embedding time	Test 1=Mean embedding time
Test 2=Mean extraction time	Test 2=Noise addition
Test 3=Sound Low pass filter	Test 3=Image JPEG compression
[Mean embedding time]	[Mean embedding time]
Number of tests=100	Number of tests=100000
[Mean extraction time]	[Noise addition]
Number of tests=100000	Noise start level=0.25
	Noise end level=0.75
[Sound Low pass filter]	Step=0.05
Cut frequency=2000	
	[JPEG compression]
[Samples]	Quality start=100
Set 1=Radio broadcasts	Quality end=75
Set 2=Voices	Step=5
Set 3=Songs	
	[Samples]
	Set 1=Medical pictures
	Set 2=Photographs

TABLEAU 4.5: Exemples simplifiés de profils d'évaluation : chaque profil est composé d'un ensemble de tests avec leurs paramètres et d'une liste d'ensembles d'échantillons.

pour la compression JPEG, on donne la qualité minimale, la qualité maximale et le pas. Enfin, il est possible de préciser des ensembles de média.

4.6.3 Arborescence de classes

Le programme est écrit en `C++` afin de bénéficier des avantages de l'héritage et du polymorphisme de classes. Notre ambition est de fournir un outil permettant d'évaluer des schémas indistinctement pour les différents média (images, sons, vidéos). La figure 4.11 présente une version simplifiée de l'arborescence de classes.

`CBench` est une classe générale pour toutes les évaluations possibles. Elle initialise une liste de tests selon le profil désiré. Elle utilise la classe `CMarkingScheme` comme interface entre la bibliothèque fournie par l'utilisateur et les tests. La classe `CMedium` gère les données audiovisuelles, et en particulier l'allocation de mémoire. `CTest` prend une liste de média et un schéma de marquage en entrée. Un test correspond à tout ce qui peut être évalué dans une méthode de tatouage, comme le taux de fausses alarmes, le temps d'insertion ou de détection, la robus-

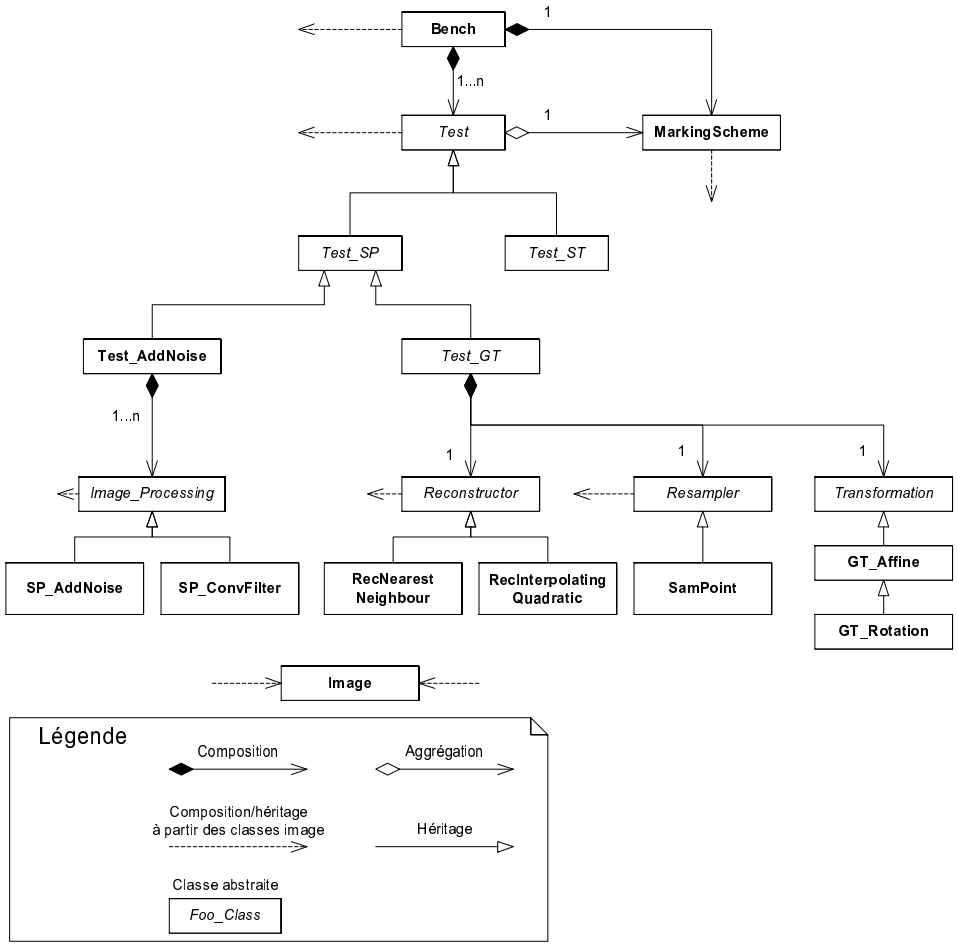


FIGURE 4.11: Schéma UML simplifié de l'arborescence des classes de `StirMark Benchmark 4`

tesse. Enfin, la classe `CMediumTransformation` représente une transformation quelconque applicable à un médium. Par exemple, pour des images, cela comprend des opérations de filtrage, des transformations géométriques. Un `CTest` fait donc couramment appel à une succession de `CMediumTransformation`, bien que ce ne soit pas systématique, comme nous le verrons avec le test pour les faux positifs (voir paragraphe 4.7.2.3).

Bien que les méthodes de tatouage changent en fonction des média, beaucoup de tests demeurent applicables à plusieurs catégories. Ainsi, un test de robustesse se déroule de la manière suivante :

– pour chaque médium dans un ensemble donné

1. dissimuler la marque de sorte que la qualité[8] du médium résultant

[8]Actuellement, le PSNR est utilisé en guise de mesure. Néanmoins, la modularité de `Stirmark Benchmark 4` permet d'y substituer n'importe quelle autre mesure.

soit supérieure à un minimum donné ;

2. appliquer une série de transformations sur le médium marqué.

– pour chaque médium marqué et modifié, tenter d'extraire la marque et mesurer le taux d'erreur.

La mesure de robustesse est donnée par la probabilité de détection ou le taux d'erreur sur les bits après extraction. La procédure décrite est paramétrable par un profil. De plus, elle se doit d'être répétée plusieurs fois car un test pourrait réussir par chance ou échouer par malchance. Il apparaît clairement que ce test ne dépend nullement du médium employé : la classe `CMedium` encapsule les média supportés. Cependant, les transformations sous-jacentes doivent, elles, être définies pour chaque type de médium. Par exemple, un filtre Gaussien ne se programme pas de la même manière selon qu'on souhaite l'appliquer à une image ou à du son. Chaque transformation nécessite donc une spécialisation en fonction du médium visé.

L'utilisation d'une structure orientée objet simplifie l'emploi du logiciel. En effet, il est alors très simple de rajouter une nouvelle attaque. Par exemple, dans le cas d'une image, il suffit de dériver `CMediumTransformation` pour décrire comment change la valeur d'un pixel. Il en va de même pour ajouter un test. Les tâches «administratives», comme lire le médium à marquer, appliquer la méthode de tatouage ou sauvegarder les résultats, sont prises en charge par `StirMark Benchmark 4`. Ainsi, un utilisateur n'a qu'à se concentrer sur le code approprié pour les attaques et tests, sans se soucier des problèmes annexes. L'application propose une grande variété de résultats et graphiques, comme ceux introduits dans [KP99b].

4.7 Critères d'évaluation

4.7.1 Mesures de performances

L'évaluation complète d'un schéma de tatouage nécessite de définir précisément, pour chaque caractéristique (imperceptibilité, fiabilité, capacité, rapidité), un niveau d'assurance souhaité qui correspond à un ensemble de contraintes. Pour qu'un niveau soit validé, il faut que la méthode de tatouage les vérifie toutes. Ainsi, des résultats clairs sont fournis en donnant, pour chaque critère, le niveau d'assurance obtenu lors de l'évaluation.

Les niveaux d'assurance s'étalonnent difficilement. Si on en définit trop, l'évaluation devient très complexe et la lisibilité des résultats y perd grandement. Au contraire, peu de niveaux n'offrent pas une granularité suffisante pour différencier les méthodes de tatouage. Les mécanismes d'évaluation des systèmes de sécurité informatique définissent généralement de cinq à sept niveaux. Cette fourchette semble offrir un compromis raisonnable.

4.7.1.1 Perceptibilité

Le problème est similaire à celui de l'évaluation des algorithmes de compression. Tout comme dans le cas du tatouage, le but est de quantifier les modifications subies par le médium lors d'une transformation. On ne cherche toutefois pas ici à mesurer la perceptibilité de la marque, mais les conséquences de son insertion dans le médium. Cette question (*i.e.* celle de l'évaluation objective *vs.* subjective) est analogue à celle déjà étudiée dans le cadre du codage de source avec pertes (voir [Wat87, JJS93, Com95, KW97]).

Le manque d'une mesure qualitative performante est flagrant. Le PSNR, calculé entre le médium vierge et celui marqué, est trop restrictif dans notre cas. Cette mesure ne prend pas en compte le système visuel/auditif humain. Par exemple, pour les images, tous les schémas fondés sur de légères distorsions géométriques, bien souvent imperceptibles à l'œil, seraient automatiquement bannis.

Un niveau d'assurance bas correspondrait à accepter un médium légèrement dégradé, d'un point de vue perceptif. Typiquement, des diffusions audio *via* un réseau présentent déjà ce niveau de qualité. Un niveau moyen rendrait les modifications du médium imperceptibles dans des conditions d'utilisation «grand public». Le niveau suivant correspond à une amélioration de l'environnement, comme un enregistrement studio pour du son. Enfin, idéalement, la perceptibilité devrait être mesurée à l'aide d'un groupe d'observateurs examinant attentivement le médium marqué dans des conditions très strictes. Ces différents niveaux sont présentés dans le tableau 4.6.

Niveau d'assurance	**Critères**
Bas	– PSNR (lorsqu'applicable) – Légèrement perceptible, mais pas trop gênant
Modéré	– Mesure fondée sur le système visuel humain – imperceptible dans des conditions normales d'utilisation, *i.e.* grand public
Haut	– Différences imperceptibles lors d'une comparaison avec le médium original dans des conditions type studio
Extrême	– Évaluation par un large panel de personnes dans des conditions strictes

TABLEAU 4.6 : Niveaux d'assurance possibles pour la perceptibilité

Il apparaît difficile de mettre au point des niveaux d'assurance fiables tant que nous ne disposons pas d'une mesure précise. Les conditions d'utilisations permettent alors d'introduire une distinction, mais l'automatisation du procédé devient délicate.

4.7.1.2 Fiabilité

Deux critères permettent de définir la fiabilité d'un système, la *robustesse* et le taux de *fausses alarmes* (voir 4.7.2.3), dont il existe deux types :

- les *faux positifs* : la détection d'une marque dans un stégo-médium est positive alors que celui-ci ne contient pas la marque recherchée (ou pas de marque du tout) ;
- les *faux négatifs* : la détection d'une marque dans un stégo-médium échoue alors qu'elle y est bien présente.

Robustesse

Les menaces centrées sur une modification du signal relèvent de la robustesse. Ce terme, selon l'application, revêt des sens différents. Par exemple, dans le cas de la protection des droits d'auteur, il est synonyme de résistance aux attaques qui invalident la marque, soit en la retirant, soit en la rendant illisible. De légères altérations du médium ne doivent pas porter à conséquence. En revanche, lorsqu'il s'agit de vérifier l'intégrité du médium, une autre granularité est recherchée ([WL98, RD00, LC00]). Certaines applications nécessitent la détection immédiate de la moindre modification. D'autres supportent les changements tant que ceux-ci ne dénaturent pas l'interprétation d'un document (e.g. suppression d'un personnage d'une image).

La robustesse peut se mesurer en donnant la probabilité de détection de la marque (ou son taux d'erreur) pour un ensemble de critères appropriés à l'application considérée (cf. 4.6.3). Ainsi, pour chaque transformation applicable au médium, on en augmente la puissance, en fonction du niveau d'assurance souhaité. Le tableau 4.7 page ci-contre présente un exemple des conditions minimales à satisfaire afin d'obtenir l'accréditation du niveau considéré. Un schéma qui veut obtenir le niveau «modéré» doit fonctionner pour des images compressées jusqu'à 50% de leur taille initiale, ou auxquelles on aura appliqué un filtre médian 3×3.

Le niveau *zéro* ne fournit pratiquement aucune protection supplémentaire. Il correspond aux contraintes imposées par les objectifs de l'algorithme et son environnement. Ainsi, dans le cadre de diffusion radiophonique, cela signifie que la marque résiste à l'émission puis la réception du signal audio.

Le niveau *bas* doit empêcher des utilisateurs «honnêtes» d'altérer la marque dans des conditions normales d'utilisation du médium. L'invalidation de la marque nécessite peu de moyens. Pour des photographies, cela signifie que des opérations de compression, de redimensionnement ou de recadrage de l'image ne retirent pas la preuve de propriété.

Le niveau *modéré* est atteint lorsque des outils plus complexes sont indispensables pour compromettre le schéma. En reprenant l'exemple des photographies, l'emploi d'un outil avancé de traitement d'images ne permettrait pas systématiquement d'altérer la marque.

Le niveau *haut* requiert, en plus d'outils spécifiques, une bonne connaissance

	Niveaux				
	Zéro	Bas	Modéré	Haut	Extrême
Qualité de compression JPEG	100–90	100–75	100–50	100–25	100–5
Réduction de couleurs (GIF)	256	128	64	32	16
Cadrage	100–90%	100–75%	100–50%	100–25%	100–5%
Correction Gamma		0.7–1.2	0.5–1.5	0.3–1.8	-
Changement d'échelle		1/2–3/2	1/3–2	1/4–4	-
Rotation		±0–2°	±0–5°, 90°	±0–7°, 180°	-
Symétrie horizontale		•	•	•	-
Bruit uniforme		1–5%	1–15%	1–25%	-
Contraste		±0–10%	±0–25%	±0–40%	-
Luminosité		±0–10%	±0–25%	±0–40%	-
Filtre médian			3×3	3×3	-

TABLEAU 4.7 : Niveaux d'assurance possibles pour la robustesse

et des compétences dans le domaine. Toutes les tentatives pour invalider la marque ne réussissent pas systématiquement, plusieurs tentatives et un peu de travail sur l'approche sont nécessaires.

Le niveau *extrême* implique la mise en œuvre de moyens démesurés, comme des recherches par un groupe de spécialistes, qui rendent le coût de l'attaque bien plus élevé que celui requis pour l'obtention du même médium marqué. Le tableau 4.7 ne fournit pas de détails sur les valeurs requises car ce niveau correspond à une résistance pratiquement parfaite pour chaque attaque.

La robustesse est *démontrable* lorsqu'il est calculatoirement irréalisable pour un adversaire d'invalider la marque.

4.7.1.3 Capacité

La quantité d'information dissimulable dans le médium résulte souvent d'un compromis entre la robustesse et l'imperceptibilité de la marque. Cette valeur est souvent fixée, plus ou moins arbitrairement.

Lors de la mise en œuvre d'un schéma de tatouage, il est très utile d'avoir une idée précise de ces compromis. Un graphique faisant varier deux contraintes, la troisième restant constante, est un moyen simple d'y parvenir. Ainsi dans le modèle simple de tatouage à trois paramètres, on peut étudier la relation entre la robustesse et la force de l'attaque lorsque la qualité du médium tatoué est fixée, entre la force de l'attaque et la qualité du médium tatoué, ou encore entre la robustesse et la qualité [KP99b].

Le premier graphique évoqué ci-dessus est certainement le plus important. Pour une attaque donnée et une qualité de médium après tatouage donnée, il montre le taux d'erreurs en fonction de la force de l'attaque. Le deuxième graphique est utile pour l'utilisateur : la performance du système est fixée (par exemple on pourrait imposer qu'au plus 5% des bits d'information véhiculés

soient corrompus afin de pouvoir appliquer des techniques de correction d'erreur) et il permet de trouver à quels types d'attaques le système peut résister si l'utilisateur peut s'accommoder de telle ou telle perte de qualité. Ces graphiques seront réalisés automatiquement lors de l'évaluation du schéma.

4.7.1.4 Rapidité

Notre outil ne traite que l'implantation logicielle des schémas de tatouage. En général, le temps d'exécution n'est pas un critère très fiable de performances[9], sauf si ces exécutions se déroulent toutes dans un environnement identique. Or `StirMark Benchmark 4` offre justement ce cadre. Tous les tests sont entrepris sur une même plate-forme et une seule méthode est évaluée à la fois. Il est évident que cette mesure n'est pertinente que dans un but de comparaison entre les schémas, l'utilisateur final possédant un matériel potentiellement différent de celui employé pour les tests.

4.7.2 Les tests

La version précédente de `StirMark` proposait essentiellement des transformations géométriques. Toutes les opérations présentes dans ce logiciel sont intégrées dans la nouvelle version de `StirMark Benchmark`.

Les tests sont classés en différentes catégories. Ceux actuellement disponibles sont :

- les tests «géométriques» : transformation affine de l'image, redimensionnement (*scaling*) horizontal et/ou vertical, rotation, distorsions géométriques locales aléatoires, permutation de lignes ;
- les tests «traitement du signal» : ajout de bruit, filtrages, modification d'histogramme, compression JPEG, filtre médian ;
- les tests «spéciaux» : découpage (*cropping*) et retrait de lignes ;

Nous commençons par introduire l'adaptation de nos travaux pour les fichiers son. En effet, notre démarche étant indépendante du type du médium, nous présentons le portage qui en a été réalisé. Ensuite, une évaluation précise ne se limitant pas, comme nous l'avons vu, à éprouver la robustesse du schéma, nous détaillons quelques tests plus généraux concernant les clés, les fausses alarmes, le marquage multiple et le fingerprinting.

4.7.2.1 L'audio

La précédente version de `StirMark` se préoccupait uniquement des images. Celle-ci s'intéresse également aux fichiers audios. En fait, la structure que nous avons initialement adoptée permet volontairement une adaptation simple à n'importe quel type de médium, comme l'expérience l'a montré. En effet, nous avions déjà mis en place la structure générale de `StirMark Benchmark` et commencé à

[9] D'autant plus que le temps d'exécution d'une technologie sous une version logicielle n'est pas forcément un bon indicateur de son potentiel pour une implantation matérielle (si l'application visée le nécessite).

l'appliquer à des images lorsque Martin Steinebach et Jana Dittman nous ont proposé de développer la partie audio.

N'importe quelle manipulation d'un fichier audio constitue potentiellement une attaque à l'encontre de la marque dissimulée. En fonction de la manière dont le son est utilisé certaines attaques sont plus probables que d'autres. Nous proposons d'utiliser le type de post-production en studio comme référence pour ces attaques. Par exemple, la préparation d'un enregistrement sonore pour une retransmission radiophonique inclut souvent la normalisation et la compression de l'enregistrement, afin d'obtenir un niveau de volume sonore compatible avec la transmission, une équalisation pour optimiser la qualité perçue, un débruiteur et différents filtres permettant de retirer les fréquences non transmissibles.

Cette partie repose sur le modèle que nous avons initialement proposé pour les images, mais il est intégralement développé par Martin Steinebach et Jana Dittman de l'Université de Darmstadt ([SPR+01]).

Les attaques audio

Pour rendre possible l'évaluation de méthodes de marquage audio, nous avons constitué des groupes d'attaques. Comme des attaques d'un même groupe reposent sur des principes identiques, il est probable qu'un schéma qui résiste à une attaque donnée dans un groupe résiste également à toutes les autres appartenant à ce même groupe (et inversement si l'algorithme y est vulnérable). Accroître la résistance à une attaque augmente donc la robustesse de la méthode à toutes les attaques du groupe considéré.

Nous avons donc identifié les groupes suivants :

- *dynamique* : porte sur le profil d'amplitude d'un fichier audio. Son augmentation ou sa diminution constitue en soi une attaque. Un limiteur, un expanseur ou un compresseur sont des systèmes plus complexes car ils reposent sur des changements non linéaires dépendant du matériel ;
- *filtrage* : il coupe ou amplifie certaines parties du spectre. Les plus classiques sont les passe-bas et passe-haut, mais les *equalizers* sont aussi assimilables à des filtres ;
- *ambiance* : ces effets simulent la présence d'un endroit (stade, studio, salle de concert, ou autres). Les effets les plus classiques sont la réverbération (*reverb*) et le retard (*delay*), leur paramétrage permettant différentes simulations ;
- *conversion* : le matériel audio est souvent soumis à des changements de format. Par exemple, les sons mono sont dupliqués pour donner du stéréo, la fréquence d'échantillonnage peut passer de 32kHz à 44kHz (voire 96kHz), les conversions numériques/analogiques et analogiques/numériques. Toutes ces conversions impliquent du bruit et des artifices ;
- *compression avec pertes* : ces algorithmes s'appuient sur un modèle psychoacoustique précis et permettent ainsi de réduire la taille des données d'un facteur 10 ou mieux. Ils s'appuient sur une destruction des informations imperceptibles par un auditeur ;

- *bruit* : la plupart des attaques présentées précédemment introduisent du bruit dans le signal. Les composants matériels dans une chaîne audio injectent eux aussi du bruit dans le signal. Une attaque consiste alors à dégrader le signal en y ajoutant volontairement du bruit ;
- *modulation* : les effets de modulation, comme le vibrato, le chorus, la modulation d'amplitude ou le flanging[10] sont rarement accessibles dans des conditions normales d'utilisation du fichier audio. Cependant, comme la plupart des logiciels les incluent, ils peuvent être utilisés comme attaques ;
- *time stretch* et *pitch shift* : ces opérations changent la durée d'un événement audio, sans en modifier la hauteur, ou bien changent la hauteur en laissant la durée intacte. Ils permettent d'obtenir un accordage précis ou de faire rentrer un signal donné dans une fenêtre temporelle ;
- *permutations d'échantillons* : ce groupe comprend des manipulations qui n'apparaissent jamais dans un environnement usuel. Entre autres, il s'agit de permuter des échantillons, ou d'en abandonner ;

Les premiers résultats présentés dans [SPR+01] montrent que les effets d'une attaque dépendent très fortement de la cible. La même attaque est imperceptible sur un morceau alors que ses conséquences sont audibles sur le suivant.

4.7.2.2 Espace des clés

Considérons un médium marqué à l'aide de la clé k. Le programme de détection/extraction ne doit répondre que ce médium est bien marqué que lorsqu'il utilise cette clé k. Dans tous les autres cas, il doit répondre par la négative. Le nombre total de clés constitue également un facteur important puisqu'un attaquant ne doit pas pouvoir essayer toutes les clés pour déterminer la seule valide. Il est donc vital que l'espace des clés soit de grande taille (au moins 2^{64} éléments). Cela signifie que la clé doit au moins comporter 64 bits sans contrainte (bits d'information). En effet, certaines méthodes fixent des bits, pour des raisons de robustesse, d'imperceptibilité, mais ceux-ci ne doivent alors plus compter dans les degrés de liberté de la clé.

Malheureusement, cela ne suffit pas. En effet, deux clés différentes peuvent produire des interférences entre les marques et fausser ainsi la détection. De ce fait, l'espace des clés est beaucoup plus petit qu'il ne semble. Ainsi, il est important de tester la détection d'une même marque, mais en utilisant des clés différentes pour tenter de relever d'éventuelles interférences. On peut choisir des clés plus ou moins proches de la clé authentique, relativement à la distance de Hamming, pour réaliser ces expériences.

Bœuf et Stern présentent dans [BS01] une faille liée à l'utilisation d'une clé identique pour marquer différents média. L'attaque repose sur une analyse de corrélation de la marque incrustée afin d'en déterminer le profil. Une fois celui-ci connu, il est retiré de chaque médium afin de faire échouer la détection. L'analyse est poursuivie dans [ST01] pour les techniques par étalement de spectre où la

[10] Le flanging est créé en mixant un signal avec une copie de lui-même, légèrement «retardé» ; ce retard change constamment.

nécessité d'employer des clés décorrélées est démontrée.

4.7.2.3 Fausses alarmes

Deux situations conduisent à une erreur de type *faux positif* :

1. l'algorithme de détection/extraction découvre une marque dans un médium n'en contenant pas ;
2. l'algorithme de détection/extraction découvre une marque m' dans un médium contenant en fait la marque m.

Le premier cas se mesure en considérant un ensemble de média puis en tentant d'y retrouver une même marque. Cette opération se doit d'être répétée plusieurs fois avec des marques différentes.

Pour le second, nous tentons de détecter une marque m' dans le médium tatoué avec m, en utilisant la même clé que celle employée pour dissimuler m. Comme la marque est un mot binaire, nous pouvons tester un nombre significatif de mots binaires situés à une distance donnée de la marque insérée m, en accroissant cette distance au fur et à mesure. Le résultat du test est présenté sous forme d'un vecteur où la composante i représente le taux de faux positifs découverts pour un mot se trouvant à une distance i de m.

4.7.2.4 Marquage multiple

Il est essentiel de connaître les réactions d'un algorithme à de multiples insertions de marques différentes. Dans une telle situation, soit plusieurs marques sont alors détectables, soit aucune ne l'est ([MB99]). Si de multiples marques sont exploitables, qu'est-ce qui permet de distinguer celle qui est légitime ? Si aucune ne l'est, ceci démontre que la méthode n'est pas assez résistante au tatouage multiple.

Puisque les marques ont des aspects très différents, il est délicat d'estimer l'impact réel de tatouages multiples. En utilisant la distance de Hamming sur l'espace de toutes les marques binaires possibles, étant donnée une marque de référence, on insère, dans le même médium, cette marque de référence et une seconde dont la distance de Hamming à la marque de référence croît. Dans ce type d'attaque, on considère que la première marque est la seule légitime. En effet, cette opération a pour but d'invalider un médium déjà marqué.

4.7.2.5 Fingerprinting

Dissimuler une marque, propre un à utilisateur, relève du *fingerprinting*. Comme les empreintes employées sont différentes pour chaque personne, un attaquant peut comparer plusieurs copies d'un même médium afin de trouver les emplacements où le signal diffère dans la perspective de modifier ou retirer l'empreinte. Ces emplacements délimitent des régions dans l'image, que nous appelons *blocs*. Connaissant ces blocs, un attaquant peut en reconstruire de

nouveaux à l'aide de ceux présents dans chaque copie. Il existe une grande variété de reconstructions :

- remplacer un bloc différent par sa moyenne ;
- remplacer un bloc différent par la moyenne de tous les blocs au même emplacement ;
- remplacer un bloc différent par celui qui revient le plus fréquemment ;
- remplacer un bloc différent par celui qui revient le moins fréquemment.

De nombreuses autres possibilités existent encore.

La figure 4.12 illustre les deux premières attaques sur des images. La figure présente des images différentes provenant de deux clients, ainsi que l'image des différences.

FIGURE 4.12 : Différences de blocs

Dans cet exemple, la technique de fingerprinting employée repose sur une décomposition par DCT en blocs 8×8. Tout d'abord, on insère deux empreintes dans une même image, obtenant alors les images I_1 et I_2. On calcule alors l'image des différences. Pour la première attaque, connaissant maintenant les emplacements des blocs modifiés, on calcule la valeurs moyenne de ces blocs dans les images I_1 et I_2. On remplace les blocs différents par les blocs «moyennés» dans chaque image, ce qui nous donne les images I'_1 et I'_2, dans lesquelles on recherche alors les empreintes respectives. Pour la deuxième attaque, au lieu de remplacer par la moyenne d'un bloc, on substitue la moyenne des deux blocs qui diffèrent dans chaque image.

Une autre approche consiste à utiliser différents média comportant tous la même empreinte. C'est la situation décrite dans [BS01] pour mettre en échec le concours SDMI[11]. L'attaque suppose de disposer d'un original qui permet de calculer des différences fiables. L'auto-corrélation des différences donne suffisamment d'informations pour déduire la structure de l'empreinte et pourvoir la retirer des média. Cette attaque, très intéressante, semble malheureusement difficile à mettre en œuvre de manière automatique.

[11] Le SDMI a soumis à la communauté scientifique quelques morceaux de musique, protégé par différentes méthodes de tatouage, non divulguées. Le but était de parvenir à tromper la fonction de vérification.

4.8 Problèmes ouverts et conclusion

Un problème qui nous préoccupe concerne la soumission du code en elle-même. En effet, un utilisateur peut délibérément envoyer du code malicieux : virus, DoS (*Deny of Service*), cheval de Troie ou autre. Nous réfléchissons à la structure à donner à notre outil pour limiter les risques liés à ces attaques. Dans la lignée d'une attaque moins directe, nous avons également envisagé le cas d'une personne soumettant une méthode de tatouage au nom d'une autre. Deux situations malveillantes peuvent alors apparaître :

1. la bibliothèque soumise sert à s'attribuer l'algorithme d'une autre personne ;
2. la bibliothèque soumise dégrade volontairement ses résultats, afin de nuire à la «réputation» du concepteur initial ;

Ces deux points restent encore sans réponse mais la fréquentation attendue (relativement faible) de ce service laisse espérer la possibilité d'un contrôle manuel.

Nous avons décrit dans cette partie l'architecture générale de `StirMark Benchmark 4`, un outil d'évaluation automatique pour les algorithmes de tatouage, ainsi qu'un ensemble de nouveaux tests destinés à mesurer les performances de la méthode soumise. Cette évaluation repose sur une librairie fournie par l'utilisateur. Il sélectionne un profil, adaptant ainsi les tests aux objectifs poursuivis par la méthode. Les tests effectués permettent d'attribuer un niveau d'assurance pour chaque critère. Un niveau n'est validé que lorsque toutes les conditions requises pour l'atteindre le sont.

Au moment de la rédaction de ce mémoire, de nombreux tests sont déjà programmés et quelques autres sont encore en gestation. Si l'architecture générale du système automatique d'évaluation est élaborée, sa mise en pratique n'en est encore qu'à ses débuts sur le site[12]. Il reste essentiellement deux aspect à améliorer. Tout d'abord, la métrique choisie, le PSNR, ne convient pas à la problématique du tatouage. Elle devra donc être remplacée afin que les tests donnent des informations pertinentes. Enfin, l'autre point à travailler encore concerne la présentation des résultats. Si, comme nous venons de le voir, nous avons précisément défini les mesures à effectuer, nous n'avons encore rien développé, cet aspect ne nous apparaissant pas comme primordial.

[12] http://ms-smb.darmstadt.gmd.de/stirmark

Chapitre 5

Étude des liens entre la cryptographie et la dissimulation d'information

Lors de l'élaboration de `StirMark Benchmark`, nous nous sommes posés avec Caroline Fontaine de nombreuses questions sur les moyens d'adapter efficacement le savoir-faire cryptographique en dissimulation d'information.

En effet, tout le monde s'accorde à dire que des liens existent entre la cryptographie et la dissimulation d'information. Néanmoins, si quelques études s'appuient sur des concepts cryptographiques, aucune n'analyse la pertinence des relations entre ces deux disciplines.

À ce titre, le cas du tatouage est particulièrement démonstratif. Les techniques de tatouage ne sont pas aussi robustes que celles de cryptographie, en particulier car elles ne peuvent fournir une réponse aussi précise que oui ou non à la question «cette marque est-elle présente dans le médium ?». C'est pourquoi la marque sert souvent de canal de communication entre les entités qui interviennent dans le processus.

Dans ce chapitre, nous reprenons les principales notions de cryptographie et montrons en quoi elles peuvent être en rapport avec la dissimulation d'information. Une présentation non exhaustive de ces notions est fournie en annexe B page 189.

Chaque concept est étudié dans l'idée d'être utilisé en dissimulation d'information, que ce soit directement dans l'algorithme (preuve à divulgation nulle de connaissance ou partage de secret) ou bien dans un cadre plus général lié au protocole et à sa mise en œuvre autour d'un schéma quelconque (aspect asymétrique ou signature numérique).

Nous ne proposons pas ici de solutions directement applicables, mais indiquons des directions dans lesquelles prospecter afin d'adapter les solutions efficaces fournies par la cryptographie à la problématique de la dissimulation d'information.

Une première présentation de ces résultats est disponible dans [FR02].

5.1 Espace des clés

Kerckhoffs énonçait au $19^{ème}$ siècle, un résultat qui stipule que la sécurité d'un crypto-système devait reposer sur ses clés. Depuis, les algorithmes sont rendus publics et les attaques se focalisent sur l'obtention de ces dernières (cf. B.2.5). En dissimulation d'information, la question des clés apparaît assez rarement [ST01, PSR+01], mais elle est tout aussi importante : qui contrôle la clé contrôle le médium.

5.1.1 Gestion des clés

Prenons l'exemple d'un artiste qui détient les droits sur ses œuvres mais souhaite en céder une partie. Il possède bien sûr les versions non tatouées, mais des versions contenant sa marque sont déjà diffusées. Que doit-il alors vendre à son acheteur ? S'il s'agit de sa clé, il doit dans ce cas en avoir une par médium afin de ne pas révéler celles des média qu'il conserve. Sa marque ? Si toutes ses créations comportent la même, il s'expose à deux problèmes :

1. des attaquants peuvent se servir de ceci pour déterminer la marque et l'enlever, comme le montrent les résultats du challenge SDMI ([BS01]) ;
2. comment signifier à l'autorité qui régit les droits d'auteur qu'une même marque, selon qu'elle apparaît dans un médium ou un autre, concerne des entités distinctes ?

S'il en existe une différente par médium, il faut alors disposer d'une fonction de vérification capable de tester toutes les marques possibles dans un médium litigieux, ce qui rejoint le problème du «*hachage mou*» présenté par la suite au paragraphe 5.3.2 page 140.

Herrigel *et al.* [HOP+98] proposent un modèle fondé sur la cryptographie à clé publique. Ils insèrent trois marques dans le médium :

- le tatouage privé dont l'expression dépend du médium et la manipulation d'une clé privée ;
- le tatouage de détection, indépendant du médium, détectable à l'aide d'une clé secrète ;
- le tatouage public détectable à l'aide d'une clé publique.

L'ayant droit du médium commence par créer un certificat qu'il enregistre auprès d'une autorité de certification (le tatouage privé). Lorsqu'un client souhaite acheter un médium, une paire clé privé et clé publique est fabriquée pour ajouter les tatouages de détection et public dans le médium. Cette technique repose sur du marquage multiple et suppose donc que les marques sont différenciables.

5.1.2 Attaques exhaustives

En cryptographie, la résistance d'un algorithme est donnée par la complexité de la meilleure attaque connue. Idéalement, lorsque le crypto-système ne donne aucune information, cette attaque est dite *exhaustive*, c'est-à-dire qu'un attaquant n'a d'autre choix que d'essayer des clés au hasard en espérant tomber

sur la bonne[1]. Il est difficile de donner un ordre de grandeur pour cette attaque en dissimulation d'information. Cependant, cette complexité donnerait une idée précise de la fiabilité d'un schéma. Par ailleurs, tout comme en cryptographie, il y a sûrement des *clés faibles* qui peuvent donner lieu à des attaques particulières, plus efficaces.

Dans les schémas de tatouage, la clé sert à sélectionner les composantes qui sont modifiées par la marque, soit en leur ajoutant un bruit[2], soit en leur substituant une autre composante qui vérifie une propriété voulue. Si le choix de ces coefficients résulte d'un tirage uniforme, toutes les combinaisons sont-elles équivalentes ? Pour les méthodes par étalement de spectre appliquées dans les domaines fréquentiels ou multi-échelles, toutes les composantes n'ont pas la même importance par rapport au médium : les coefficients hauts contiennent plus d'information que les bas qui codent les détails.

Ainsi, des tests montrent souvent qu'une clé détecte bien la marque voulue, mais pas les autres. Néanmoins, la réciproque est-elle vérifiée : pour une marque donnée, existe-t-il une clé unique ? S'il existe des collisions, quelles en sont les fréquences ?

En conclusion, depuis que Kerckhoffs a énoncé son principe sur les clés, les cryptographes tiennent essentiellement compte de la clé pour évaluer la sécurité d'un système. En tatouage par exemple, cette préoccupation n'existe malheureusement pas encore dans la mesure où les schémas proposés s'évertuent principalement à gagner en robustesse, cette contrainte étant préalablement nécessaire.

5.2 Chiffrement et attaques

Le chiffrement assure que l'accès à l'information est protégé d'une source non-autorisée (voir B.2 page 190). Nous présentons dans cette partie les conséquences de cette notion par rapport à la dissimulation d'information.

5.2.1 Chiffrement

Pour la stéganographie, le message peut ou non être chiffré. Les systèmes dits purs ne le requièrent pas. Cependant, un message chiffré ressemble à du bruit, ce qui rend le message plus difficile à détecter au travers du canal. Toutefois, si Charlie est un attaquant actif, une simple opération de débruitage sur le stégo-médium suffit à retirer le message. L'attaquant ne peut pas toujours être actif, soit parce qu'il risque alors d'être détecté, soit parce qu'il ne possède pas les moyens (matériels, financiers, ou autres) pour tester tous les média rencontrés. Par exemple, N. Provos et al. [PH02] présentent un système pour tester toutes les

[1] Si la clé est codée sur n bits, alors, statistiquement, il faut tester en moyenne la moitié des clés, soit 2^{n-1} essais et au pire la totalité, *i.e.* 2^n.

[2] Dans ce cas, la même clé sert à la fois à la génération de la marque aléatoire et à la sélection des composantes.

images rencontrées sur Internet afin de contrôler si elles contiennent ou non un message. Après une investigation d'un mois sur quelques deux millions d'images, ils ne sont parvenus à trouver une image cachée dans une seule stégo-image. Celle-ci provenait d'un reportage sur la stéganographie sur le site de la chaîne américaine ABC[3].

Cachin définit la sécurité d'un stégo-système [Cac98] à l'aide de l'entropie relative entre les distributions de probabilités associées aux média de couverture et stégo-média (respectivement notées P_C et P_S) : un stégo-système est dit $\epsilon-$sûr contre un adversaire passif si $E(P_C||P_S) \leq \epsilon$. De plus, lorsque $\epsilon = 0$, le stégo-système est alors qualifié de *parfaitement sûr*.

Ainsi, lorsque le médium de couverture est une chaîne binaire engendrée selon une loi uniforme, le *One Time Pad* (OTP) est un stégo-système parfaitement sûr. En effet, le médium de couverture et le stégo-médium suivent alors la même loi de probabilités. Les données d sont transformées en stégo-médium m par un XOR binaire avec une clé k de même taille que d : $\mathsf{m} = \mathsf{d} \oplus \mathsf{k}$. La chaîne binaire m semble aléatoire et Ève, qui écoute sur le canal, ne peut distinguer cette chaîne m des bits aléatoires qui traversent habituellement. Bob, pour récupérer le message initial, n'a plus qu'à inverser le schéma : $\mathsf{d} = \mathsf{m} \oplus \mathsf{k}$

Cet exemple illustre bien qu'en stéganographie, le médium de recouvrement n'a aucune importance. On retrouve d'ailleurs cette notion de génération du recouvrant dans les schémas de signatures (cf. 5.5 page 150).

Pour le tatouage et le fingerprinting, les données peuvent contenir un message sémantiquement valide. La présence de ces données est généralement connue de tous et les informations servent justement à fournir des renseignements (identification de l'ayant-droit, *smart images* ou autres). Le chiffrement de ces données en interdit la lecture à tous ceux qui ne détiennent pas la bonne clé, ce qui semble contraire au but affiché.

Le chiffrement, pour protéger l'accès à l'information, chiffre celle-ci la rendant incompréhensible à qui ne possède pas le secret nécessaire à son déchiffrement. Néanmoins, une autre notion forte du chiffrement est extrêmement importante en dissimulation d'information : l'autorisation de lire les données. Ces deux principes sont intimement liés en chiffrement : qui détient la clé est supposé être autorisé à accéder aux informations. En revanche, en tatouage ou en fingerprinting, cette hypothèse n'est plus vérifiée dans la mesure où tout le monde peut potentiellement extraire la marque ou l'empreinte. Ainsi, il faut distinguer entre l'aptitude à accéder aux données et celle de les altérer (*i.e.* ajouter, enlever ou modifier).

5.2.2 Chiffrement asymétrique

La cryptographie asymétrique permet à n'importe qui de chiffrer un message à l'aide d'une clé publique tel que seul le détenteur de la clé privée correspondante puisse le déchiffrer. La dissimulation d'information fournit un modèle

[3] http://www.citi.umich.edu/u/provos/stego/abc.html

confortable d'utilisation et de nombreuses études sont disponibles (par exemple [AP98, FD00, SD99, ESB00, FIP01, PA01]).

Nous avons déjà vu dans l'introduction (cf. 2.2.2 page 34) une présentation de la stéganographie asymétrique. Nous n'en rappellerons donc que le principe général ici. Une personne qui souhaite envoyer un message discrètement à Bob utilise sa clé publique. Bob, recevant un médium, le teste avec sa clé privée qui lui révèle le contenu du message. Du point de vue de l'attaquant (actif ou passif) qui écoute le canal, que le schéma soit public ou secret ne change rien : il voit toujours passer des média et n'est pas supposé être capable de distinguer les stégo-média parmi ceux qui transitent.

Comme tout schéma asymétrique, cette stéganographie est vulnérable à l'attaque de l'homme du milieu (cf. tab. 2.3 page 36). Dans le cas où Alice n'a pas connaissance de la clé publique de Bob, ce dernier lui transmet sa clé publique. Wendy, qui surveille ce qui se passe, remarque cette transaction : elle intercepte la clé de Bob et donne à Alice sa propre clé, usurpant ainsi l'identité de Bob. Ensuite, lorsqu'Alice envoie un message à Wendy, qu'elle croit être Bob, cette dernière est à même d'extraire le message puis de l'insérer à nouveau à l'aide de la vraie clé publique de Bob. Alice et Bob ne remarquent pas que les échanges sont interceptés, mais Wendy parvient à tous les obtenir.

Néanmoins, si Wendy n'intervient pas dans l'échange des clés, elle ne doit pas pouvoir distinguer la présence d'une communication cachée dans le canal en ne faisant que l'observer. De même, à moins d'être convenu par ailleurs avec Alice qu'un échange allait avoir lieu, Bob ne peut pas savoir que quelqu'un utilise sa clé publique pour lui transmettre un message secret. En effet, aucun échange préalable n'étant requis, comme en stéganographie pure, Bob doit donc tester tous les média qu'il reçoit pour vérifier s'ils contiennent ou non quelque chose. Il semble donc nécessaire là aussi d'introduire une étape de négociations afin que Bob sache quand chercher un message.

Dans la perspective du tatouage, la clé publique ne doit pas servir à insérer des informations dans le médium : seul l'ayant-droit doit pouvoir réaliser cette opération. Un tel schéma à clé publique, au sens du chiffrement, n'est donc pas du tout souhaitable. À l'inverse, on espère plutôt qu'une personne découvrant un médium soit capable de vérifier s'il contient ou non une marque. Le tatouage semble donc s'apparenter plus aux schémas de signatures (voir paragraphe B.5).

La problématique du fingerprinting n'est pas très éloignée de celle de la cryptographie à clé publique. En effet, lorsque le détenteur d'un médium le distribue, chaque client y imprime alors sa propre empreinte. Cependant, seul le propriétaire doit être capable d'agir sur cette empreinte (la modifier ou la retirer par exemple). Ainsi, à la différence du chiffrement, le fingerprinting nécessite une clé privée mais autant de clés publiques que de clients (cf. fig. 5.1). Dans ce schéma, la clé publique permet de vérifier la présence ou non de l'empreinte correspondante dans le médium.

La configuration inverse est également intéressante pour le fingerprinting (voir fig. 5.2). Si on dispose de plusieurs clés privées auxquelles correspond une unique clé publique, celle-ci permet à tout le monde de vérifier la source du

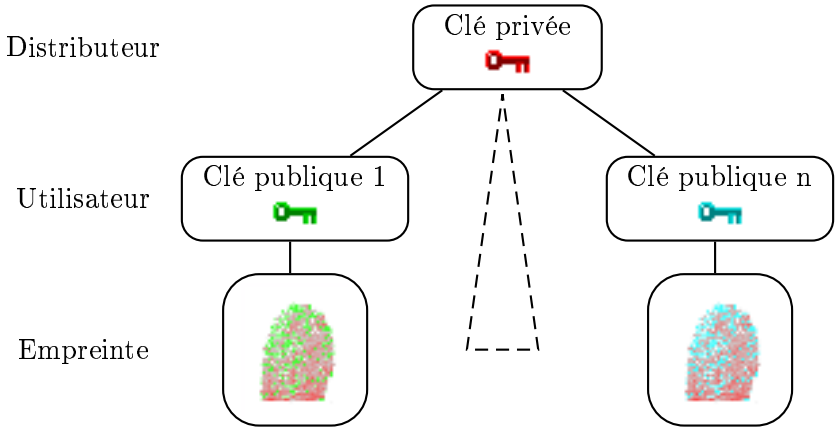


FIGURE 5.1: Le distributeur possède sa propre clé privée, chaque utilisateur une clé publique différente afin que toutes les empreintes ne soient pas identiques.

médium. Par exemple, dans le cas d'un médium distribué par Internet, le détenteur vend les droits d'exploitation à différents sites. À chacun d'eux, il alloue une clé privée. Lorsqu'un des distributeurs diffuse le médium, il imprime alors son empreinte construite à partir de sa clé privée et d'un identifiant pour la transaction. Ultérieurement, si une copie est découverte, la clé publique du propriétaire ne permet pas d'identifier la clé privée utilisée. L'information retirée du stégo-médium doit donc contenir un identifiant de la clé privée utilisée. L'avantage de ce schéma par rapport au précédent est qu'une seule clé est nécessaire pour contrôler l'origine du stégo-médium.

Un tel dispositif serait très utile pour l'industrie cinématographique par exemple. De plus en plus de copies illégales de films sont réalisées à partir d'une caméra numérique directement depuis une salle de cinéma. Si le distributeur du film accompagne la bobine d'une clé privée, l'insertion à chaque projection dans une salle donnée d'une empreinte permet de remonter précisément à la source de la copie.

Ainsi, par rapport au chiffrement à clé publique classique, le fingerprinting impose des contraintes plus fortes : une des clés (la privée ou la publique) doit exister en de multiples «exemplaires» distincts, indépendants les uns des autres. Cette diversité pose également un problème de confiance vis-à-vis des détenteurs dans le cas de collusions.

5.2.3 Les attaques

Dans les différentes attaques évoquées ci-dessous, nous considérons que l'attaquant dispose de toutes les connaissances sur la méthode employée, selon le principe de Kerckhoffs.

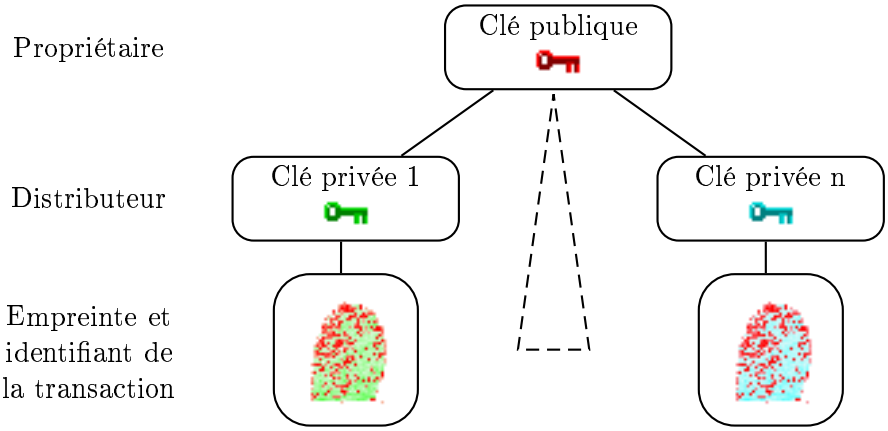


FIGURE 5.2: Une unique clé publique permet de contrôler la provenance d'un stégo-médium

5.2.3.1 Définitions des attaques

Les attaques à l'encontre des schémas de chiffrement positionnent l'attaquant dans des situations différentes, selon les ressources dont il dispose (voir B.2.5). De même, nous définissons plusieurs modèles d'attaques. Celles-ci sont plus nombreuses qu'en cryptographie car trois éléments interviennent : le médium, les données et le stégo-médium. Alors que le médium et le stégo-médium sont respectivement comparables au clair et au chiffré, les données tiennent un rôle particulier :

- *attaque à stégo-média seuls* : seul le stégo-médium est disponible. Lorsqu'une telle attaque est réalisable, le schéma visé ne fournit plus aucune garantie ;
- *attaque à clairs connus* : le médium original et le stégo-médium sont tous les deux nécessaires à l'attaque. Les schémas non *aveugles* sont particulièrement sensibles à ce type d'attaque, le médium initial étant requis lors de la détection, si l'attaquant parvient à en intercepter la transmission ;
- *attaque à clairs choisis* : l'attaquant possède certains média initiaux dans les paires (médium, stégo-médium) dont il dispose. Par exemple, une méthode de tatouage en ligne permet à l'attaquant de soumettre les média qui l'arrangent pour ensuite tenter de casser la méthode ;
- *attaque à stégo-média choisis* : l'attaquant dispose d'un ensemble de stégo-média, parmi lesquels il peut obtenir les originaux de certains.

On relativise la portée de chacune des attaques précédentes en fonction de la connaissance qu'a l'attaquant des données : elles sont soit ignorées, soit connues, soit choisies.

Ainsi, tout comme en cryptographie, les attaques contre les schémas de dissimulation de données se classent selon trois critères :

1. le contexte : position vis-à-vis du médium original et du stégo-médium ;

2. les données : disponibilité de celles-ci pour l'attaquant ;
3. virulence de l'attaque : quels pouvoirs obtient l'attaquant à l'issue de son action.

Dans le cas de la stéganographie, la connaissance ou le choix du message ne doivent avoir aucune influence sur la force de l'attaque. En effet, un attaquant n'est pas censé pouvoir distinguer un médium vierge d'un stégo-médium (voir [Cac98]), ce qui implique que la connaissance des données d n'apporte aucune information supplémentaire :

$$I(m|d) = I(m),$$

où m est le médium et I la quantité d'information. Toutefois, en pratique, les systèmes actuelles ne vérifient pas cette propriété. Westfeld et Pfitzmann [WP00] décrivent des attaques fondées sur une analyse statistique des images et les modifications qu'elles subissent en fonction des schémas de stéganographie utilisés.

La similitude entre un stégo-système parfait et un crypto-système parfait est assez évidente. En cryptographie, ce cas idéal se produit lorsque la connaissance du chiffré c n'apporte pas plus de connaissance sur le message m que m n'en donne sur lui-même, c'est-à-dire quand l'entropie vérifie $H(m|c) = H(m)$.

Shannon a démontré en 1949 [Sha49b] que, dans un crypto-système parfait, le nombre de clés devait être au moins aussi grand que le nombre de messages, mais aucun résultat similaire n'existe pour un stégo-système parfait.

5.2.3.2 Exemples d'attaques

– *attaque à stégo-médium seul* :
En stéganographie, ces attaques surviennent lorsque le médium utilisé pour le transport est créé expressément à cette fin. En effet, il n'existe alors pas de «médium clair» dont peut disposer l'attaquant.

Pour la protection des droits d'auteur et des copies, cette attaque n'est envisageable que si le médium n'existe pas dans une forme vierge de toute incrustation supplémentaire[4].

Les *attaques par collusion* rentrent majoritairement dans cette catégorie. En effet, elles surviennent lorsque plusieurs personnes se liguent afin de façonner un médium qui ne contient plus la protection[5]. Par exemple, dans le cas de morceaux de musique vendus sur Internet, plusieurs acheteurs se regroupent afin de reconstruire le morceau à partir d'extraits appartenant à chacun. La chanson ainsi reconstruite est phonétiquement identique à

[4]En toute rigueur, une telle attaque est également possible si le clair est disponible, mais dans ce cas, on suppose que l'attaquant l'utilise.

[5]Ou tout du moins qui en contient une forme trop altérée pour être reconnue par une autorité.

On découpe ... ...et on recolle

FIGURE 5.3 : Attaque par mosaïques

la version initiale, mais la protection est inutilisable.

En image, l'attaque par mosaïque ([PAK98]) conduit à une situation identique. Elle a pour but de contrer les robots automatiques sur Internet qui recherchent des images pour en vérifier le tatouage. Son principe est de découper une image en plusieurs petits morceaux et de la présenter «recollée» sur une page html.
Une version avec collusion est de rassembler les morceaux non pas issus d'une même image, mais de versions différentes, chacune contenant des données distinctes.
Le fingerprinting est principalement la cible de ce type d'attaque. En effet, comme chaque stégo-médium comporte sa propre empreinte, il existe de nombreuses copies différentes d'un même médium. Si plusieurs utilisateurs se regroupent pour reconstruire un médium à partir des leurs (on parle alors de *coalition*), l'empreinte nouvellement générée doit permettre de retrouver chaque membre de la coalition. Pour permettre ceci, [BS95] et [CEZ99] font appel à la théorie des codes. Par exemple, dans [CEZ99], les auteurs utilisent des codes intersectants ([CZ94, CEL01]) où le «mélange» de plusieurs mots du code conduit à un mot n'appartenant plus au code, mais dont il est possible de retrouver les origines.
Cependant, ces codes sont limités dans la taille maximale où la coalition reste détectable. Une personne supplémentaire au-delà de cette limite permet de contourner la protection.

– *attaque à clair connu* :
Pour la stéganographie, de telles attaques sont possibles lorsque le support est un médium connu. Si Alice dissimule un message dans l'image de Lena, Charlie qui tente de détecter la communication peut alors comparer

cette version de Lena à une «image de référence»[6]. À l'inverse, l'emploi de médium créé pour la circonstance, s'il ne permet pas des attaques à clair connu (puisque ce clair n'est pas disponible) éveille les soupçons de Charlie. En guise de compromis, Alice pourrait utiliser ses photos de vacances prises avec un appareil numérique.

Dans le cas du tatouage ou du fingerprinting, si le médium protégé existe également en version «sans incrustation», alors un attaquant se retrouve dans cette configuration. Par exemple, les œuvres musicales sont protégées cent ans, mais des enregistrements, actuellement disponibles, ne contiennent aucun tatouage ou empreinte. Dans ce cas, il n'y a d'ailleurs pas besoin de monter une attaque puisque se procurer un telle copie non protégée suffit.

- *attaque à clair choisi* : le logiciel photoshop propose une option de marquage des images, fondée sur le produit de la société Digimarc. Un utilisateur peut alors proposer les images initiales de son choix, les tatouer avant de s'attaquer à contrer l'algorithme.

- *attaque à stégo-médium choisi* : ces attaques semblent plus difficiles à trouver puisqu'elles nécessitent l'accès à la fonction de détection.

  Le challenge SDMI correspond néanmoins à cette situation puisque les candidats avaient à leur disposition un oracle testant la présence de la marque dans le médium.

En conclusion, si le chiffrement en lui même ne semble pas quelque chose d'important pour la dissimulation d'information, il n'en est pas de même avec la symétrie des algorithmes de chiffrement. Les besoins sont donc plutôt structurels. En particulier, les attentes du fingerprinting posent de nouveaux problèmes de cryptographie avec l'utilisation d'une clé privée pour plusieurs clés publiques (et réciproquement).

Par ailleurs, les attaques possibles à l'encontre d'un crypto-système constituent un critère important de sa qualité. La définition claire d'attaques envers les schémas de dissimulation d'information apparaît donc aussi comme une étape importante dans la présentation du schéma proposé, en plus de ses performances en termes de capacité, robustesse ou imperceptibilité. Néanmoins, à la différence de ces trois caractéristique, les attaques possibles dépendent essentiellement de la mise en œuvre du schéma et, par conséquent, du protocole qui l'accompagne.

[6]En fait, il existe tellement d'images de Lena que trouver celle qui sert de référence n'est peut-être pas si facile, mais de telles images sont rares. Par conséquent, si elles étaient utilisées pour de la stéganographie, elles seraient immédiatement suspectes.

5.3 Fonctions de hachage et intégrité des données

Les fonctions de hachage (cf. annexe B.3 page 193) interviennent dans différents processus en cryptographie. En particulier, elles servent à contrôler l'intégrité des données.

5.3.1 Hachage et collisions

D'un point de vue structurel, une comparaison entre le hachage et la dissimulation ne semble pas appropriée, essentiellement parce que le hachage ne nécessite qu'une fonction là où la dissimulation d'information en requiert deux.

Néanmoins, pour ce qui est des propriétés, il n'en est pas de même. En effet, une première illustration nous est donné par l'*attaque par recopie*, présentée dans [KVH00]. Elle est ciblée contre les schémas de tatouage agissant dans le domaine spatial, mais devrait être reproductible contre ceux de fingerprinting. Étant donné une image tatouée, cette attaque reproduit la marque dissimulée dans une autre image, sans utiliser d'informations relatives à la méthode de tatouage ou autres, comme la clé secrète par exemple. Elle se déroule en trois étapes :

1. estimation de la marque présente dans l'image tatouée ;
2. adaptation de cette estimation à l'image cible afin de conserver indétectabilité ;
3. insertion dans l'image cible.

Cette attaque illustre ce qui se passe quand un schéma n'est pas faiblement résistant aux collisions. En effet, un attaquant dispose d'un stégo-médium, sans autre information, mais parvient tout de même à construire un autre stégo-médium qui fournit une réponse identique lors de la phase de détection.

Les conséquences de cette attaque sont importantes vis-à-vis des trois types de schéma qui nous intéresse :

1. stéganographie : l'étape d'estimation semble plus difficile à réaliser ici que pour les deux autres car le message doit être indétectable statistiquement. La réussite de cette attaque contre le tatouage repose sur la robustesse de la marque qui permet ainsi de l'estimer plus facilement pour la recopier. Or, cette contrainte n'est pas nécessaire en stéganographie. De plus, si cette attaque réussit, on peut douter que la qualité du signal estimé soit suffisante pour produire un message intelligible ;
2. tatouage : reproduire la marque présente dans un stégo-médium conduit à deux interprétations :
   (a) Charlie veut nuire à Alice en recopiant la marque de cette dernière sur un médium qui conduira à la diffamer. Par exemple, dans le cas de la vidéo *via* Internet, un attaquant pourrait ajouter ou modifier des séquences tatouées avec la marque d'une chaîne de télévision, ce qui peut conduire à changer le sens du reportage diffusé, au point de nuire à l'émetteur original ;

(b) Charlie veut s'approprier un médium sans recourir aux autorités de validation. Il utilise alors un de ses propres stégo-média pour en extraire une estimation de sa marque et l'insère ensuite dans le médium de son choix. Dans ce cas, il ne faut pas que la marque contienne un identifiant de Charlie, sans quoi on prouve qu'il est à l'origine de la copie illégale. Cette attaque sert par exemple contre des lecteurs de fichiers audio ou vidéo qui vérifieraient la présence d'une marque, la lecture du médium dépendant de la présence de la marque dans le médium.

Il apparaît donc clairement qu'une méthode de protection des droits d'auteur qui ne résiste pas à cette attaque ne peut pas être considérée comme fiable ;

3. fingerprinting : les conséquences sont similaires à celles présentées pour le tatouage. Si Charlie souhaite diffuser des copies d'un médium qu'il possède (qui contient donc son empreinte), il peut y ajouter d'autres empreintes afin de faire croire à une collusion et rendre ainsi difficile, voire impossible, la découverte de l'origine de la copie.

Cet exemple illustre l'importance de définir clairement un terme *collision* adapté à la dissimulation d'information. En hachage, une collision apparaît lorsque deux entrées différentes donne le même résultat. En dissimulation d'information, l'entrée n'étant pas unique, nous devons étendre cette notion. Ainsi, nous proposons de dire qu'une *collision en dissimulation d'information* se produit lorsque deux fonctions d'insertion, chacune avec ses propres paramètres, conduisent une même fonction de vérification à une réponse identique.

5.3.2 Intégrité et authentification de message

Qu'il s'agisse de stéganographie, de tatouage ou de fingerprinting, l'intégrité des informations insérées dans le médium est recherchée. Cependant, nous montrons qu'elle n'est plus suffisante dès que le médium doit lui aussi être protégé.

Stéganographie

Dans le cas de la stéganographie, l'intégrité recherchée est celle du message dissimulé. Il n'est en effet pas souhaitable que Charlie puisse agir en attaquant actif et changer le sens du message sans que cela ne soit détecté. Ceci est possible en ajoutant au message un haché du message lui-même. Ce haché doit être associé à un secret, et résulte donc soit d'un schéma de signature, soit d'une fonction de hachage à clé (fonction MAC, voir B.3.1 page 193).

Dans un schéma de stéganographie pure, la sécurité est censée résider uniquement dans le canal de communication. Cependant, l'ajout d'un haché du message au message inséré dans le médium permet au destinataire d'en vérifier l'intégrité. Cette démarche, inutile sur les canaux de communication sans erreur (protocole TCP par exemple) est indispensable dès que le stégo-médium risque

d'être altéré dans le processus de transmission tolérant les erreurs (téléphonie mobile - GSM, UMTS -, protocole UDP).

Dès lors que cette condition est requise, une clé supplémentaire, différente de celle utilisée pendant l'insertion du message, est nécessaire pour signer : elle servira soit pour un MAC, soit pour le chiffrement suivant le MDC. D'après les critères de stéganographie, ce chiffrement est superflu car le message ne doit pas être détectable.

Rajouter la signature aux données propres pour constituer le message ne représente pas une opération très coûteuse. Des fonctions comme RIPE ou SHA-1 fournissent un haché sur 160 bits alors qu'une image comme Lena en 512×512 avec 256 couleurs compressée en jpeg à 75% occupe environ 30Ko.

Toutefois, il faut bien garder à l'esprit que l'ajout ne serait-ce que d'un simple bit supplémentaire dans le stégo-médium accroît les chances de Charlie de détecter la présence d'un message.

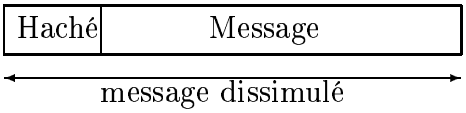


FIGURE 5.4 : Structure d'un message pour de la stéganographie

Tatouage et fingerprinting

Les problématiques du tatouage et du fingerprinting diffèrent de celles de l'intégrité des données par la souplesse requise : un stégo-médium doit être insensible à de légères modifications tant que son aspect est conservé, au contraire des fonctions de hachage qui respectent le principe d'avalanche[7].

Idéalement, la notion de «*hachage mou*» semble préférable. À l'inverse de celles considérées jusqu'ici, ces fonctions de hachage ne respectent pas le principe d'avalanche. Au lieu de changer la sortie au moindre bit modifié, elles fonctionnent par seuil dans un voisinage. Par analogie avec un code correcteur, la correction d'un mot s'effectue en le remplaçant par le mot du code le plus proche. Cependant, au-delà d'une certaine distance, cette correction ne s'effectue plus car elle ne correspond plus à rien puisque le mot contient trop d'erreurs. En tatouage ou en fingerprinting, si on considère le stégo-médium comme étant un mot d'un code, on souhaite que la détection fournisse le même résultat dans tout le voisinage. Comme nous le montrons par la suite, des définitions rigoureuses de voisinage et de distance sont extrêmement délicates à poser dans le cadre de média.

Les média présents dans le voisinage du stégo-médium correspondent à des versions modifiées de celui-ci (compression avec pertes ou autres traitements

[7] Ce principe stipule, pour les fonctions de hachage, que la modification d'un bit du message doit entraîner la modification de la moitié des bits du haché, voir annexe B.3.1.

similaires). Par exemple, plus un médium est éloigné du centre du voisinage, plus il aurait subi de changements par rapport au stégo-médium. Lorsqu'il sort du voisinage, on considère alors que le médium résultant est inutilisable.

Linnartz *et al.* présentent une attaque qui se sert de cette approche dans [LvD98] contre les schémas de tatouage d'images pour lesquelles la fonction de vérification est publique. Disposant d'un oracle[8], l'attaquant lui soumet la stégo-image dont il souhaite retirer la marque. Il en modifie un pixel et soumet ce nouveau médium à la fonction de vérification. Si celle-ci reconnaît encore la marque présente dans la stégo-image, l'attaquant modifie un autre pixel, et ainsi de suite jusqu'à ce que la vérification échoue. La complexité de cette attaque est assez basse, puisque linéairement proportionnelle au nombre de pixels présents dans l'image.

Enfin, signalons que le médium original appartient au voisinage puisqu'il fait partie des média suffisamment proches du stégo-médium pour être encore utilisable. La question se pose alors de savoir quelles mesures prendre pour le protéger lors du parcours exhaustif du voisinage. En effet, il faut d'une part que ce parcours soit «long» pour éviter qu'un attaquant ne teste tous les voisins du stégo-médium afin de retrouver l'original. D'autre part, dans le voisinage, le médium initial doit être le seul à ne pas réagir à la fonction de détection. Néanmoins, dans le cas d'images par exemple, un morphing entre une image marquée et une autre représentation du contenu de cette image permet de parcourir aisément une partie du voisinage.

Ce comportement est désiré pour les applications de tatouage et de fingerprinting, mais il n'est pas sans poser certaines questions :

1. Comment définir un voisinage acceptable ?

   Si on reprend la comparaison avec les codes, le cas du code de Gray est embarrassant. Les mots de ce code sont tous les mots binaires de longueur m, mais ils sont ordonnés de manière à ce que deux mots consécutifs ne diffèrent que sur une seule coordonnée. Par exemple, les mots du code de Gray de longueur 3 sont dans la matrice suivante :

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

   Par construction, le changement d'un unique bit permet de parcourir tous les mots du code.

[8] On considère la fonction de vérification comme une boîte noire qui ne fournit qu'un bit d'information : oui ou non la détection réussit.

Si, pour un médium d'une taille donnée, on considère tous les mots du code de Gray de même taille, on peut alors parcourir l'espace des média en ne changeant qu'un bit entre deux mots du code (*i.e.* deux média). Bien évidemment, tous les mots du code ne correspondent pas à un médium intelligible. Il semble donc nécessaire que deux média soient éloignés au maximum dans cette représentation afin de rendre la recherche exhaustive, comme présentée dans [LvD98], la plus coûteuse possible. L'attaque présentée dans [LvD98] repose justement sur cette notion.

2. Comment distinguer des copies presque identiques mais toutes légales ?

   Prenons l'exemple d'une œuvre musicale. Si un musicien professionnel l'interprète pour enregistrer un CD, il perçoit ensuite des droits liés à cette activité. Si une autre personne, musicien amateur, joue le même morceau d'une manière plus ou moins proche, et le met sur Internet, il ne doit alors pas être inquiété. La fonction de détection doit être suffisamment sensible pour distinguer qu'il s'agit bien de la même mélodie mais que les interprètes sont différents.

   Cette même situation se présente aussi avec les images. Si un photographe est engagé par le musée du Louvre pour photographier la Joconde, un touriste passant au même moment bénéficie de la même lumière et obtient une photo très proche, même si elle sera sans doute de moins bonne qualité, que celle du photographe.

   Dans ces deux cas, les média «amateurs» sont très proches de ceux soumis à un copyright. La notion de voisinage fondée uniquement sur le contenu du médium semble donc trop étroite, d'autant qu'il faut encore différencier les droits pour le compositeur de ceux pour l'interprète.

Il semble donc que le tatouage doive dépendre à la fois d'un secret et du médium dans lequel il sera inséré. Pour cela, il doit vérifier les propriétés suivantes :

- les marques calculées pour deux média distincts avec une même clé doivent être décorrélées ;
- pour deux média similaires[9], les marques calculées avec une même clé doivent être fortement corrélées ;
- les marques calculées pour un même médium, mais avec une clé différente pour chacun, doivent être décorrélées.

J. Fridrich explore cette direction en proposant des fonctions de *hachage robuste* qui permettent de construire des marques vérifiant les propriétés précédentes (voir [Fri00, FG00]). Ces fonctions sont construites de manière à être invariantes à certaines transformations (rotation, changement d'échelle, opérations sur l'intensité des pixels, etc.) Le haché ainsi construit sert alors de marque dans l'image. La solution proposée résiste à des opérations globales sur l'image, mais pas à de légères transformations locales.

[9] Une des difficultés est justement de définir une métrique capable de quantifier cette similarité.

Cependant, comme nous venons de le montrer, cette idée répandue qu'il faut qu'un voisinage du médium (l'initial ou le «stégo») soit acceptable du point de vue de la fonction de détection n'est pas sans soulever de nombreux autres problèmes au niveau du protocole.

En conclusion, on constate que le terme d'*intégrité* pose une énorme problème en dissimulation d'information. S'il est parfaitement défini en cryptographie où la modification du moindre bit est immédiatement suspecte, sa définition en dissimulation d'information demande moins de sensibilité. Cela passe par l'élaboration d'une métrique acceptable afin d'obtenir une comparaison pertinente entre plusieurs média. Néanmoins, les contraintes sur cette métrique sont importantes puisqu'elle doit, entre autre, permettre de distinguer de manière unique le médium initial des autres média.

5.4 Identification et preuve à divulgation nulle de connaissance

L'identification concerne les mécanismes qui permettent de prouver son identité à une entité (cf. B.4 page 197). En particulier, nous avons présenté des protocoles à divulgation nulle de connaissance. Nous commençons donc à nous poser la question des besoins d'identification en dissimulation d'information. Ensuite, nous rappelons les travaux antérieurs appliqués au tatouage et utilisant des protocoles *zero-knowledge*. Enfin, nous analysons les conditions nécessaires à la mise en œuvre de tels protocoles en dissimulation d'information.

5.4.1 Qui suis-je ?

En stéganographie, le message transite par le canal dissimulé dans un stégomédium. Toutefois, la question de l'identification n'apparaît jamais : comment Alice est-elle certaine que c'est bien Bob qui attend à l'autre extrémité du canal ? Comment Bob sait-il que le message a bien été écrit par Alice ? *A priori*, rien n'est jamais prévu pour répondre à ces questions. En fait, dans le cas de la stéganographie pure, la sécurité du canal repose sur la non détection de la méthode utilisée qui doit rester secrète, ce qui va à l'encontre des principes de Kerckhoffs. Donc, comme seuls Alice et Bob la connaissent, il n'est pas besoin d'identifiation. Dans le cas de schéma à clé secrète, un accord préalable permet le choix de la clé. La sécurité réside alors dans la non divulgation de celle-ci : si Charlie parvient à se procurer cette clé, que ce soit en «cassant» l'algorithme ou par un autre moyen, il parvient à se substituer à qui il veut. Enfin, la stéganographie à clé publique, tout comme la cryptographie du même nom, requiert que les clés soient certifiées ou échangées de manière sûre afin de se prévenir des attaques type «homme du milieu».

Dans le cas du tatouage et du fingerprinting, le problème n'est plus le même. Cette fois, le médium contient les données qui permettent d'identifier son origine, qu'il s'agisse de l'ayant-droit ou de l'utilisateur selon le domaine. Néanmoins,

les objectifs des attaques menées dans ces disciplines nécessitent moins de résultats. En effet, lorsqu'un attaquant veut rendre un tatouage ou une empreinte indétectable, dans la majorité des cas, il cherche simplement à tromper une autorité afin que la vérification échoue. Cependant, le remplacement d'une marque ou d'une empreinte par une autre s'apparente exactement à une impersonnification. Pour le tatouage, l'attaquant s'approprie le médium en se substituant à son propriétaire légitime. Pour le fingerprinting, l'identité de l'utilisateur est remplacée par celle d'un autre. La question de l'identification sont donc très importante pour ces deux disciplines.

Dans la suite de cette section, nous ne nous intéressons plus à la stéganographie pour laquelle les problèmes d'identification sont plus proches de ceux rencontrés en cryptographie.

5.4.2 Utilisation d'un protocole à divulgation nulle de connaissance

Le tatouage et le fingerprinting attendent énormément de la mise en place de schémas *zero-knowledge*. En effet, cette approche garantit que le vérifieur n'obtient aucune information sur les données insérées dans le médium. Par exemple, Bob pourrait ainsi mettre sur son site Internet ses propres média, tatoués, et sa clé publique. Toute personne voulant vérifier l'appartenance d'un médium à Bob n'apprendrait rien de la marque qu'il contient. Sans cette connaissance, il est plus difficile de mener des attaques comme celle décrite dans [KVH00] où Charlie estime la marque présente dans un médium avant de la recopier dans un autre.

Aucun programme publiquement disponible ne repose sur ce type de protocoles. Néanmoins, des études ont déjà été menées dans cette direction. Nous commençons donc par les présenter. Ensuite, nous proposons deux schémas théoriques dans la mesure où, comme nous le verrons par la suite, la robustesse de tels solutions semble difficile à obtenir.

5.4.2.1 Solutions existantes

S. Craver ([Cra00]) propose différentes solutions pour utiliser un protocole à divulgation nulle de connaissance dans le cadre du tatouage d'images. H. Kinoshita et T. Kobayashi présentent également une solution dans [KK00] où ils prennent en compte une composante oubliée par Craver : la complexité du problème sur lequel repose le protocole *zero-knowledge* .

Néanmoins, les solutions proposées par S. Craver dans [Cra00] ne sont pas à divulgation nulle de connaissance. Il a publié un article ([CK01]) dans lequel il montre que des problèmes sont présents dans toutes ses propositions, à l'exception de celle que nous détaillons ci-après. Nous montrons toutefois que ce protocole n'est pas non plus *zero-knowledge* .

Protocole fondé sur le schéma de Pitas proposé par S. Craver

Le schéma introduit par Pitas dans [Pit96] propose de modifier une propriété statistique liant deux ensembles de pixels générés aléatoirement. La version la plus classique modifie la différence moyenne entre ces ensembles. Normalement, celle-ci est nulle. Toutefois, en ajoutant une valeur k à chaque pixel d'un des ensembles, et en retranchant k à ceux de l'autre, cette différence vaut alors $2\mathsf{k}$. Le tatouage en lui-même, noté d, est une image binaire qui révèle la topologie des ensembles.

La marque d est assimilable à la matrice d'un graphe. Pour rendre ce protocole *zero-knowledge*, S. Craver suggère d'appliquer le tatouage tel que décrit précédemment puis de révéler publiquement une permutation $\sigma(\mathsf{d})$ de la marque. Ensuite, tous les graphes isomorphes[10] à la marque sont considérés comme preuve de l'identité de Bob. Le médium tatoué $\tilde{\mathsf{m}}$ et la marque permutée $\sigma(\mathsf{d})$ sont rendus public. La vérification s'effectue alors comme dans le protocole cryptographique équivalent à l'aide d'un défi répété plusieurs fois. Le tableau 5.1 décrit ce protocole.

Génération du stégo-médium

1. Bob tatoue le médium m avec une marque d et obtient $\tilde{\mathsf{m}}$ (toutes les marques valides sont les permutations de d)
2. Bob choisit une permutation σ qu'il conserve secrète
3. Bob révèle $\tilde{\mathsf{m}}$ et $\sigma(\mathsf{d})$

Détection de la marque

1. Alice récupère la paire $(\tilde{\mathsf{m}}, \sigma(\mathsf{d}))$ et demande à Bob de lui prouver que $\tilde{\mathsf{m}}$ contient bien d
2. Bob génère aléatoirement une permutation π, calcule $\tilde{\mathsf{m}}_\pi = \pi(\tilde{\mathsf{m}})$ qu'il transmet à Alice
3. Alice demande à Bob de lui révéler :
   - soit π, auquel cas Alice vérifie que $\tilde{\mathsf{m}}_\pi = \pi(\tilde{\mathsf{m}})$
   - soit $\pi(\mathsf{d})$ et Alice contrôle que $\tilde{\mathsf{m}}_\pi$ contient bien $\pi(\mathsf{d})$
4. Bob doit enfin prouver à Alice que la marque contenue dans $\tilde{\mathsf{m}}_\pi$ est valide, *i.e.*, qu'elle résulte bien d'une permutation de $\sigma(\mathsf{d})$: il dévoile l'isomorphisme ρ, tel que $\rho(\sigma(\mathsf{d})) = \pi(\mathsf{d})$

TABLEAU 5.1 : Schéma de Pitas en version *zero-knowledge*

[10] Deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$, où V_i et E_i sont respectivement l'ensemble des sommets et des arêtes, sont isomorphes ssi il existe une bijection f telle que $\forall \{u, v\} \in E_1, \{f(u), f(v)\} \in E_2$.

En fait, outre les problèmes relevés dans l'article (robustesse et attaque des anniversaires), **nous avons remarqué que ce schéma n'est pas à divulgation nulle de connaissance**. Dans la première phase du défi, lorsqu'Alice demande l'isomorphisme π, elle est alors en mesure de reconstruire d. À la fin du protocole, Bob doit prouver que la marque contenue dans $\tilde{\mathsf{m}}_\pi$ est valide en montrant qu'elle est égale à $\sigma(\mathsf{d})$, à une permutation près. Pour y parvenir, il donne $\pi(\mathsf{d})$ à Alice. Elle dispose alors de $\pi(\mathsf{d})$ et de π : il lui suffit d'inverser l'isomorphisme π pour récupérer la marque initiale $\mathsf{d} = \pi^{-1}(\pi(\mathsf{d}))$. Ainsi, Alice peut utiliser cette marque soit pour se faire passer pour Bob (impersonnification), soit pour la reproduire dans d'autres média (attaque par recopie).

5.4.2.2 Isomorphisme de graphes

Comme le montre le premier protocole présenté par S. Craver, il est nécessaire de lier entre eux un des graphes et l'image à tatouer. Le schéma proposé ici reprend celui général présenté dans le tableau B.2 page 199. Bob est le propriétaire d'un médium m. Il dépose auprès d'un tiers de confiance un isomorphisme de graphes σ associé à ce médium. Il dévoile publiquement deux graphes $\mathsf{G_0}$ et $\mathsf{G_1} = \sigma(\mathsf{G_0})$.

Ultérieurement, quand Alice souhaite vérifier que Bob est bien le propriétaire légitime de m, elle lui lance un défi. Bob choisit alors aléatoirement une autre permutation π et transmet à Alice le graphe $\mathsf{H} = \pi(\mathsf{G_0})$. Le protocole suit son cours tel que décrit précédemment. Ici, les graphes $\mathsf{G_0}$ et $\mathsf{G_1}$ sont publics, tout comme le stégo-médium. Toutefois, l'attaquant ne sait pas *a priori* quel graphe est inséré dans le stégo-médium.

D'un point de vue tatouage, cette méthode repose principalement sur les graphes $\mathsf{G_0}$ et $\mathsf{G_1}$. Nous proposons différentes solutions pour construire ces graphes dans le cas d'une image, mais chacune comporte des lacunes :

- utilisation des points d'intérêts :
  lorsque l'œil regarde une image, il remarque naturellement les différentes régions qui la composent (textures, bords, etc.), principalement parce que le cerveau interprète l'image en même temps que l'oeil la perçoit. Les détecteurs de points d'intérêt cherchent à automatiser cette démarche. Il en existe de nombreux dont la présentation dépasse le cadre de cette étude (voir [HS88, Nob88] ou des ouvrages plus généraux de traitement d'images [Pra78]).
  Le graphe $\mathsf{G_0}$ est construit à partir des ces points qui servent de sommet. Les arêtes sont données par la méthode de tatouage elle-même, et donc paramétrées par une clé secrète.
- utilisation de l'image elle-même en guise de graphe :
  un graphe est défini par une matrice de transitions. Or, une image est une matrice[11]. On choisit aléatoirement un sous-ensemble de lignes et de

[11] C'est direct pour les images en niveaux de gris, à une transformation près pour les images couleurs.

colonnes pour déterminer la matrice du graphe G_0.

- utilisation d'une autre représentation de l'image :
  plutôt que d'extraire un graphe directement de l'image, on manipule une autre représentation (DCT, fréquentielle, par ondelettes, par ses valeurs propres ou singulières). Le graphe G_0 est construit en sélectionnant certains des points dans ce nouvel espace, comme précédemment avec les pixels de l'image.

La sécurité des protocoles à divulgation nulle de connaissance repose des *problèmes difficiles*[12] Néanmoins, l'ordre de grandeur du nombre de points requis pour se protéger d'une recherche exhaustive est relativement important, ce qui n'est pas toujours compatible avec le médium considéré. De plus, il faut que le graphe soit robuste par rapport à l'image, c'est-à-dire que le vérifieur doit pouvoir le récupérer à l'identique pour appliquer la phase de vérification *zero-knowledge*. En effet, la moindre modification du graphe conduira le protocle à échouer. Ainsi, outre ces problèmes de robustesse, il en existe également deux autres liés à la sécurité du système.

Tout d'abord, une personne malveillante s'approprie facilement le médium $\tilde{\mathsf{m}}$ tatoué par Bob. Supposons que Charlie se procure le médium tatoué $\tilde{\mathsf{m}}$ et les graphes G_0 et G_1. Sachant que G_0 est lié à l'image, il construit une autre permutation σ_{C} telle que $\mathsf{G}_1^{\mathsf{C}} = \sigma_{\mathsf{C}}(\mathsf{G}_0)$. S'il rend public le triplet $(\tilde{\mathsf{m}}, \mathsf{G}_0, \mathsf{G}_1^{\mathsf{C}})$, il est en mesure de revendiquer le médium exactement comme son propriétaire initial. On considère dans ce cas que le protocole a échoué. Cette attaque ressemble à celle décrite par S. Craver dans [CMYY98], à la différence que Charlie est incapable de produire le médium initial, au contraire de Bob. Cette solution ne dévoile certes aucune information sur la marque, mais elle pousse Bob à révéler le médium original, ce qui n'est guère mieux.

Cette même raison justifie l'étape supplémentaire prévue par S. Craver dans le protocole présenté dans le tableau 5.1 page 146. Cependant, comme nous l'avons montré, cette étape supplémentaire a un prix : le protocole n'est plus *zero-knowledge*. Le problème vient ici de la divulgation du graphe G_0 contenu dans l'image. Toutefois, si celui-ci n'est pas public, le protocole ne fonctionne pas car le vérifieur ne peut contrôler l'exactitude de la réponse du prouveur qu'une fois sur deux.

L'autre faiblesse réside dans la combinatoire. Si on considère les points d'intérêt, leur nombre est assez peu élevé dans une image. Un attaquant patient, mais néanmoins volontaire, peut tenter une attaque exhaustive pour découvrir la permutation σ liant G_1 à G_0. Ainsi, les auteurs de [KK00] fournissent un ordre de grandeur d'un graphe afin qu'il soit fiable : ils estiment qu'il doit contenir plus de 1000 sommets afin qu'une attaque exhaustive soit irréalisable en pratique.

Une telle approche semble donc compromise par les multiples défauts que révèle cette analyse. Il nous semble alors nécessaire d'envisager une autre solution.

[12] En simplifiant, il s'agit de problèmes dont la complexité est exponentielle.

5.4.2.3 Graphe contenant un cycle Hamiltonien

Le schéma que nous proposons maintenant reprend celui présenté dans le tableau B.3 page 200. Cette fois, le graphe G dissimulé dans l'image contient un circuit Hamiltonien, conservé secret. Une fois tatouée, l'image et le graphe sont rendus publics. Le protocole de vérification s'effectue alors normalement.

Cette application du protocole est potentiellement sensible aux mêmes problèmes que la précédente, à une exception près. Tout d'abord, le graphe a besoin d'être de grande taille pour éviter une recherche exhaustive. Ensuite, il doit être inséré de manière robuste dans l'image.

Cependant, la différence principale entre ce protocole et le précédent est que celui-ci ne nécessite de rendre public qu'un seul graphe (G, qui contient le circuit Hamiltonien). Ce changement suffit à rendre l'impersonnification décrite auparavant impossible, sous la condition que le graphe ne contienne qu'un unique chemin Hamiltonien.

S'il en recèle plusieurs, une attaque par impersonnification est alors envisageable. L'attaquant peut utiliser cet autre circuit pour se faire passer pour le propriétaire légal dans les multiples défis du protocole. Toutefois, la complexité de cette attaque est identique à celle qui conduit l'attaquant à retrouver le circuit Hamiltonien initialement introduit par l'ayant-droit : ces deux situations nécessitent de découvrir un parcours Hamiltonien dans un graphe. Or, cette tâche est plus facile lorsque plusieurs chemins sont présents dans le graphe.

5.4.2.4 Enjeux des protocoles *zero-knowledge* pour le tatouage et le fingerprinting

Aussi bien en tatouage qu'en fingerprinting, les données insérées dans le médium poursuivent l'objectif d'identifier une entité de manière unique, cette entité pouvant être aussi bien une personne qu'une transaction.

Pour déjouer un des ces schémas, il n'est nullement besoin de retirer les données ajoutées : les modifier suffisamment ou les remplacer pour tromper le détecteur constitue déjà un échec de celui-ci (voir les différents niveaux d'attaques en 5.5.4 page 154). Ainsi, ces protocoles permettraient de vérifier la validité d'une marque ou d'une empreinte sans révéler la moindre information sur celle-ci. La plupart des fonctions de détection actuelles donnent de l'information sur les données car elles vérifient une hypothèse publiquement connue mais simplement paramétrée par une clé. On ajoute ainsi une contrainte supplémentaire par rapport à une approche asymétrique classique.

Pour être réellement efficace, une solution *zero-knowledge* pour la dissimulation d'information suppose que les données sont insérées de manière robuste dans le médium : si elles sont modifiables, un protocole à divulgation nulle de connaissance n'apporte rien. Ce constat nous laisse penser que plutôt que les données, le stégo-médium lui-même devrait intervenir dans le processus, et non pas servir simplement de support à des données.

Actuellement, la sécurité d'un schéma de tatouage repose essentiellement sur sa robustesse. Toutefois, dans la pratique, l'attaque menée dans [BS01] contre

les tatouages proposés par le SDMI repose sur une estimation de la marque ainsi que les réponses successives fournies par un oracle chargé de contrôler la présence d'une marque dans un médium. La réussite de cette attaque montre donc l'intérêt de disposer d'un schéma donnant un minimum d'information à un attaquant.

En conclusion, l'identification est une question centrale en dissimulation d'information, et surtout dans la perspective du tatouage et du fingerprinting. Pour cela, l'adaptation des protocoles à divulgation nulle de connaissance semble être une solution confortable, mais l'intégration de ces protocoles dans les schémas de tatouage pose deux problèmes. D'une part, la taille des objets manipulés dans ces protocoles est relativement élevée, ce qui n'est pas toujours compatible avec un médium. D'autre part, l'attaque par recopie ou [BS01] évoqué ci-dessus montrent que des connaissances sur la marque ne sont pas toujours requises, des hypothèses sur sa structure suffisant en pratique.

5.5 Signature numérique

Les objectifs des schémas de signature numérique (cf. B.5 page 200) semblent bien correspondre à ceux qui apparaissent en dissimulation d'information, en particulier dans le cas du tatouage et du fingerprinting. Toutefois, il existe de nombreuses différences.

5.5.1 Types de schéma

Il existe deux types de signature numérique : avec annexe ou avec recouvrement (cf. B.5 page 200). Du fait de la proximité relative des schémas de signature et de dissimulation, la question se pose de savoir si les solutions du premier sont applicables au second.

Une signature avec annexe ajoute des données (la signature) au message, mais celle-ci peut voyager indépendamment du message. En général, elle est simplement ajoutée à la fin. À l'autre extrémité du canal, une personne récupère le message et sa signature pour en vérifier la validité. Cette séparation va à l'encontre des contraintes imposées en dissimulation d'information où le médium sert à transporter les données.

Au contraire, les signatures par recouvrement offrent une perspective intéressante. On peut en effet voir le stégo-médium comme un recouvrant des données qu'il contient. Par exemple, en stéganographie, où le stégo-médium ne sert que de vecteur de transport, nous pouvons construire un schéma théorique où le message et la clé servent à construire un stégo-médium unique. À l'autre extrémité du canal, lors de la réception, la clé et le stégo-médium permettent de revenir au message ainsi dissimulé.

5.5.2 Comparaison des schémas de signature et de dissimulation

Les schémas de signature numérique aussi bien que ceux de dissimulation d'information nécessitent l'utilisation de deux fonctions distinctes. Des points communs entre ces fonctions apparaissent assez naturellement et sont présentés dans le tableau 5.2.

Signature numérique	**Dissimulation d'information**
message : m	médium : m
haché/redondant : $\tilde{m}$	données : d
fonction de signature	fonction d'insertion
fonction de vérification	fonction de détection ou d'extraction

TABLEAU 5.2 : Analogies entre signature et dissimulation d'information

On définit un schéma de dissimulation d'information général très semblable aux schémas de signature présentés (voir tableaux B.5 et B.6).

Génération du stégo-médium

1. Bob génère ses données d
2. Bob insère ses données dans le médium $\mathsf{e}_{\mathsf{k_B}}(\mathsf{m}, \mathsf{d}) = \tilde{\mathsf{m}}_{\mathsf{d}}$
3. Bob rend public son stégo-médium $\tilde{\mathsf{m}}_{\mathsf{d}}$

TABLEAU 5.3 : Schéma général de dissimulation

Présenter une application identique pour la détection n'est pas aussi simple. Les objectifs des schémas de dissimulation étant très variés, la fonction de détection fournit des résultats adaptés :

- une valeur binaire parmi $\{\mathsf{vrai}, \mathsf{faux}\}$: de plus en plus rares, ces schémas reposent souvent sur un seuil tel que présenté ci-après ;
- un indice de confiance sur la présence des données d dans le stégo-médium $\tilde{\mathsf{m}}_{\mathsf{d}}$:
  1. extraction des données d' de $\tilde{\mathsf{m}}_{\mathsf{d}}$
  2. corrélation entre d et d'

  Tout comme la fonction de vérification, la fonction de détection utilise ici le médium à tester pour extraire des données et les comparer à celles attendues.
  En toute rigueur, de nombreux schémas de tatouage n'extraient pas explicitement la marque et se contentent de vérifier, à partir de la marque supposée d, une hypothèse sur sa présence dans le médium ;

– les données d elles-mêmes : les données sont extraites tout comme précédemment, mais elles sont utilisées sans autre forme de contrôle, par exemple lorsque les données initiales ne sont pas disponibles (comme en stéganographie).

Cependant, la nature même de la fonction de vérification change entre ces deux disciplines. En cryptographie, une méthode n'est acceptée en tant que schéma de signature qu'une fois qu'il est démontré que la fonction de vérification remplit bien son office. Ainsi, dès les prémisses, ces schémas réfutent l'idée de faux-positifs ou faux-négatifs : une signature est valide ou non, mais il n'y a pas d'intermédiaire.

Toutefois, la similitude de ces schémas n'empêche pas de profondes différences de subsister. Nous les détaillons dans la partie suivante, en étudiant les propriétés requises pour la signature numérique.

5.5.3 Analyse des propriétés des schémas de signature appliquées à la dissimulation d'information

Pour définir les schémas de signature, les cryptographes ont commencé par poser les propriétés nécessaires afin de parvenir à des solutions sûres. Une signature numérique se doit donc d'être :

1. *non reproductible* : personne d'autre que le signataire ne peut copier la signature associée à un document et une personne ;
2. *publique* : n'importe qui doit pouvoir vérifier la validité d'une signature[13] ;
3. *non réutilisable* : l'utilisation d'une signature avec tout autre message que celui pour lequel elle a été générée doit être impossible ;
4. *conforme au principe d'avalanche* : si le message est modifié, la signature ne doit plus être valide ;
5. *non répudiable* : le signataire ne peut nier, *a posteriori*, avoir signé.

Lorsque ces propriétés sont vérifiées, le schéma de signature est considéré comme sécurisé.

Nous ne reviendrons pas ici sur l'aspect asymétrique des schémas de dissimulation d'information puisque nous l'avons déjà évoqué en 5.2 page 131 et qu'il ne s'agit pas d'un impératif pour la sécurité.

La différence majeure entre la stéganographie d'une part, le tatouage et le fingerprinting d'autre part réside dans la relation entre le médium et les données. Dans ces deux dernières disciplines, tout comme dans les schémas de signature numérique, une entité cherche à créer un lien entre des données (le haché ou le redondant pour les signatures) et le médium (respectivement le message).

En stéganographie, soit le message est complètement indépendant du médium dans lequel il est dissimulé, soit le médium est construit spécialement pour dissimuler le message. La comparaison avec la signature numérique n'est donc

[13] Ceci n'est pas nécessaire du point de vue de la sécurité, il s'agit juste d'une fonctionnalité supplémentaire.

pas très pertinente. Comme nous l'avons évoqué dans la partie sur l'intégrité des données (cf. B.3 page 193), les seules garanties nécessaires en stéganographie concernent le message, et non le stégo-médium qui le transporte. Elles incluent son intégrité et l'identification de sa provenance.

Pour le tatouage et le fingerprinting, l'attaque par recopie que nous avons déjà évoquée précédemment montre précisément pourquoi les schémas proposés se doivent d'être non reproductibles. Nous avons également déjà présenté les problèmes liés à la réutilisation d'une marque dans la partie 5.1 page 130 sur la gestion des clés. Le principe d'avalanche va à l'encontre même des attentes de ces domaines, au contraire de la notion de «hachage mou». Enfin, la non répudiation pose un autre problème.

En tatouage, que faire si un propriétaire qui, après avoir bénéficié pendant quelques temps de ses média, décide de les vendre ? À ce moment, il existe déjà une multitude de copies des média qui contiennent la marque du propriétaire : il est pratiquement impossible d'échanger chaque exemplaire afin de remplacer l'ancienne marque par un tatouage propre au nouvel ayant-droit. L'identité du propriétaire est attachée à un médium à l'aide de sa clé secrète k et d'une marque d insérée dans le médium. Supposons qu'Alice détienne des média que Bob souhaite acheter :

- si une même clé k et une même marque d sont toujours utilisées par Alice, elle ne peut alors céder ses droits que sur l'intégralité de ses œuvres, puisque pour protéger les média qu'il achète, Bob a besoin de la paire (k, d) : il contrôle alors en même temps tous les autres média d'Alice ;
- si Alice emploie une clé distincte dans chaque médium qu'elle protège, mais utilise une marque identique[14], Bob n'a besoin que des clés correspondant aux média qu'il achète et de la marque d'Alice. Cependant, celle-ci doit gérer un énorme trousseau de clés. Par ailleurs, une même marque permet alors d'identifier des personnes différentes ;
- si Alice dissimule dans ses média une marque différente, toujours à l'aide de la même clé, elle ne peut alors se contenter de céder à Bob les marques utilisées dans les média de la transaction car sans la bonne clé, Bob est incapable d'agir sur ses nouvelles acquisitions, mais possédant la clé d'Alice, il est alors aussi capable d'agir sur les média d'Alice qu'il ne possède pas ;
- si la protection résulte d'une clé et d'une marque différentes à chaque fois, la cession de ses droits est réalisée en changeant simplement l'identité attachée à la paire (k, d) mais au prix de la gestion fastidieuse d'un très grand nombre de ces paires.

La non-répudiation semble donc problématique en tatouage.

En fingerprinting, cette propriété est absolument indispensable. En effet, si un schéma devait être répudiable, alors un utilisateur pourrait changer l'empreinte contenue dans sa copie par celle de n'importe qui d'autre, ce qui est exactement le contraire du but recherché.

[14] Ceci suppose que les clés ne servent pas à la génération de la marque.

5.5.4 Portée des attaques

Les niveaux d'attaque contre les schémas de signature numérique sont assez étendus. L'attaquant peut simplement parvenir à produire une signature valide appartenant à sa cible pour un message qu'il ne contrôle pas du tout, ou encore à récupérer sa clé (voir B.5.3 page 203). En dissimulation d'information, la granularité des objectifs est similaire :

- **Stéganographie :**
  - Détection de la présence d'un message.
  - Lecture du message dissimulé.
  - Modification/suppression du message.
- **Tatouage :**
  - Modification du stégo-médium ou de la marque rendant celle-ci indétectable.
  - Insertion d'une autre marque rendant les deux marques détectables (celle initiale et celle de l'attaquant).
  - Retrait complet de la marque du stégo-médium ou remplacement par une autre marque.
- **Fingerprinting :**
  - Modification du stégo-médium ou de l'empreinte rendant celle-ci indétectable.
  - Insertion d'une autre empreinte rendant les deux empreintes détectables (celle initiale et celle de l'attaquant).
  - Retrait complet de l'empreinte du stégo-médium ou remplacement par une autre empreinte.

5.5.5 Exemples : l'attaque d'IBM et l'attaque par recopie

Le haché (ou le redondant) d'un côté et les données de l'autre ne sont pas de même nature. Néanmoins, dans certains cas, ils se rejoignent. S. Craver, dans [CMYY98], a montré que les schémas de tatouage inversibles étaient vulnérables à une attaque protocolaire lorsque le médium initial était requis pour la détection.

Soient m un médium et d une marque. L'insertion de la marque est réalisée selon une opération facilement inversible et linéaire notée e. Le médium marqué est donné par $\tilde{\mathsf{m}}_{\mathsf{d}} = \mathsf{e}(\mathsf{m}, \mathsf{d})$. La fonction e étant connue et inversible, un attaquant peut alors prétendre détenir un original m' et une marque d' tels que $\mathsf{d}(\tilde{\mathsf{m}}_{\mathsf{d}}, \mathsf{m}', \mathsf{d}') = \mathsf{vrai}$. En effet, il lui suffit de prendre sa propre marque d' et d'inverser la fonction e : $\mathsf{m}' = \mathsf{e}^{-1}(\tilde{\mathsf{m}}_{\mathsf{d}}, -\mathsf{d}')$.

On retrouve ici la propriété nécessaire énoncée par Craver dans [CMYY98] pour les méthodes inversibles. En effet, à l'aide de la fonction de hachage ou de redondance, le message et la signature sont indissociablement liés. La solution proposée par Craver est justement de rendre les données insérées dans le médium dépendantes de celui-ci, afin d'obtenir un lien aussi fort, en les calculant à partir d'un haché de l'image.

L'attaque par recopie (voir 5.3.1 page 139) conduit aux mêmes conséquences par rapport aux signatures, même si l'attaque d'IBM ajoute une nouvelle marque alors que celle-ci en duplique une existante. Cette attaque est possible car la marque est reproductible et réutilisable, ce qui est à l'encontre des propriétés requises pour les signatures numériques. Un schéma sensible à cette attaque est donc répudiable car le propriétaire n'est alors plus le seul à pouvoir tatouer un médium.

Ces attaques illustrent un cassage complet (cf. B.5) par lequel Charlie parvient à mettre en place un algorithme équivalent qui produit des signatures valides. De telles attaques sont impossibles avec des signatures numériques car les propriétés nécessaires à leur sécurité sont bien connues, et intégrées directement dans le processus.

En conclusion, la démarche adoptée pour construire un schéma de signature – *i.e.* définir les propriétés recherchées pour ensuite créer les schémas – permet à la cryptographie de fournir des solutions efficaces et adaptées aux besoins. En dissimulation d'information, la question des propriétés requises pour le protocole qui utilise le schéma n'apparaît jamais. La difficulté à définir ces attentes vient sans doute de la multiplicité des objectifs poursuivis. Le cas de la non-répudiation est particulièrement explicite. Par rapport au tatouage, elle pose le problème de la gestion du lien entre la clé et les données. En revanche, pour le fingerprinting, elle est absolument nécessaire de la même manière que pour une signature numérique.

5.6 Partage de secret

En pratique, le partage de secret sert à protéger des serveurs dans le cas où l'un d'eux devient inaccessible. Ainsi, lorsque l'information est propagée selon un schéma de partage de secret à seuil, la défection d'un serveur ne condamne pas l'ensemble et le système continue à fonctionner normalement.

Par rapport à leur organisation, les méthodes de dissimulation d'information nécessitent rarement que plusieurs personnes disposent du secret :

- en stéganographie, la possibilité d'extraire un message d'un stégo-médium en respectant une structure d'accès correspond exactement à la situation en cryptographie ;
- en tatouage, seules deux intervenants sont concernés : l'ayant-droit et le vérifieur, ce qui ne nécessite pas de contrôle particulier, d'autant plus que le propriétaire ne souhaite pas partager le secret avec le vérifieur ;
- en fingerprinting, nous avons vu un schéma reposant sur l'utilisation parallèle de plusieurs clés privées. Or, le partage de secret s'appuie plutôt sur une utilisation conjointe du secret.

Pour ces deux derniers domaines, le secret peut être partagé entre tous les intervenants qui détiennent le médium. Par exemple, pour une chanson, le compositeur, l'interprète et le distributeur détiennent chacun une part du secret.

Une autre utilisation du partage de secret semble plus appropriée lorsqu'elle se produit dans le schéma de tatouage lui-même afin de contrer efficacement toutes les attaques fondées sur l'extraction de «morceaux» de médium.

Si chaque extrait du stégo-médium contient une part du secret, il faut alors rassembler un nombre minimal (*i.e.* le seuil) de parts afin de retrouver le tatouage ou l'empreinte. Ainsi, les attaques comme le *cropping* ou la mosaïque seraient probablement moins efficaces, selon que les parts seraient réparties dans n'importe quels «morceaux» ou des blocs qualibrés.

5.7 Conclusion

Dans ce chapitre, nous avons présenté différentes utilisations pour bénéficier des propriétés de la cryptographie en dissimulation d'information. Deux approches sont possibles pour jouir de cet apport :

- intégration dans le processus de dissimulation d'information : nous avons montré de telles possibilités avec les protocoles à divulgation nulle de connaissance ou encore le partage de secret ;
- intégration dans le protocole de dissimulation d'information : nous avons vu l'intérêt de rendre les schémas de dissimulation d'information asymétriques, ou encore comment l'adaptation des protocoles de signatures pour obtenir des propriétés similaires à celles garantissant la sécurité du système.

Comme nous l'avons montré, les solutions existant en cryptographie sont rarement utilisables directement en dissimulation d'information.

Les perspectives les plus prometteuses proviennent d'une part de l'association avec des protocoles à divulgation nulle de connaissance et d'autre part du hachage mou. L'intérêt du premier est d'offrir un niveau de protection supplémentaire dans la vérification de la marque. Sans la connaissance de celle-ci, nous avons vu qu'il était néanmoins possible de réaliser une attaque par impersonnification (cf. section 5.4.2.2) si le protocole est mal construit. Au contraire, comme nous l'avons illustré, l'utilisation d'un protocole *zero-knowledge* fondé sur l'emploi d'un circuit Hamiltonien permet de combler cette faille.

Par ailleurs, la notion de hachage mou apparaît comme centrale car de telles fonctions permettraient de conserver l'information significative d'un médium, et donc de parvenir à une meilleure définition de celui-ci, indépendante de ses représentations (formats, modifications plus ou moins importantes, etc). La problématique de ces fonctions n'est pas propre à la dissimulation d'information. Elles peuvent également servir en indexation de documents : le haché obtenu sert d'index et la recherche s'effectue alors sur ceux-ci à l'aide d'une métrique appropriée pour quantifier les similarités.

Il nous semble difficile de vouloir mettre en place des protocoles pour la dissimulation d'information tant que les attentes précises des primitives élémentaires,*i.e.* les fonctions d'insertion et de détection, ne sont pas précisément définies. Si on compare, par exemple, à ce qui se fait avec les fonctions de ha-

chage, elles sont construites de manière à vérifier certaines propriétés. Ensuite, les protocoles qui les emploient sont sécurisés car ils reposent, entre autre sur ces propriétés. Une démarche similaire nous semble indispensable pour parvenir à un niveau raisonnable de sécurité.

Gestion des clés	
Constat :	Nécessité d'une autorité de gestion des clés (et marques/empreintes)
Objectif :	Évaluation de la complexité des attaques visant à retrouver la clé (notamment la recherche exhaustive)
Chiffrement	
Constat :	Autorisation d'accéder aux données *via* un secret
Objectif :	Schémas asymétriques
Hachage et intégrité des données	
Constat :	Élargissement de la notion de collision par rapport aux fonctions de hachage cryptographique
Objectif :	Définition de fonctions de hachage mou
Identification	
Constat :	Attaques par impersonnification
Objectif :	Contrôle des informations par un protocole *zero-knowledge*
Signatures numériques	
Constat :	Analogies avec la dissimulation d'information, mais aussi de profondes différences
Objectif :	Définition de propriétés préalablement requises pour construire des schémas
Partage de secret	
Constat :	Inapproprié tant que seules deux entités interviennent
Objectif :	Protection contre les attaques type mosaïque

TABLEAU 5.4: Constats et perspectives sur liens entre certains domaines issus de la cryptographie et la dissimulation d'information

Chapitre 6

Application : stéganographie *via* le protocole SSH

La plupart des études sur la stéganographie portent sur l'emploi de textes ou d'images en guise de vecteur de transport. Nous nous intéressons ici à un autre support : les protocoles réseau.

Un *canal stéganographique*[1] sur un réseau a pour but :

1. de dissimuler la communication ;
2. d'être fiable (*i.e.* sans erreur) :
3. de permettre la transmission d'autant de données que souhaitées.

La mise en œuvre d'un tel canal par l'intermédiaire du protocole TCP/IP est assez délicate. En effet, toute génération de trafic superflu est (potentiellement) immédiatement détectée par les outils de supervision réseau[2].

Nous partons du postulat que tout trafic anormal sur le réseau est soit écouté, soit détecté et qu'il ne peut donc servir comme canal stéganographique.

Dans ce chapitre, commençons par présenter un bref état de l'art ([Row96, HI96]) proposant d'utiliser les protocoles réseau du modèle OSI. Nous analysons ces solutions puis montrons en quoi elles ne sont pas acceptables. Nous étudierons également les protocoles ICMP et UDP respectivement situés aux mêmes niveaux que IP et TCP dans le modèle OSI. La deuxième partie est exclusivement réservée à la présentation du protocole SSH. Même s'il est de plus en plus répandu, nous donnons ici quelques indications sur son fonctionnement. Enfin, la dernière partie présente l'utilisation d'un canal stéganographique au travers de SSH, par deux méthodes distinctes : le message `SSH_MSG_IGNORE` et les octets de bourrage.

[1] On parle aussi de *canal caché*.

[2] Les *Intrusion Detection Systems* (IDS) analysent les paquets soit selon un ensemble de règles, soit selon l'adéquation à un modèle statistique afin de déterminer ceux qui interviennent dans une attaque.

6.1 Introduction : problématique d'un canal caché sur un réseau

Après avoir analysé l'unique article traitant de la question et proposant d'utiliser les protocoles IP et TCP, nous montrons que d'autres protocoles situés aux mêmes niveaux ne sont pas non plus adaptés.

6.1.1 Rappels de réseau

Afin de faciliter la compatibilité des matériels, le *modèle OSI* propose une abstraction en couches des différents services nécessaires au bon fonctionnement d'un réseau. Il en découle que les données sont *encapsulées* pour passer d'un niveau à l'autre lors de leur émission sur le réseau pour être ensuite désencapsulées à la réception. Nous rappelons dans cette partie ces deux concepts.

Signalons toutefois qu'il existe d'autres modèles comme le DoD (du nom du *Department of Defense* américain) qui ne contient que 4 couches

6.1.1.1 Le modèle OSI

Le modèle OSI (*Open Systems Interconnection*) a été défini par l'*Internatinal Standardization Organisation* (ISO). Afin de mettre en place un standard de communication entre les ordinateurs d'un réseau, c'est-à-dire les règles qui gèrent les communications entre des ordinateurs. En effet, aux origines des réseaux chaque constructeur avait un système propre (on parle de système propriétaire). Ainsi de nombreux réseaux incompatibles coexistaient. C'est la raison pour laquelle l'établissement d'une norme a été nécessaire.

Il s'agit d'un modèle en *couches*[3] dont l'intérêt est de séparer le problème en différentes parties (les couches) selon leur niveau d'abstraction. Chaque couche du modèle communique avec une couche adjacente : elle utilise les services de la couche inférieure et en fournit à la couche supérieure. Enfin, un *protocole* (ou une *relation protocolaire*) est un lien entre deux couches distantes de même niveau.

Le modèle OSI comporte sept couches :

[3] On parle aussi de *niveaux*.

Niveau	Nom	Fonction
7	Couche Application	Assurer l'interface avec les applications
6	Couche Présentation	Formater des données (leur représentation, éventuellement leur compression)
5	Couche Session	Fournir les moyens d'organiser et synchroniser les dialogues et les échanges de données
4	Couche Transport	Transporter les données et, selon le protocole, gérer les erreurs
3	Couche Réseau	Gérer l'adressage et le routage
2	Couche Liaisons	Définir l'interface avec la carte réseau et méthode d'accès
1	Couche Physique	Convertir des données en signaux numériques

Ce modèle est une référence mais aucune implémentation ne la respecte scrupuleusement (mis à part les protocoles OSI comme TP4 ISO - équivalent à TCP). Par exemple, les Unix et les OS de Microsoft permettent de passer directement de la couche Application à la couche Transport à l'aide des *sockets*.

6.1.1.2 Encapsulation des données

Lors d'une transmission, les données traversent chacune des couches dans le sens descendant pour le client (*i.e.* du niveau 7 au niveau 1), puis dans l'autre sur le serveur.

À chaque niveau, le paquet de données change donc d'aspect, car on lui ajoute un en-tête, ainsi les appellations changent suivant les couches :

- le paquet de données est appelé *message* au niveau de la couche application ;
- le message est ensuite encapsulé sous forme de *segment* dans la couche transport ;
- le segment, une fois encapsulé dans la couche réseau, prend le nom de *datagramme* (ou *paquet* par abus de langage) ;
- enfin, on parle de *trame* au niveau de la couche liaison.

Les protocoles IP et ICMP[4] se situent au niveau 3. La couche 4 comprend TCP et UDP. Enfin, les applications comme telnet, ftp ou ssh sont placés au niveau 7.

Le protocole TCP fonctionne en *mode connecté* : chaque paquet émis est acquitté par la destination. Pour IP et UDP, ce n'est pas la cas : on dit qu'il s'agit d'un protocole *non connecté* car l'émetteur n'a aucun moyen de savoir si le paquet UDP est bien arrivé.

[4] *Internet Control Message Protocol* (ICMP - RFC 792) est un protocole de gestion et de contrôle d'erreurs d'IP.

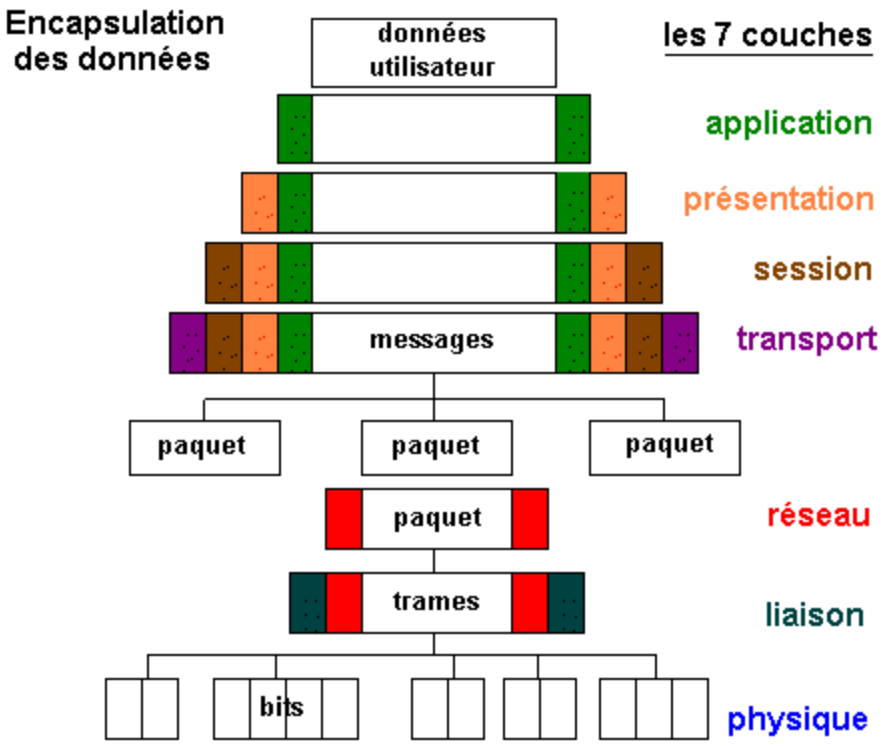


FIGURE 6.1 : encapsulation des données

6.1.2 Utilisation des en-têtes TCP/IP

Une étude réalisée en 1996 [Row96] propose trois approches pour dissimuler une communication dans les couches réseau TCP/IP. Le principe général est toujours le même : construire de faux paquets en utilisant certains champs des en-têtes IP et TCP pour dissimuler de l'information :

1. le champ `identification` (ou `id`) : au niveau IP, il permet de reconstruire les paquets fragmentés lors du passage de la couche 4 à la couche 3 (voir fig. 6.1). L'auteur propose de lui substituer le code ASCII du caractère à transmettre, à une transformation près ;
2. le champ `ISN` (*Initial Sequence Number*) : lors de l'initialisation d'une communication au niveau TCP (*i.e.* avec le bit `SYN` - *synchronize*), cette valeur est la première de la suite des numéros de séquence (voir fig. 6.2) ;
3. le champ `acquittement` par *rebond* : lorsqu'un serveur reçoit un paquet TCP avec le bit `SYN` pour initialiser une connexion, il répond à l'émetteur en inscrivant la valeur $ISN + 1$ dans le champ `acquittement` et en positionnant le bit `ACK` (*acknowlege*) du paquet[5].

[5]le rebond est utilisé pour dissimuler la source de la communication. En fait, le paquet émis est construit depuis une machine X mais avec, en guise d'adresse source, celle de la cible.

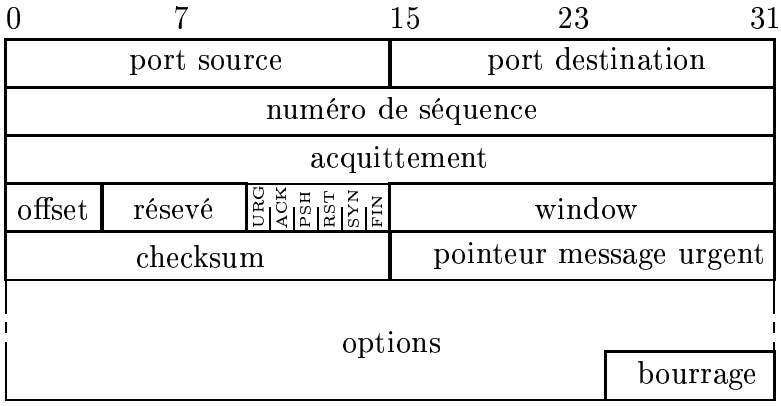


FIGURE 6.2 : En-tête TCP

Cependant, chacune de ces méthodes demeure détectable. Le champ `id` caractérise de manière unique les paquets émis par un serveur de sorte à ce qu'ils soient ré-assemblés en cas de fragmentation lors du transport. Selon les systèmes d'exploitation (*Operating System* - OS), ce champ n'est pas géré de manière identique : à chaque nouveau paquet émis, certains OS y ajoutent simplement $+1$ alors que d'autres utilisent un incrément aléatoire (cf. tab. 6.1).

OS	**Incrément**
Linux $\leq$ 2.2	$+1$
Windows 9x/Me	$+1$
Windows NT	$+256$
OpenBSD	pseudo-aléatoire

TABLEAU 6.1 : Variations du champ `id` pour différents OS

Ainsi, lorsque l'OS est connu[6], l'analyse des valeurs prises par le champ `id` montre une différence entre des paquets normaux et ceux modifiés. Mais surtout, ce champ étant nécessaire à la reconstruction des paquets fragmentés, l'unicité de sa valeur pendant un intervalle de temps suffisamment grand est indispensable au bon fonctionnement des communications. Par conséquent, cette contrainte interdit la répétition de caractères identiques dans le message, sous peine de rendre le canal détectable, et donc inefficace.

Les paquets utilisés dans la deuxième technique positionnent le bit `SYN` au niveau TCP. Normalement, une connexion TCP est initialisée de la manière suivante :

Ainsi, lorsque la machine intermédiaire examine le paquet, elle constate que son origine est la machine cible et lui répond donc, servant à faire «rebondir» le paquet.

[6]De nombreuses techniques, appelées *OS fingerprinting* permettent d'identifier un OS et sa version ; rien à voir avec le fingerprinting défini en dissimulation d'information.

1. le client envoie un paquet avec un numéro de séquence (ISN) x et positionne le bit `SYN` à 1 ;
2. le serveur répond avec comme numéro d'acquittement $x+1$ (ce qui signifie qu'il a bien reçu le paquet précédent et qu'il attend le suivant) et donne son propre numéro de séquence y. Les bits `SYN` et `ACK` sont positionnés à 1, le premier pour signifier le début de la connexion dans le sens serveur → client, le second pour confirmer l'ouverture de la connexion dans le sens client → serveur[7] ;
3. le client répond avec un paquet contenant comme numéro d'acquittement $y+1$ et le bit `ACK` à 1. Sur le serveur, le port destination est ouvert.

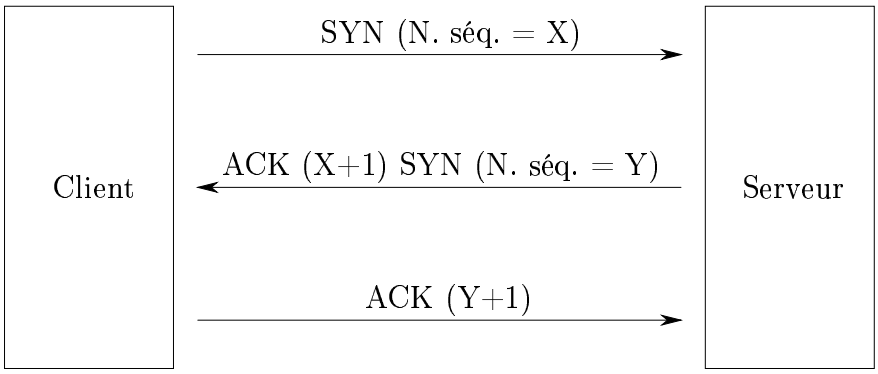


FIGURE 6.3 : *3 ways handshake* pour initier une connexion TCP

La solution proposée consiste à n'émettre que des paquets `SYN` en modifiant le numéro de séquence initial ISN pour qu'il corresponde à une lettre. La machine destination répond avec un paquet `SYN-ACK`, mais rien ne vient terminer l'initialisation de la connexion. Ce comportement est encore anormal. Pire, la machine cible peut même considérer cette communication comme une attaque puisque c'est exactement le schéma du *déni de service* appelé *SYN flooding* : ouvrir beaucoup de connexions TCP consomme des ressources sur le serveur, ce qui peut conduire à une paralysie complète de celui-ci.

Enfin, la troisième méthode soulève d'autres problèmes. Tout d'abord, elle suppose que la machine source construit des paquets en remplaçant l'adresse source (normalement la sienne) par celle de la machine cible (cette technique s'appelle du *spoofing*) afin que la machine «rebond» réponde à la cible et non à la source. Toutefois, la plupart des routeurs bloquent les paquets dont l'adresse source n'est pas sur le réseau local, ce qui limite donc d'autant ce schéma. Par ailleurs, recevoir des paquets `SYN-ACK` sans avoir reçu de `SYN` au préalable n'est pas normal. Cela est donc détectable au niveau de la machine cible. Mais il existe également une incidence au niveau réseau : à ces paquets, la machine répond par des `RST` (*reset*), ce qui génère des paquets sur le réseau.

[7] On appelle ce second paquet `SYN-ACK`.

Comme nous venons de le montrer, ces méthodes génèrent toutes un trafic suspect sur le réseau, et ne sont donc pas applicables pour la stéganographie. Il apparaît clairement qu'il faut éviter de générer ce trafic inutile.

6.1.3 Utilisation de données «délaissées»

Dans [HI96], les auteurs passent en revue les sept couches du modèle OSI à la recherche des octets non utilisés dans les en-têtes des paquets, comme par exemple les six octets réservés dans l'en-tête TCP (voir fig. 6.2 page 163). Les IDS actuels deviennent très performants et la modification de ces champs serait facilement détectable.

Il est nécessaire de dissimuler le canal dans une communication déjà établie, ou tout du moins «naturelle». L'utilisation de paquets laissant de la place pour des données apparaît alors comme une solution. Nous montrons dans cette partie que cela ne constitue toutefois pas une solution satisfaisante.

Par exemple, les messages `echo-request` du protocole ICMP[8] peuvent contenir n'importe quoi en guise de données. Toutefois, si un attaquant écoute les communications sur le réseau, il remarquera que ces paquets transportent des données, qu'elles soient diffusées en clair ou en chiffré. Par ailleurs, cela ne permet pas de communication importante. En effet, une communication de plusieurs centaines d'échanges `echo-request` / `echo-reply` est fortement suspecte car la commande `ping` sert habituellement à émettre juste quelques paquets pour tester si une machine est toujours active sur le réseau. Enfin, signalons que de plus en plus d'IDS ou de pare-feu surveillent le contenu de ces paquets car ils peuvent servir à contrôler à distance un système compromis.

Nous avons donc étudié quelques unes des possibilités offertes par les protocoles de niveaux 3 (IP et ICMP) et 4 (TCP) sans trouver de solution convaincante. Le protocole IP est lié à l'adressage. Toute modification d'un de ces champs semble étrange dès l'émission du paquet car celui-ci ne correspond plus à l'état du réseau local (comme une adresse source n'y appartenant pas). De plus, beaucoup de champs de l'en-tête IP peuvent être modifiés par les routeurs que le paquet traverse. Dans le cas de TCP et d'ICMP, du trafic anormal apparaît sur le réseau.

Nous n'avons pas abordé le protocole UDP, également au niveau 4. Toutefois, celui-ci semble plus propice à une telle communication car il ne fonctionne pas en mode connecté et n'appelle donc aucune réponse lorqu'un paquet parvient à la cible. Néanmoins, ce protocole n'est pas fiable et ne dispose d'aucun contrôle d'erreur, ce qui s'avère gênant lorsque les données à transmettre sont précises. On pourrait envisager de multiplier le nombre de paquets contenant le message et d'utiliser des techniques avancées de correction d'erreurs pour palier ce problème.

Ces niveaux ne semblent donc pas propices à l'installation d'un canal stéganographique. Nous avons donc décidé de remontrer dans le classique modèle

[8]Ces messages servent à la commande `ping` bien connue. La machine cible répond par une paquet ICMP de type `echo-reply`.

OSI jusqu'au niveau 7, aussi appelé *applications*.

6.1.4 Passage à la couche «applications»

Puisque générer du trafic supplémentaire sur le réseau est exclu et que nous voulons mettre en place une communication fiable, il ne nous reste plus qu'à chercher un canal stéganographique dans une transmission existante.

Parmi les serveurs disponibles au niveau «applications» (telnet, FTP, X-window, finger, NFS...), celui considéré comme le plus sûr est SSH.

Toutefois, dans la description de la couche transport de SSH, les auteurs de la norme [SSL] écrivent :

> `The protocol was not designed to eliminate covert channels. For example, the padding, SSH_MSG_IGNORE messages, and several other places in the protocol can be used to pass covert information, and the recipient has no reliable way to verify whether such information is being sent.`

6.2 Présentation du protocole SSH

Dans cette section, nous allons étudier les possibilités offertes par SSH pour dissimuler une communication, et donc créer ainsi un *canal stéganographique*. Nous commençons par présenter succinctement ce protocole qui s'appuie sur le protocole de transport TCP.

Initialement, le protocole *SSH* (*Secure SHell*) a été conçu pour remplacer *rsh* (*Remote SHell*) et *rlogin* (*Remote login*). En effet, si ces deux derniers protocoles permettent aisément d'exécuter des commandes sur une machine distante, leur niveau de sécurité est quasiment nul :

- l'authentification des entités reposent sur les adresses IP des machines. Or, ces adresses ne constituent pas une preuve d'identité. Le serveur est ainsi vulnérable aux attaques dites de *spoofing* par lesquelles un attaquant substitue l'adresse IP d'une machine et se fait ainsi passer pour un client valide auprès du serveur[9] ;
- les données transmises sur le réseau (*i.e.* dans le canal de communication) circulent en clair, mot de passe compris. Par conséquent, un attaquant à l'écoute de la transmission (attaque par *sniffing*) récupère à la fois les données du client et son mot de passe.

Pour remédier à ces failles, le protocole SSH repose sur plusieurs techniques cryptographiques avancées :

- authentification forte fondée sur des algorithmes à clés publiques, elle est disponible à la fois pour les machines et les utilisateurs ;
- chiffrement des données à l'aide d'un algorithme à clé secrète.

[9] Ceci fonctionne si un fichier `.rhost` approprié est présent sur le serveur. Dans le cas contraire, un mot de passe est nécessaire.

Il existe deux versions (respectivement appelées *v1* et *v2*) de ce protocole (voir les normes établies par l'IETF [SSL]). Toutes deux comportent de nombreuses différences dont les principales sont présentées dans le tableau 6.2.

	SSHv1	**SSHv2**[10]
Intégrité	CRC32	négociée entre SHA1 et MD5
Clé(s) de session	transmise par le client et valable pendant toute la durée de la communication (identique pour les deux sens de la transmission)	négociées avec un protocole Diffie-Hellman et renouvelées régulièrement (unique pour chaque sens de la transmission)
Authentification	RSA	RSA et DSA
Chiffrement	3DES, blowfish	3DES, blowfish, idea, AES, ...

TABLEAU 6.2 : Différences majeures entre SSHv1 et SSHv2

À noter également qu'une seule clé de session est négociée entre le client et le serveur en SSHv1, celle-ci servant à chiffrer les communications dans les sens client → serveur et serveur → client. Au contraire, avec SSHv2, chaque sens est considéré indépendamment de l'autre. Ainsi, dans le sens client → serveur, le chiffrement peut être du 3DES et l'intégrité vérifiée avec la fonction MD5 pendant que le sens inverse serveur → client chiffre avec l'algorithme blowfish et vérifie l'intégrité avec SHA.

Comme l'utilisation de SSHv1 est fortement déconseillée pour des raisons de sécurité, nous nous focaliserons principalement dans la suite sur la version 2.

paquet vs. message Dans la suite, nous désignerons par *paquet* ou *datagramme* les informations transmises sur le réseau et par *message* le contenu d'un paquet associé au protocole SSH. Comme nous l'avons vu en 6.1.1.2, les messages SSH sont encapsulés dans les paquets TCP/IP.

6.2.1 Messages SSH

Un message est composé de différents champs, positionnés à des endroits précis, afin que le client et le serveur puissent les interpréter identiquement. Dans le cas de SSHv2, il contient les champs suivants (voir fig. 6.4 page suivante) :

- *taille du message* : donne la longueur du message en octet, sans les parties MAC et *taille du message* elle-même (codée sur 4 octets) ;
- *taille du bourrage* : donne en nombre d'octets la longueur du bourrage[11] (codée sur 1 octet) ;
- *données* : les données «utiles» du message (codées sur (*taille du message* - *taille du bourrage* - 1) octets) ;

[10] On se limite ici aux algorithmes les plus connus.

[11] Le bourrage sert à «remplir» un message de sorte à ce qu'il ait une taille voulue.

- *bourrage* : octets générés aléatoirement afin que la concaténation du bourrage et des trois champs précédents occupe une place dont la taille soit égale à $max(8,$ *taille des blocs de chiffrement*$)$ (occupe de 1 à 255 octets) ;
- *MAC* (*Message Authentication Code* - cf. B.3) : présent uniquement si il a été négocié pendant l'initialisation de la transmission (la taille dépend alors de la fonction de hachage choisie).

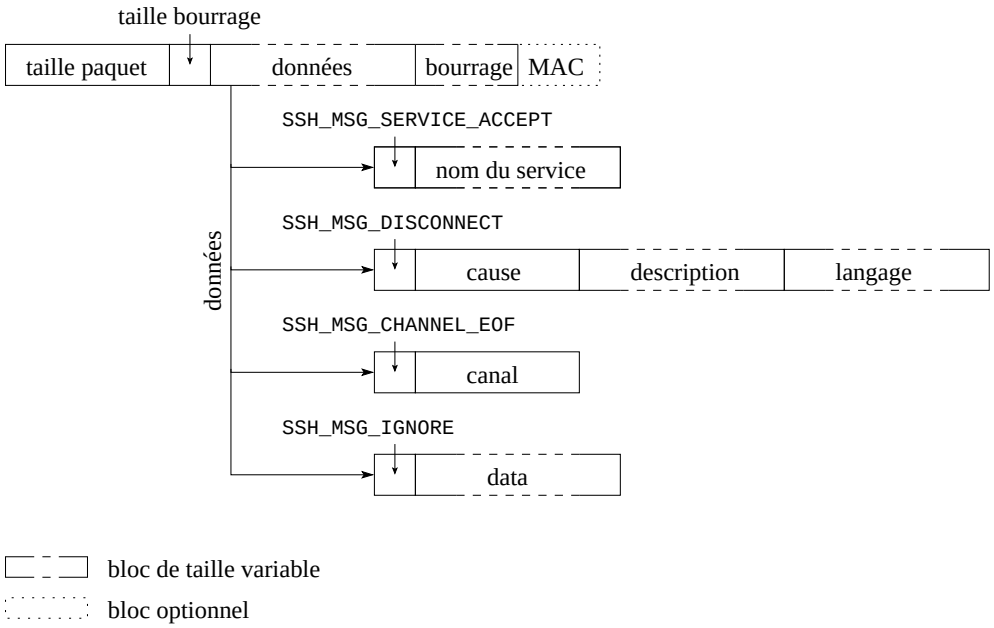


FIGURE 6.4 : Messages SSH

Lors de la communication, les quatre premiers champs sont chiffrés avec la clé de session. Dans [DS01], les auteurs montrent qu'il est possible de deviner la taille exacte du mot de passe pour SSHv1[12], et un intervalle de tailles probables pour SSHv2.

6.2.2 Exemples de messages

Tout au long de la communication, le champ *données* contient des informations différentes selon le but du message : échange de clés, transfert de données, ouverture de session, identification d'utilisateur... À chacune des actions possibles est associée un nombre appelé *type*.

Le type des données est codé sur un octet, la suite dépendant alors de la nature du message. Pour chaque type, le contenu possède donc une structure différente, ce qui est naturel puisque signaler l'ouverture d'une connexion n'a

[12]Entre autre parce que SSHv1 ne chiffre pas le champ *taille du message*, au contraire de SSHv2.

Type	Action
SSH_MSG_SERVICE_ACCEPT	acceptation d'un service (ssh-userauth ou ssh-connection)
SSH_MSG_DISCONNECT	provoque la fin immédiate de la connexion
SSH_MSG_CHANNEL_OPEN	demande l'ouverture d'un nouveau canal
SSH_MSG_CHANNEL_EOF	ferme un canal
SSH_MSG_USERAUTH_REQUEST	demande d'identification de l'utilisateur
SSH_MSG_KEXINIT	initialise l'échange de clés
SSH_MSG_IGNORE	données ignorées par le receveur

TABLEAU 6.3 : Exemples de messages de contrôle SSH

aucun rapport avec l'échange des clés de session. Ils sont placés dans le champ *données*, présenté en 6.2.1 page 167.

6.3 Utilisation détournée du message SSH_MSG_IGNORE

D'un point de vue logiciel, il existe de nombreuses versions différentes de SSH. Nous n'avons travaillé qu'avec celle construite pour OpenBSD portée sous Linux : OpenSSH. En particulier, nous avons utilisé la dernière version disponible au moment de la rédaction de ce mémoire, la 2.9p2.

6.3.1 Le message SSH_MSG_IGNORE

Le protocole SSH décrit ainsi ce message :

```
Ignored Data Message

  byte      SSH_MSG_IGNORE
  string    data

  All implementations MUST understand (and ignore) this message at any
  time (after receiving the protocol version). No implementation is
  required to send them. This message can be used as an additional
  protection measure against advanced traffic analysis techniques.
```

Comme nous l'avons déjà évoqué en 6.2.1, il est possible d'obtenir des informations sur la taille des mots de passe par une analyse passive du trafic (voir [DS01]). Il en va de même avec les commandes passées lors d'une session interactive. Une étude parallèle présentée dans [SWT01] montre même qu'il est possible d'en deviner encore plus à l'aide d'une analyse statistique fondée sur la fréquence des échanges des paquets entre le client et le serveur.

Pour remédier à cela, les versions actuelles de SSH utilisent ce message. Lors de l'identification par mot de passe, le client construit un message SSH_MSG_USERAUTH_REQUEST puis un autre de type SSH_MSG_IGNORE de sorte

que la somme de leurs tailles reste constante. Ils sont ensuite transmis à la couche transport qui les fait passer sur le réseau dans un unique paquet.

Par ailleurs, lorsque l'utilisateur entre des commandes dans un shell au travers de SSH, le serveur répond à chaque caractère par un petit paquet appelé *écho*. Une fois l'instruction terminée, le paquet suivant est beaucoup plus gros car il contient la réponse à la commande. Il est ainsi possible de deviner la longueur des commandes. Cependant, lorsque le caractère saisi n'apparaissait pas à l'écran[13] (mode `echo off`), le paquet écho n'était pas retourné au client. Il était donc possible de deviner que la commande entraînait la demande d'un mot de passe et la longueur de celui-ci simplement en comptant le nombre de paquets émis du client vers le serveur restant sans réponse. Pour remédier à ce problème, lors du fonctionnement en mode `echo off`, un message `SSH_MSG_IGNORE` est émis.

Ainsi, le message `SSH_MSG_IGNORE` ne sert que dans deux occasions :

1. lors de l'authentification pour dissimuler la longueur du mot de passe ;
2. en mode `echo off` pour cacher le nombre de caractères ainsi saisis.

6.3.2 Canal stéganographique avec les messages `SSH_MSG_IGNORE`

Le protocole mis en place par l'IETF ne stipule rien de particulier quant au contenu des messages de type `SSH_MSG_IGNORE`. Dès lors, il est possible d'y mettre n'importe quoi.

Toutefois, cette solution comporte deux inconvénients :

1. manque de fiabilité : puisque ces messages doivent être ignorés par le destinataire, l'émetteur ne peut savoir s'ils sont bien arrivés. Nous nous retrouvons donc dans une situation similaire à celle des paquets UDP ;
2. trafic réseau exceptionnel : tout comme avec les paquets `echo-request`, les messages `SSH_MSG_IGNORE` sont rares. Cependant, à la différence des paquets ICMP, l'attaquant n'est pas capable de distinguer qu'il s'agit d'un message `SSH_MSG_IGNORE` puisque le type de celui-ci est chiffré.

Nous avons vu que ces messages ne servaient que dans deux occasions. À titre d'exemple, voici les statistiques sur ces paquets issues de la connexion suivante :

[13] Ceci se produit par exemple lorsque l'utilisateur invoque la commande `su` dans le shell distant.

Session ssh utilisant le mode `echo off`

```
[client]$ ssh serveur
raynal@serveur password:
[serveur]$ su
Password:
su: incorrect password
[serveur]$ logout
Connection to minimum closed.
RECV SSH2_MSG_IGNORE:       8 pkt ->    72 bytes
SEND SSH2_MSG_IGNORE:       1 pkt ->    48 bytes
[client]$
```

Le client n'a émis qu'un seul message, celui prévu lors de l'authentification. Le serveur a envoyé 8 messages : le mot de passe saisi comportait donc 7 caractères, le dernier étant le retour chariot validant la ligne.

Le total d'octets échangés n'est pas très important : 48 octets pour le client contre 72 pour le serveur. Cependant, il est possible que les versions futures de SSH fassent une plus grande utilisation de ces messages puisqu'ils permettent de réduire les résultats obtenus en analysant le trafic. Dès lors, l'intérêt de ce canal grandirait.

Actuellement, ce canal est utilisable pour transférer de très courtes données. Avec les versions courantes du protocole, on pourrait forcer l'envoi de tels messages. Toutefois, ceux-ci n'engrendrant aucune réponse du destinataire, une analyse du trafic pourrait en révéler l'existence :

- soit ces messages sont envoyés dans un seul paquet, et aucune réponse ne revient ;
- soit ils sont accolés à d'autres messages, comme pour `SSH_MSG_USERAUTH_REQUEST`, mais dans ce cas, la taille des donnés des paquets augmente anormalement.

Dans le second, l'analyse du trafic nécessite beaucoup de données pour parvenir à différencier les paquets contenant entre autre un message `SSH_MSG_IGNORE`, ce qui semble assez difficile à réaliser en pratique.

6.4 Bourrage et canal caché

L'utilisation des messages de type `SSH_MSG_IGNORE` n'est pas satisfaisante car ces messages sont encore trop rares. Cependant, SSH n'est pas sans ressource et une autre possibilité s'offre à nous.

6.4.1 Comment utiliser le bourrage

Le protocole défini par l'IETF impose que le bourrage présent dans les messages SSH soit aléatoire. Nous devons donc chercher à remplacer cet aléa par un autre que nous serons capables d'interpréter.

Une des contraintes des algorithmes de chiffrement est justement de faire ressembler un message chiffré à de l'aléa. Nous pouvons donc remplacer les octets de bourrage par un message chiffré. Deux utilisations de ce canal caché, correspondant à des objectifs antagonistes, sont envisageables.

6.4.2 Utilisation du bourrage par un pirate

Un pirate souhaitant espionner la conversation entre le client et le serveur utilise ce canal pour récupérer la clé de session. On note E l'algorithme de chiffrement négocié.

D'un point de vue pratique, pour réussir, un attaquant doit remplacer une des extrémités de la communication par sa propre version du logiciel (*i.e.* le client ou le serveur), ce qui nécessite donc qu'il parvienne au préalable à pirater au moins une des machines impliquées. Avec SSHv1, la clé de session étant partagée entre le client et le serveur, le pirate n'a besoin de corrompre qu'un seul des deux programmes. En revanche, avec SSHv2, le client et le serveur disposent chacun de leur propre clé de session : la compromission d'une des extrémités du canal ne permet alors que de déchiffrer la moitié de la communication. Pour déchiffrer l'intégralité, l'attaquant doit corrompre le client **et** le serveur.

L'attaque se déroule alors de la manière suivante :

1. une fois la clé de session k établie, elle est découpée de manière à occuper la place réservée au bourrage dans les prochains messages émis (soient k_i ces «morceaux de clé») ;
2. ces octets sont chiffrés à l'aide de la clé de session et de la fonction de chiffrement E_k^{-1} puis placés à la fin du message ;
3. l'ensemble du message est chiffré avec la clé de session et le chiffrement E. À la place des octets de bourrage, on a donc $E_k(E_k^{-1}(k_i))$.
   Or, l'algorithme étant symétrique, la clé k sert aussi à déchiffrer :

   $$E_k(E_k^{-1}(k_i)) = k_i$$

   Le bourrage contient donc le «morceau de clé» en clair ;
4. l'attaquant écoute sur le réseau, récupère les paquets qui transitent et reconstruit la clé à partir des différents morceaux. Ensuite, il n'a plus qu'à déchiffrer les données contenues dans les messages.

Avec cette attaque, Charlie parvient à espionner intégralement la conversation si le protocole SSHv1 est utilisé. En effet, celui-ci utilise la même clé de session pour chiffrer les deux sens de la transmission. Cette attaque ne nécessite donc pas de corrompre uniquement le client ou le serveur, un seul des deux suffit. En revanche, avec SSHv2, comme chaque sens est indépendant et gère sa propre clé de session, Charlie doit corrompre à la fois le client et le serveur pour récupérer les deux clés.

Du point de vue de l'utilisateur et au niveau réseau, cette attaque est difficilement détectable. Il faudrait vérifier que les octets de bourrage ne correspondent

	serveur → client		client → serveur	
	#paquets	bourrage	#paquets	bourrage
Copie d'un fichier de 1Ko	15	175 o	14	137 o
Copie d'un fichier de 10Ko	15	176 o	16	171 o
Copie d'un fichier de 100Ko	20	266 o	40	586 o
Copie d'un fichier de 1Mo	65	1.51Ko	265	4.54Ko
Copie d'un fichier de 10Mo	505	8.804Ko	2515	44.78Ko
Copie d'un fichier de 100Mo	4845	85.79Ko	25907	445.237Ko

TABLEAU 6.4 : Capacité du canal utilisant le bourrage (o = octet)

pas à des octets de la clé de session à chaque message[14]. De plus, l'enregistrement de tous les paquets de la communication est impératif car un attaquant peut très bien enregistrer la communication chiffrée et envoyer des octets de la clé dans des paquets à n'importe quel moment de la connexion pour ne déchiffrer celle-ci qu'une fois qu'elle est achevée.

6.4.3 Utilisation du bourrage par une entité authentifiée

Supposons maintenant que soit le client, soit le serveur emploie ce canal. Tous deux négocient une clé de session k. Toutefois, considérons que les deux parties ont négocié préalablement par le biais d'un autre canal considéré comme sécurisé (*One Time Password*, échange «de main à main»...) une clé de session k'.

Il est alors possible de faire transiter des données, chiffrées avec k', dans le bourrage : l'émetteur remplace les octets de bourrage par ceux résultant du chiffrement du message qu'il souhaite faire passer par le canal stéganographique. Ceux-ci ressemblent effectivement à de l'aléa si l'algorithme de chiffrement est efficace. Le message est ensuite chiffré à l'aide de la clé k puis transmis.

D'un point de vue pratique, cette opération nécessite la modification du client et du serveur afin qu'ils sachent distinguer les octets de bourrage normaux de ceux contenant de l'information. Néanmoins, Charlie qui écoute la communication est incapable de faire la distinction entre l'information chiffrée et le bourrage aléatoire. De plus, aucun trafic anormal n'est généré sur le réseau puisque la session SSH est légitime.

Se pose la question de la capacité de ce canal. Pour l'estimer, nous avons construit un fichier contenant des caractères générés aléatoirement puis l'avons transféré du client au serveur. Le tableau 6.4 donne les résultats obtenus.

[14] Ceci suppose également que la clé de session est connue d'un tiers pour effectuer cette vérification...

6.5 Conclusion : SSH contient bien (au moins) un canal stéganographique

Nous avons montré l'existence de deux canaux cachés dans le protocole SSH. Si celui fondé sur l'emploi des messages `SSH_MSG_IGNORE` est théoriquement fonctionnel, la faible utilisation de ces paquets le rend assez peu confortable. Il est cependant probable que ces messages soient plus nombreux à l'avenir afin de déjouer les attaques par analyse de trafic.

En revanche, le bourrage fournit une solution déjà exploitable pour transférer des données. Ce canal possède les caractéristiques suivantes :

1. indétectabilité : la communication prend place dans une autre, légitime. Du point de vue de l'attaquant qui écoute passivement, il ne peut pas distinguer les octets aléatoires de ceux du canal puisqu'ils semblent eux-mêmes aléatoires.
2. fiabilité : SSH repose sur le protocole TCP qui garantit que tous les messages arrivent bien à destination. De plus, SSH fournit un test d'intégrité pour toutes les données du message, bourrage compris. Il comprend donc un moyen de vérifier que nos données dissimulées n'ont pas été modifiées[15], ce qui nous protège également d'un attaquant actif puisque toute intervention de sa part est immédiatement détectée.
3. capacité : chaque message ne peut contenir qu'entre 1 et 255 octets de bourrage (1 et 8 pour SSHv1). Cependant, une connexion SSH peut durer un temps illimité sans paraître anormal, comme dans le cas des tunnels par exemple.

Par rapport aux différents modèles de stéganographie présentés en introduction de ce mémoire, ce schéma s'accorde parfaitement avec ceux à clé publique. En effet, une personne souhaitant utiliser le bourrage doit chiffrer les données qui y sont placés. Pour cela, autant se servir de la clé publique du destinataire afin que lui seul soit capable de déchiffrer le message caché.

[15] En fait, on ne peut pas savoir si ce sont les données normales ou le bourrage qui a été modifié puisque le haché les comprend tous les deux.

Conclusion et perspectives

Dans cette seconde partie, nous avons présenté un outil d'évaluation automatique et configurable pour les multiples schémas de tatouage existant. Si l'intégralité de l'application, avec son interface et ses graphiques, n'est pas encore terminée, le programme est néanmoins utilisable. Il enrichit les fonctionnalités déjà présentes dans `Stirmark`. En particulier, des tests plus axés sur le protocole commencent à voir le jour (espace des clés ou marquages multiples par exemple). L'étape suivante sera probablement d'intégrer des attaques protocolaires, comme celle par recopie.

Nous avons également exploré les relations entre la cryptographie et la dissimulation d'information. Cela nous a permis d'étudier en quoi ces relations sont pertinentes ou non :

- les problèmes liés à la gestion des clés (usage multiple ou cession des droits) sont rarement étudiés car celle-ci est souvent un nombre aléatoire ;
- la «dissimulation asymétrique» semble offrir une solution confortable, surtout pour le tatouage. Pour la stéganographie, une étape préalable s'impose entre les participants pour s'accorder sur la communication. Pour le fingerprinting, une des clés doit être multiple ;
- l'intégrité du stégo-médium repose sur la notion de hachage mou ;
- les schémas reposant sur une preuve à divulgation nulle de connaissance offre un niveau de sécurité supplémentaire en limitant l'information disponible pour monter une attaque ;
- la non répudiation des schémas de signatures est soit souhaitée (stéganographie), soit proscrite (tatouage), soit indispensable (fingerprinting) ;
- le partage de secret constitue une base intéressante pour élaborer des schémas de tatouage ou fingerprinting résistants au recadrage.

Nous avons en particulier montré comment certains aspects de la cryptographie pouvaient être directement intégrés dans un schéma de dissimulation d'information, ou bien pris en considération dans le protocole entourant le schéma.

Enfin, dans un dernier chapitre, nous avons étudié les possibilités offertes sur un réseau pour dissimuler une communication. Les rares études antérieures proposaient d'utiliser certains champs des différents protocoles. Nous avons montré en quoi ces solutions ne peuvent plus fonctionner actuellement. En revanche, le protocole SSH, en plein essor, comporte deux canaux cachés. Nous avons présenté l'utilisation qui peut en être faite selon les objectifs poursuivis, ainsi qu'une estimation de la capacité de ce canal.

Chapitre 7

Conclusion Générale

7.1 Bilan

Dans le chapitre 2, nous avons rappelé les différentes problématiques qui régissent la dissimulation d'information, puis présenté chaque domaine :

- la stéganographie dont l'objectif est de créer un canal caché ;
- le tatouage dont l'application principale est la protection des droits d'auteur ;
- le fingerprinting qui cherche à détecter l'origine de copies illégales.

Divers aspects liés à la dissimulation d'information ont été étudiés dans ce mémoire.

Nous nous sommes intéressés tout d'abord à deux outils issus de la théorie des fractales. Au chapitre 3, nous sommes partis d'une analyse approfondie du schéma proposé dans [PJ96]. Cette solution est en fait un problème de génération d'IFS (*Iterated Functions Systems*) sous contraintes que nous avons traité dans la suite du chapitre. Notre contribution principale est double.

D'une part, l'introduction des IFS polaires, reposant sur des fonctions localement contractantes vis-à-vis d'un point fixe, améliore la qualité de l'espace de recherche. En effet, les IFS mixtes, constitués de fonctions non linéaires, sont rarement contractants, au contraire des IFS polaires. D'autre part, nous avons modifié le processus classique d'un algorithme évolutionnaire pour l'adapter à des situations qui autorisent la reconstruction de la solution globale à partir d'un sous-ensemble de la population. L'avantage en est que les individus de la population ne sont plus mis au rebut dès qu'un de leurs chromosomes ne convient pas.

Ensuite, au chapitre 4, nous avons mené des expériences afin d'évaluer la pertinence d'un schéma de tatouage reposant sur les spectres multifractals mutuels. Malheureusement, nous n'avons pu déterminer de critères suffisamment distinctif pour différencier les mesures de références à l'aide du spectre. Néanmoins, nous avons retenu de cette démarche le protocole expérimental.

Celui-ci nous a permis de mettre au point un logiciel d'évaluation pour les schémas de tatouage (chap. II). Construit à partir de `Stirmark`, notre nouvel

outil intègre non seulement les fonctionnalités de son prédécesseur, mais ajoute également de nouveaux tests. L'architecture que nous avons mis en place a permis le développement parallèle de tests tant pour les images que pour les fichiers audios. L'autre innovation de `StirMark Benchmark` est l'ajout de tests rarement utilisés dans les publications, comme l'emploi de différentes clés pour une même marque, le marquage multiple ou une estimation des taux de fausses alarmes.

Pour mettre au point ce logiciel, nous avons étudié les multiples types de schémas de tatouage qui existent. Pour élaborer les nouveaux tests concernant des problèmes moins liés à la robustesse du schéma, nous nous sommes alors intéressés aux liens entre la cryptographie et la dissimulation d'information (chap. 5). Nous avons montré en quoi les solutions proposées en cryptographie pouvaient être intégrées soit dans le schéma lui-même, soit dans le protocole qui l'entoure. Nous avons proposé de nombreuses directions à explorer afin d'améliorer la sécurité des applications de dissimulation d'information.

Enfin, le dernier chapitre de ce mémoire porte sur la stéganographie et son utilisation sur un réseau. Après avoir présenté les solutions antérieures, nous avons montré que la couche «Applications» du modèle OSI offrait un terrain propice. Nous avons montré que le protocole SSH contient deux canaux cachés qu'il est possible d'exploiter de plusieurs manières.

7.2 Perspectives

La dissimulation d'information est une discipline paradoxale. Les premières applications de tatouage commencent à voir le jour car le besoin devient de plus en plus important. Certains *majors* du disque annoncent que leurs CDs sont maintenant protégés contre la copie. Des firmes vendent de la musique à écouter sur un téléphone portable, ce qui nécessite une protection afin d'éviter que les téléphones mobiles ne puissent devenir des sources de piratage. Dans le même temps, les évènements qui se sont produits aux États-Unis en Septembre 2001 laissent planer un sentiment de suspicion quant à l'emploi de la stéganographie. En effet, les terroristes sont soupçonnés d'avoir utilisés cette technique pour communiquer. Certains pays veulent restreindre son utilisation, tout comme celle de la cryptographie. Les personnes malveillantes qui veulent utiliser ses techniques ne se préoccupent qu'assez peu des lois en vigueur. On peut alors douter qu'elles respectent une telle interdiction. Or, les recherches en tatouage rejaillissent en stéganographie, et réciproquement.

Le domaine du tatouage est encore jeune, au contraire de la cryptographie qui bénéficie de longues années de recherche. Depuis ses débuts, les études ont principalement été orientées sur les schémas eux-mêmes, et en particulier sur l'amélioration du compromis entre la robustesse, l'indétectabilité et la capacité. Avec le besoin croissant d'applications utilisant le tatouage, il nous apparaît comme essentiel de commencer à se préoccuper des protocoles qui entourent les solutions proposées. Cela passe tout d'abord par une définition claire et précise

des propriétés requises pour chaque application afin d'en évaluer exactement la sécurité. Pour cela, il est important de continuer à explorer les voies ouvertes dans ce document en analysant l'apport de la cryptographie à la dissimulation d'information. Il nous semble en effet actuellement inenvisageable de proposer un schéma de tatouage grand public fiable sans continuer à prospecter dans ce sens.

Dans cette perspective, deux orientations privilégiées se dégagent. Tout d'abord, l'élaboration de solutions asymétriques s'avère critique pour les applications commerciales. De tels schémas permettraient d'assouplir la gestion des clés et des données dissimulées, question encore trop rarement étudiée actuellement. Ensuite, les protocoles à divulgation nulle de connaissance constituent une autre direction à explorer pour la protection des droits d'auteur. La cryptographie fournit déjà de nombreux protocoles de ce type. Même si leur adaptation aux média requiert des précautions supplémentaires, l'apport en terme de sécurité des schémas est indéniable puisqu'ils restreignent l'information disponible sur les données dissimulées.

Par ailleurs, un nouvel axe de recherche concernant la stéganographie apparaît. La plupart des supports numériques utilisés sont des média. Il semble difficile, voire impossible, de tous les filtrer à la recherche de messages secrets. Cependant, le monde de l'informatique contient de nombreuses zones potentielles de dissimulation. Nous en avons présenté une ici à partir d'un canal caché dans le protocole SSH. À l'heure où les réseaux commencent à se répandre (e-buisness, vote électronique,...), ces canaux constituent une arme à double tranchant.

Annexe A

Introduction aux algorithmes évolutionnaires

Les algorithmes évolutionnaires (AE) sont des algorithmes d'optimisation stochastique. Ils imitent, en le simplifiant, le principe d'évolution des populations énoncé par Darwin. Les AE regroupent différentes tendances :

- les algorithmes génétiques (AG) ([Hol62]) ;
- les stratégies d'évolution (SE) ([Rec73]) ;
- la programmation évolutionnaire (PE) ([FOW66]) ;
- la programmation génétique (PG) ([Koz92]) ;

Ils simulent tous le processus d'évolution : une population, de taille fixe contrairement à ce qui se passe dans la nature, évolue afin d'adapter les individus qui la composent à leur environnement. Seuls les plus robustes survivent à la sélection naturelle et parviennent à se reproduire. La répétition de ce mécanisme pendant plusieurs générations aboutit à des individus qu'on espère être des optima pour la fonction à optimiser.

A.1 L'évolution

Le but d'un AE est de parcourir l'espace de recherche associé au problème afin de découvrir la (les) meilleure(s) solution(s) à un problème donné. Chaque point de cet espace est appelé *individu*. Une *population* est constituée d'un ensemble d'individus. Afin d'évaluer la pertinence d'une solution, on définit une *fonction de pertinence* (ou *fonction de fitness*) dont les optima correspondent aux meilleurs individus.

L'objectif d'un AE est alors de faire évoluer la population afin de converger vers un optimum. Pour y parvenir, les individus sont croisés et mutés selon des probabilités respectives fixées. Seuls les plus aptes se perpétuent dans la population suivante. Ce processus est répété pendant un nombre fixé de *générations* (cf. fig. A.1 page suivante) :

1. génération d'une population initiale, soit en créant aléatoirement les individus qui la composent, soit en y insérant quelques individus jugés inté-

ressant issus d'une évolution antérieure ;

2. des individus, les *parents*, sont sélectionnés, selon leur performance, afin d'être croisés et mutés, donnant ainsi naissance à des *enfants* ;
3. la nouvelle population est construite, soit uniquement à partir des enfants, soit à partir des meilleurs individus pris parmi les enfants et les parents ;
4. les deux étapes précédentes sont répétées jusqu'à ce qu'un optimum soit découvert ou une condition d'arrêt satisfaite (*e.g.* nombre maximal de générations ou un seuil de fitness atteint).

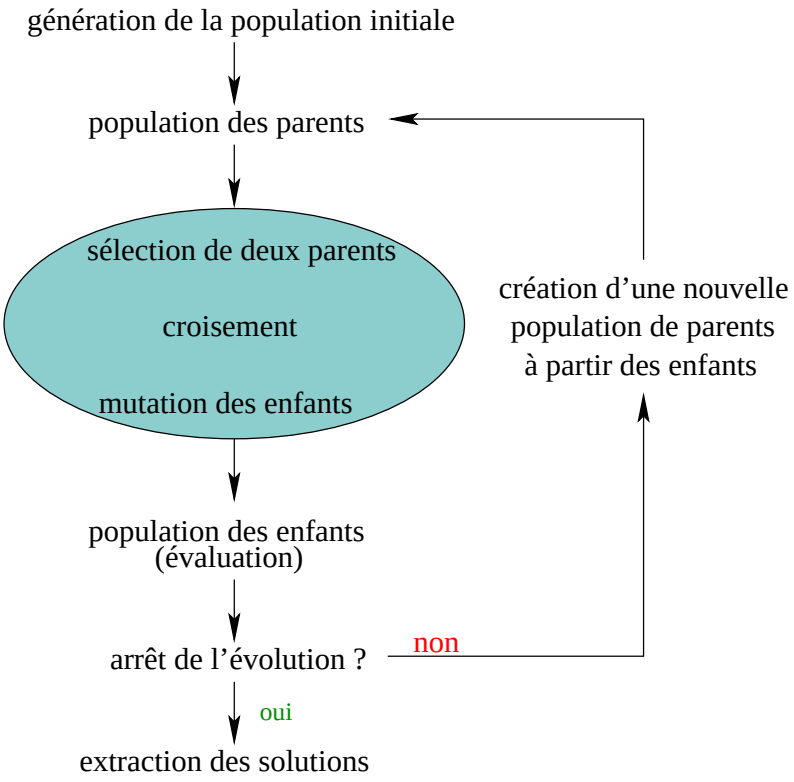


FIGURE A.1 : Évolution d'un AE classique

A.2 Codage des individus

L'espace de recherche ne coïncide pas toujours avec l'espace des solutions. Pour les distinguer dans la terminologie génétique, on appelle le premier *espace des génotypes* et le second *espace des phénotypes*.

Les individus sont constitués de différents *chromosomes*, chacun d'eux correspondant à une variable du problème. Ces chromosomes sont codés à l'aide de *gènes* qui appartient au génotype.

Bien que ce ne soit pas leur seule différence[1], le codage employé permet de distinguer les domaines évoqués en introduction de cette partie :

- les AG travaillent essentiellement sur des chaînes binaires ;
- les SE se servent de vecteurs de réels ;
- la PE se situe dans l'espace des automates à états finis puis ont été élargis afin de travailler également sur des réels ;
- la PG manipule des arbres, pouvant représenter des fonctions mathématiques ou de petits programmes.

Les exemples que nous donnons dans la suite de ce chapitre se focalisent sur les arbres et la programmation génétique afin de présenter l'outil dont nous nous sommes servi dans le cadre de la génération d'IFS sous contraintes.

La figure A.2 illustre la représentation d'une fonction sous forme d'arbre. Pour le décoder, on utilise une lecture infixe[2] des noeuds de l'arbre.

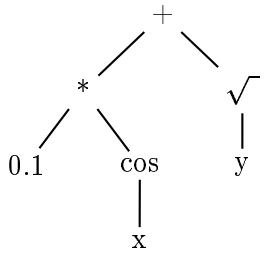


FIGURE A.2: Représentation sous forme d'arbre de la fonction $0.1 \times cos(x) + \sqrt{y}$

A.3 Opérateurs génétiques

L'évolution des populations successives se produit grâce à l'intervention d'*opérateurs génétiques* qui caricaturent le phénomène réel d'évolution :

- la *sélection* : elle a pour but de choisir dans la population courante des individus qui constituent des parents pour la génération suivante. La plupart du temps, les individus ont une probabilité d'être sélectionnés proportionnelle à leur fitness. Par exemple, la sélection par *roue de la fortune* (ou rws) alloue une aire sur le disque proportionnelle à la qualité de l'individu, puis un tirage uniforme sur le contour du disque désigne l'individu sélectionné (voir fig. A.3).

  Cette solution favorise les individus «trop bons», en particulier lorsqu'ils arrivent tôt dans l'évolution. Le *scaling* permet alors de limiter leur influence en maintenant une *pression sélective* C raisonnable au sein de la population. Celle-ci a pour objectif de limiter l'écart entre les bons individus de la population et les autres : $C = \frac{fit_{max}}{fit_{moy}}$. Pour cela, une méthode

[1] Les différences plus fondamentales liées aux mécanismes internes de ces algorithmes sortent du cadre de ce document.

[2] À chaque noeud, on lit le fils gauche, puis le noeud, puis le fils droit.

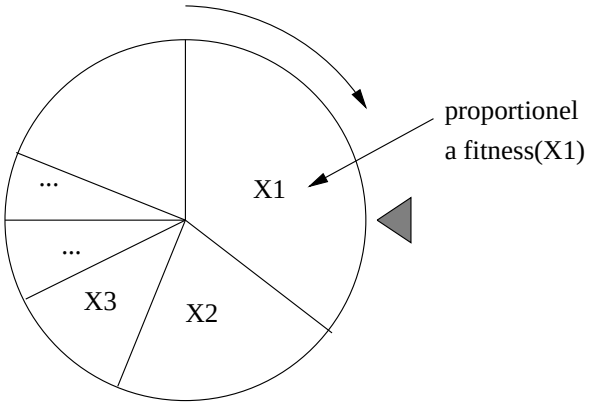


FIGURE A.3 : Sélection des parents : une loterie biaisée

courante est de faire subir une transformation linéaire sur la fonction fitness de manière à ce que le maximum des nouvelles valeurs de fitness observées dans la population soit C fois supérieur à la moyenne de ces valeurs. De manière générale, C est fixé entre 1.2 et 2.
La sélection par *ranking* repose également sur le principe de la «roue de la fortune». Toutefois elle diffère de la version classique par la probabilité qu'un individu a d'être choisi, car elle ne dépend plus que de son classement dans la population courante. La population de taille n est triée dans par ordre de valeur de fitness décroissante, la probabilité de sélection de l'individu situé en $i^{ème}$ position est fixée à :

$$p_i = k_1 + k_2 * (n - i),$$

où les paramètres k_1 et k_2 sont calculés en fonction de la pression sélective C, constante tout au long de l'algorithme.

- le *croisement* : il cherche à mélanger le patrimoine génétique de deux individus pour en engendrer un, deux ou plus.. Le but est qu'ils conserveront les «bons» gènes de chacun des parents pour qu'ils soient «meilleur(s)» que ceux-ci au sens de la fitness.
  Comme le montre la figure A.4, un emplacement est désigné aléatoirement sur les chromosomes et les deux «enfants» sont obtenus en échangeant les parties de chromosomes autour de ce site.
  Cet opérateur est appelé avec une certaine probabilité. S'il n'est pas appliqué, les deux individus préalablement sélectionnés sont alors directement soumis à l'opérateur de mutation. Lorsqu'il est appliqué, les enfants sont transmis à l'opérateur de mutation.
- la *mutation* : elle permet de modifier légèrement le génotype d'un individu avec une probabilité donnée. Chaque gène des chromosomes est soumis à cette mutation. Dans la cas d'un codage par des arbres, la mutation consiste simplement à inverser un bit, comme l'illustre la figure A.5.
- le *remplacement* : il détermine la manière dont est construite la popula-

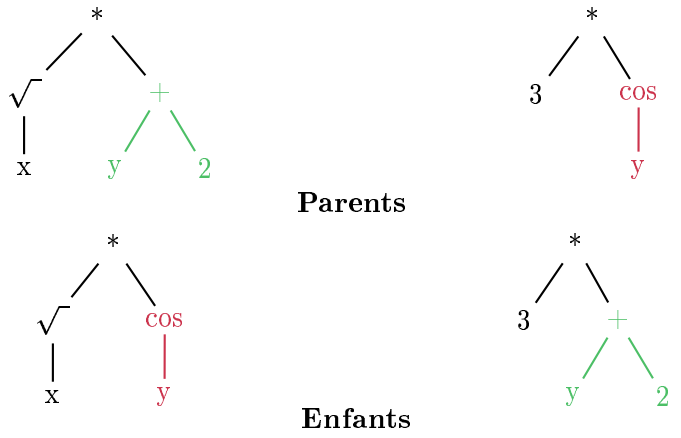


FIGURE A.4: Opérateur de croisement pour des arbres : échanges de deux sous-arbres entre les parents pour construire les enfants

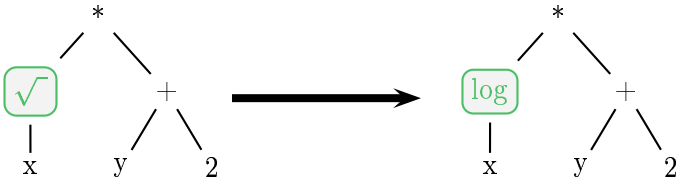


FIGURE A.5 : Opérateur de mutation d'un noeud en PG

tion pour la génération suivante (dite *population des enfants*). Il existe différentes politique, comme le remplacement total des parents par de nouveaux enfants, ou d'autres, plus protectrices à l'égard des meilleurs parents, puisqu'ils sont reportés à l'identique dans la population des enfants.

Par exemple, avec le remplacement par superposition de populations, on génère un nombre d'enfants qui correspond à un pourcentage de la taille de la population des parents. On mélange alors la population des parents et ces enfants nouvellement créés, puis on trie cet ensemble pour ne conserver que le bon nombre d'individus dans la population des enfants.

Le but de ces opérateurs est de parvenir à un compromis entre *exploitation* et *exploration*. L'exploitation cherche à utiliser au mieux l'information contenue dans les meilleurs individus afin de déterminer les nouveaux points à explorer dans le but d'améliorer les performances. Il s'agit donc d'une recherche locale dans le voisinage des meilleurs individus.

Néanmoins, puisque l'exploitation dirige la population vers le plus proche optimum local, elle menace la *diversité génétique* qui permet d'explorer différentes régions de l'espace de recherche.

A.4 Partage et nichage dynamique

Afin de préserver un compromis exploitation / exploration efficace, il est parfois utile de recourir à des techniques avancées.

Le partage de la fitness (ou *sharing*) a été formulé par Goldberg et Richardson en 1987 dans [GR87], à partir d'un parallèle biologique avec les niches écologiques. La fitness d'un individu est recalculée en fonction du nombre d'individus qui lui sont proches afin de pénaliser ceux trop semblables. Soient x un individu d'une population de taille n, dont les différents individus sont notés $x_i, 1 \leq i \leq n$, et $f(x)$ sa fitness, alors la fitness partagée $f_{sh}(x)$ est définie comme :

$$f_{sh}(x) = \frac{f(x)}{\sum_{i=1}^{n} sh(d(x, x_i))}$$

où sh désigne la fonction de partage, paramétrée par la distance entre deux individus. La fonction sh vaut 1 si les deux individus sont identiques, et 0 si la distance qui les sépare dépasse un seuil σ_{share} donné :

$$sh(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{share}}\right)^{\alpha}, & \text{si } d < \sigma_{share} \\ 0, & \text{sinon} \end{cases}$$

où α est une constante qui contrôle l'allure de la fonction de partage.

La complexité du partage étant en $O(n^2)$ puisqu'il nécessite de comparer tous les individus entre eux, cette procédure est coûteuse en temps de calcul. Afin de diminuer cette complexité, Yin et Germay ont proposé dans [YG93] de rassembler les individus identiques par *niches* avant d'appliquer la procédure de partage à chacune des niches.

Les techniques de nichage ont pour objectif de simuler la préservation de *niches écologiques* afin d'éviter une convergence prématurée de l'algorithme vers un optimum local. Elle permet d'identifier plusieurs optima : à chaque niche correspond une région de l'espace de recherche explorée par les individus qui la peuplent.

Nous ne présentons ici que la méthode d'identification des niches utilisée dans [MS95] :le *Greedy Dynamic Peak Identification* suppose connu le nombre maximal de niches q. Le tableau A.1 décrit l'algorithme d'identification des niches.

Une fois les niches identifiées, la fitness de chaque individu présent est divisée par la taille de la niche (*i.e.* son nombre d'individus), à l'exception de la tête de niche qui est préservée. Pour les «sans-niches», on applique la procédure de partage décrite par Goldberg. Comme ces individus non classés sont de moins en moins nombreux au fur et à mesure de l'évolution, l'algorithme n'est pénalisé qu'au début de l'évolution, le temps que les optima soient localisés.

Identification des niches

1. Tri de la population par ordre de fitness initiale décroissante.
2. Extraction d'un individu i par ordre de tri.
3. Si i appartient à une ou plusieurs niches, il est affecté à la niche dont la tête (le meilleur individu) est la plus proche de i au sens de la distance d. Sinon i devient une nouvelle tête et le nombre de niches identifiées augmente de 1.
4. Si le nombre de niches identifiées est inférieur à q ou s'il reste des individus à classer : retour à la phase 2.
5. Fin. Tous les individus non classés sont affectés à la classe des «sans-niche».

TABLEAU A.1 : *Greedy Dynamic Peak Identification*

Annexe B

Notions de cryptographie

Nous présentons ici quelques primitives de cryptographie. Nous détaillons les mécanismes qui régissent le chiffrement, l'authentification et les signatures numériques. Cette introduction ne se prétend pas exhaustive, mais vise simplement à fournir quelques précisions utiles dans le cadre de cette thèse.

Plus de détails sont disponibles dans de nombreux ouvrages de références comme [Kah96], [Sch96] ou encore [MvOV99].

B.1 Quantité d'information et entropie

Un outil a été proposé par C. Shannon en 1949 pour étudier les algorithmes cryptographiques [Sha49b]. Il y définit, entre autre, la *quantité d'information* :

Définition B.1 *Soient une variable aléatoire X à valeurs dans un ensemble fini E, une distribution de probabilité P, et $x \in E$. La* quantité d'information *associée à la réalisation de l'événement $X = x$ est donnée par :*

$$I(X = x) = -\log_2 P(X = x)$$

La quantité d'information se mesure en bits.

Cette définition signifie qu'un événement «presque sûr» (*i.e.* avec une probabilité proche de 1) ne nous apprend pas grand chose lorsqu'il se réalise puisqu'on s'y attendait. En revanche, si sa probabilité est basse, sa réalisation étant une surprise, la quantité d'information engendrée est plus importante. Intuitivement, une information revêt d'autant plus d'importance qu'elle reflète un événement exceptionnel.

La définition posée par Shannon suit précisément cette direction :

- $I(X = x) = 0$ ssi $P(X = x) = 1$, c'est-à-dire quand un événement est certain ;
- $\lim_{P(X=x)\to 0^+} I(X = x) = +\infty$, c'est-à-dire quand un événement est exceptionnel ;

Shannon définit également l'*entropie* de X :

Définition B.2 *Soient une variable aléatoire X à valeurs dans un ensemble fini E et une distribution de probabilité P. L'*entropie *de X est donnée par :*

$$H(X) = \sum_{x \in E} P(X = x) I(X = x) = - \sum_{x \in E} P(X = x) \log_2 P(X = x)$$

L'entropie représente la quantité moyenne d'information apportée par un événement quelconque. On peut également voir l'entropie comme la mesure de l'incertitude liée à la réalisation d'un événement :

- si un événement possède une probabilité proche de 0 ou de 1, alors l'entropie de X est presque nulle. En effet, dans ce cas, l'issue du tirage aléatoire ne fait pratiquement aucun doute ;
- si tous les événements sont équiprobables, l'entropie de X est alors maximale : on ne peut rien déduire sur le résultat du tirage aléatoire à partir de notre connaissance sur X.

Ceci montre que l'entropie ne dépend pas de la réalisation d'un événement, mais uniquement de sa distribution de probabilité.

L'entropie mesure l'information *a posteriori* apportée par un événement lorsqu'il se réalise, ou de manière équivalente, la quantité d'incertitude liée à cet événement *a priori*.

Si on considère maintenant une seconde variable aléatoire Y, on peut alors définir l'*entropie conditionnelle de X par rapport à Y* comme étant :

$$H(Y|X) = - \sum_{(x,y) \in E^2} P(X = x | Y = y) \ \log_2 P(X = x | Y = y)$$

Elle mesure l'information que la connaissance de Y donne sur X, ou, de manière équivalente, l'incertitude restant autour de X lorsqu'une valeur de Y est donnée.

B.2 Confidentialité : chiffrement des données

Historiquement, le premier but de la cryptographie était de protéger un message contre les personnes ne devant pas y avoir accès. Cette clause est appelée *confidentialité*. Le *chiffrement* et le *déchiffrement* aspirent à cela.

B.2.1 Modélisation du chiffrement

Pour un schéma donné, soient $\mathbf{M}, \mathbf{C}, \mathbf{K}$ l'ensemble respectivement de tous les messages clairs, des messages chiffrés et des clés. Soient $m \in \mathbf{M}$ un message et $k \in \mathbf{K}$ une clé. Le message chiffré associé est donné par l'application de chiffrement :

$$\begin{array}{rcl} E & : & \mathbf{M} \times \mathbf{K} \to \mathbf{C} \\ & & (m, k) \mapsto c = E_k(m) \end{array} \tag{B.1}$$

Pour retrouver m à partir de c, on détermine un procédé inverse de la forme :

$$\begin{array}{rcl} D & : & \mathbf{C} \times \mathbf{K} \to \mathbf{M} \\ & & (c, k') \mapsto m = D_{k'}(c) \end{array} \tag{B.2}$$

Ces deux fonctions et les clés k, k' doivent au moins vérifier $D_{k'}(E_k(m)) = m$, afin que le déchiffrement soit possible et unique (voir fig. B.1).

Dès lors, on constate que deux schémas de chiffrement sont possibles, selon la relation entre k et k'.

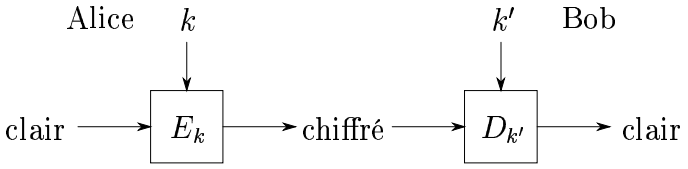


FIGURE B.1 : Schéma de chiffrement

B.2.2 Chiffrement à clé secrète

Lorsqu'il est facile de calculer k' à partir de k, on parle alors de *chiffrement symétrique* ou encore de *chiffrement à clé secrète*. Les deux entités impliquées dans la communication doivent alors partager un secret, k par exemple, afin que chacun puisse chiffrer et déchiffrer.

Il est essentiel que personne d'autre ne sache quoique ce soit sur k, sans quoi il serait capable de retrouver k' et ainsi de déchiffrer la communication.

D'une manière plus générale, on considère qu'il y a seulement une clé k dans de tels algorithmes. Ceux-ci existent depuis des siècles et sont encore très présents actuellement (DES, AES, SAFER, IDEA, ...). Ces schémas sont très rapides mais nécessitent une étape préalable afin d'échanger une clé k pour chaque couple d'utilisateurs. Ainsi, dans un groupe de n personnes, chacune d'elle doit posséder $n-1$ clés différentes afin de chiffrer ses communications avec chacun des $n-1$ autres membres du groupe : il faut une clé secrète par paire d'intervenants. Ces n personnes manipulent un total de $(n-1)+(n-2)+\ldots+2+1 = n(n-1)/2$ clés distinctes.

B.2.3 Chiffrement à clé publique

L'autre type schéma de chiffrement est dit *asymétrique* ou *à clé publique*. Suite aux nouvelles directions présentées par W. Diffie et M. Hellman en 1976 dans [DH76], les recherches se sont intensifiées pour parvenir à un système cryptographique qui utiliserait deux clés, l'une servant au chiffrement, l'autre au déchiffrement. Le premier schéma fut proposé en 1977 par R. L. Rivest, A. Shamir et M. Adelman [RSA77, RSA78] : le système RSA.

Les schémas asymétriques reposent sur l'impossiblité[1] de retrouver k' à partir de k et l'unicité de la relation liant k et k'. La structure générale utilisée pour construire ces schémas s'appuie sur des problèmes mathématiques extrêmement difficiles à résoudre lorsqu'on ne connaît pas k', mais triviaux dans le cas contraire.

[1] impossibilité au moins calculatoire à défaut de théorique.

Supposons qu'Alice souhaite envoyer un message chiffré à Bob. Ici, la clé k est publiquement connue alors que k' est conservée cachée par Bob. Alice se sert de k pour chiffrer le message qu'elle destine à Bob. Ce dernier retrouve le message clair à l'aide de sa clé privée[2] k'.

Les schémas les plus connus sont RSA et ElGamal. Ils sont moins rapides que les schémas symétriques, mais ne requièrent pas un échange préalable de clé. Par exemple, lorsque n personnes souhaitent communiquer, l'ensemble ne nécessite que n paires de clés.

Dans la pratique, les schémas asymétriques servent principalement à échanger une clé secrète qui est ensuite utilisée pour chiffrer symétriquement la transaction (on parle alors de *clé de session*). Alice choisit une clé secrète k et la chiffre avec la clé publique de Bob avant de la lui envoyer. Bob est le seul à pouvoir déchiffrer le message et obtient donc la même clé k. Le reste de la communication est ensuite chiffré à l'aide de k selon un schéma symétrique.

B.2.4 Chiffrement et théorie de l'information

Les notions introduites par C. Shannon sont utilisées dans un cadre cryptographique pour définir précisément le comportement d'un schéma de chiffrement.

Dans les espaces $\mathbf{M}, \mathbf{C}, \mathbf{K}$, on associe à chaque élément sa probabilité d'être sélectionné. Par exemple, chaque clé de $\mathbf{K}$ possède la probabilité *a priori* d'être choisie parmi toutes les clés possibles (de même pour $\mathbf{M}$). On note M, C, K les variables aléatoires correspondantes, et m, c, k leurs réalisations. Les distributions de probabilité sur $\mathbf{M}, \mathbf{K}$ induisent celles sur l'espace des chiffrés $\mathbf{C}$ puisque M et K sont indépendantes :

$$P(C = c) = \sum_{(m,k)/c=E_k(m)} P(M = m)P(K = k)$$

L'entropie de Shannon permet d'évaluer la qualité du système. Par exemple, comme pour un chiffré et une clé donnés correspondent un unique clair, on a $H(m|k, c) = 0$. Pour mesurer la connaissance dont dispose un attaquant *a priori*, on utilise les distributions de probabilités des variables aléatoires M, C, K. Les entropies $H(c|k)$ liant le chiffré à la clé et $H(c|m)$ le chiffré au message sont révélatrices des performances du crypto-système : idéalement, le chiffré ne doit rien révéler, ni sur le clair, ni sur la clé.

Ce cas idéal se produit lorsque la connaissance du chiffré c n'apporte pas plus de connaissance sur m que m n'en donne sur lui-même, c'est-à-dire quand $H(m|c) = H(m)$. Un schéma vérifiant cette propriété est dit *parfait*.

Shannon a démontré en 1949 [Sha49b] que, dans un tel système, le nombre de clés devait être au moins aussi grand que le nombre de messages, c'est-à-dire $|\mathbf{K}| \geq |\mathbf{M}|$.

[2] Pour éviter la confusion avec la terminologie du chiffrement symétrique et sa clé secrète, le chiffrement asymétrique utilise les termes clé publique / clé privée.

B.2.5 Les attaques

Le but de la cryptanalyse est de parvenir à contrer les solutions proposées par la cryptographie. Selon les conditions dans lesquelles se déroule l'attaque, sa portée varie. Les différents types d'attaques sont :

- *attaque à chiffrés seuls* : l'attaquant ne dispose que d'un ou plusieurs messages chiffrés qu'il est parvenu à intercepter sur le canal et il cherche à retrouver le clair correspondant ;
- *attaque à clairs connus* : l'attaquant dispose de paires (clairs, chiffrés) qu'il utilise pour déchiffrer un autre chiffré ;
- *attaque à clairs choisis* : par rapport à la précédente, l'attaquant peut maintenant choisir quelques clairs dans les paires (clairs, chiffrés) dont il dispose (*i.e.* il obtient, pour un bref instant, la machine de chiffrement). ;
- *attaque à chiffrés choisis* : l'attaquant sélectionne les chiffrés qui l'intéresse et accède aux clairs correspondants (*i.e.* il obtient, pour un bref instant, la machine de déchiffrement).

Il existe également des attaques dites *adaptatives* dans lesquelles le choix des clairs ou des chiffrés dépend des résultats obtenus préalablement.

B.3 Fonctions de hachage et intégrité des données

B.3.1 Introduction aux fonctions de hachage

Une fonction de hachage prend un message binaire de longueur arbitraire t en entrée, pour construire une chaîne binaire de longueur n fixée en sortie ($t > n$). Cette dernière est appelée *valeur hachée* ou plus simplement *haché*. Ces fonctions permettent ainsi de représenter des données sur une taille fixée et réduite.

En plus d'offrir une représentation réduite de données, elles doivent également satisfaire le *principe d'avalanche*, qui stipule que si l'entrée change d'un bit, alors la moitié des bits de sortie change (en moyenne).

Soient $X = \{0,1\}^t$, $Y = \{0,1\}^n$ ($t > n$) et h une fonction de hachage $h : X \rightarrow Y$. La fonction h étant surjective, il existe des *collisions*, c'est-à-dire des éléments x_0 et x_1 de X tels que $h(x_0) = h(x_1)$.

Si toutes les images $y \in Y$ sont à peu près équiprobables, alors environ 2^{t-n} éléments de X produisent une même image. Ainsi, deux éléments choisis aléatoirement dans X ont une probabilité de $\frac{2^{t-n}}{2^t} = 2^{-n}$ d'avoir une image identique.

De telles fonctions servent en cryptographie, principalement pour détecter la modification de données. On parle alors d'*intégrité des données* (*MDC : modification detection code*), même si une fonction de hachage ne suffit pas à elle seule pour cela. Elles sont également utilisées pour l'authentification de messages (*MAC : message authentication code*), auquel cas un paramètre supplémentaire est requis : une *clé*.

Le principal intérêt des fonctions de hachage en cryptographie est de fournir une représentation compacte d'une chaîne donnée. Cette représentation est considérée comme unique dès lors que n est assez grand et que des collisions sont difficiles à calculer.

Les MDCs traitent de l'intégrité des données transmises, lorsqu'utilisées de manière appropriée. Quant aux MACs, elles fournissent un moyen de s'assurer de la provenance **et** de l'intégrité d'un message (on parle alors d'*authentification de message* ou plus simplement *authentification*). Les schémas de signature numérique permettent également de faire de l'authentification (cf. section B.5)

B.3.2 Propriétés requises

Les fonctions de hachage sont divisées en deux catégories, selon qu'elles nécessitent ou non une clé. D'une manière générale, elles requièrent au minimum les propriétés suivantes :

1. *compression* : h fait correspondre à un élément x de taille finie et arbitraire une image $h(x)$ de longueur n donnée plus petite ;
2. *simplicité de calcul* : pour h et x donnés, $h(x)$ est calculatoirement peu coûteux.

Dans le cas de MDCs, on attend de ces fonctions qu'elles vérifient au moins une des propriétés suivantes :

1. h est *faiblement sans collision* si, étant donné $x \in X$, il est calculatoirement difficile de trouver $x' \neq x$, $x' \in X$, tel que $h(x') = h(x)$;
2. h est *fortement sans collision* (ou plus simplement *sans collision*) s'il est calculatoirement difficile de trouver $x, x' \in X$ tels que $h(x) = h(x')$ (ici, x et x' sont librement choisis) ;
3. h est à *sens unique* si, pour presque tous les $y \in Y$, il est calculatoirement difficile de trouver $x \in X$ tel que $h(x) = y$.

Ces propriétés sont importantes dans les schémas de signature comme nous le verrons de manière plus approfondie dans la section B.5 page 200.

Pour les MACs, en plus de la simplicité calculatoire et de la compression, pour toutes les clés possibles k, inconnues de l'adversaire, la propriété suivante est requise :

- *résistance calculatoire* : il est calculatoirement impossible à un attaquant disposant de paires $(x_i, h_k(x_i))$ de trouver une paire $(x, h_k(x))$ pour toute entrée $x \neq x_i$ (ce qui comprend le cas où $h_k(x) = h_k(x_i)$ pour certains i).

Cette résistance implique qu'il est difficile de retrouver la clé k employée à partir d'un ensemble de paires (message, haché) : pour une clé k fixée, il est calculatoirement infaisable de retrouver k étant donné un nombre quelconque de paires $(x, h_k(x))$. On appelle cette propriété *non découverte de clé*. En revanche, l'inverse n'est pas vérifié : il n'est pas toujours nécessaire de connaître la clé k pour découvrir la paire $(x, h_k(x))$.

La sécurité des MACs réside donc dans la non divulgation de la clé puisqu'aucune propriété n'est requise dès lors que celle-ci est dévoilée.

B.3.3 Attaques

Il existe différentes attaques contre les fonctions de hachage. Certaines visent la partie de compression, d'autres leur aspect itératif. Il existe également une attaque, indépendante de l'algorithme considéré, fondée sur le *paradoxe des anniversaires*.

Paradoxe des anniversaires

La sécurité des fonctions de hachage repose donc sur la difficulté à déterminer des collisions. Le *paradoxe des anniversaires* permet d'estimer cette difficulté.

Théorème B.1 *Soit $X = \{0,1\}^t$ et $h : X \to Y$ une fonction de hachage avec $|Y| = n$. Pour trouver une collision avec une probabilité supérieure à $\frac{1}{2}$, il suffit de hacher $O(\sqrt{n})$ éléments de X.*

Pourquoi «paradoxe des anniversaires» ? Historiquement, il s'agissait de déterminer la taille d'un groupe de personnes tel que deux d'entre elles soient nées le même jour, avec une probabilité supérieure à $\frac{1}{2}$.

Notons X un groupe de personnes, Y l'ensemble des 365 jours d'une année non bissextile, et $h(x)$ la date de l'anniversaire d'une personne de X. Alors il suffit de 23 personnes dans X pour que la probabilité que deux d'entre elles aient la même date d'anniversaire soit supérieure à $\frac{1}{2}$.

Attaques des anniversaires de Yuval

G. Yuval a présenté une attaque dans [Yuv79] qui repose sur le paradoxe des anniversaires pour les schémas de signature avec annexe (cf. B.5 page 200). Si la fonction de hachage h produit des hachés de taille[3] m, alors après avoir calculé 2^m, la probabilité de collision est supérieure à 1/2.

Soient un message x_1 légitime et un autre x_2, que l'attaquant souhaite substituer à x_1. L'attaque a pour objectif de construire deux paires $(x'_1, h(x'_1))$ et $(x'_2, h(x'_2))$ telles que $h(x'_1) = h(x'_2)$, où x'_1 et x'_2 sont aussi proches que possible des messages initiaux. En effet, la résistance faible de h rend très difficile la découverte d'un autre antécédent, l'attaque cherche donc à construire une collision. Elle est détaillée dans le tableau B.1.

Classification des différents types de forges

Une *forge* est une opération par laquelle un attaquant parvient à calculer un antécédent à un haché. On en distingue trois niveaux :

1. *existence de forge* : un attaquant parvient à construire une paire (message, haché), mais sans contrôle sur le message ;

[3] La taille de l'ensemble des hachés est donc 2^m puisque ce sont des chaînes binaires de longueur m.

Attaque des anniversaires

1. x_1 un message légal et x_2 le message que Charlie aimerait bien substituer à x_1
2. Charlie génère $2^{m/2}$ modifications mineures x'_1 de x_1
3. Charlie calcule les hachés $h(x'_1)$ et sauvegarde les paires $(x'_1, h(x'_1))$
4. Charlie modifie légèrement x_2 en x'_2, calcule $h(x'_2)$ et recherche parmi les hachés $h(x'_1)$ s'il n'y est pas déjà. Il recommence jusqu'à trouver une collision.
5. Charlie possède deux paires $(x'_1, h(x'_1))$ et $(x'_2, h(x'_2))$ telles que $h(x'_1) = h(x'_2)$

TABLEAU B.1 : Attaques des anniversaires de Yuval

2. *forge sélective* : un attaquant parvient à produire une paire (message, haché) en contrôlant le message, au moins partiellement ;
3. *cassage complet* : dans le cas des MACs, ceci signifie que l'attaquant parvient à découvrir la clé à l'aide de paires $(x_i, h_k(x_i))$.

B.3.4 Intégrité et authentification des données

L'*intégrité des données* revient à vérifier que des données n'ont pas été modifiées depuis leur création. L'*authentification de source* (aussi appelée *authentification de message*) garantit qu'une entité est bien la source des données créées à un moment quelconque dans le passé.

Les fonctions de hachage constituent une première solution pour obtenir ces résultats :

- utilisation de fonctions MAC (fig. B.2) : le message est passé à une fonction MAC, paramétrée par une clé secrète. Le haché est attaché au message et ils transitent alors ensemble dans le canal. Le message doit contenir un champ permettant d'identifier la source. Si, à la réception du message, la valeur MAC calculée par le destinataire correspond à celle envoyée, alors le message n'a pas été altéré ;
- utilisation de fonctions MDC et de chiffrement (fig. B.3) : on calcule le haché du message par une fonction MDC. Ce haché est attaché au message puis un schéma de chiffrement, paramétrée par une clé, est appliqué à l'ensemble, qui transite alors chiffré par le canal. En effet, l'utilisation d'une clé n'est pas requise pour obtenir l'intégrité des données, mais uniquement pour authentifier le haché ;
- utilisation de fonctions MDC et d'un canal sécurisé (fig. B.4) : on calcule le haché du message par une fonction MDC. Ce haché transite par un canal

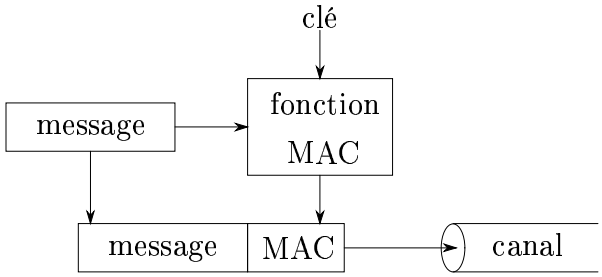


FIGURE B.2 : Intégrité *via* une fonction MAC

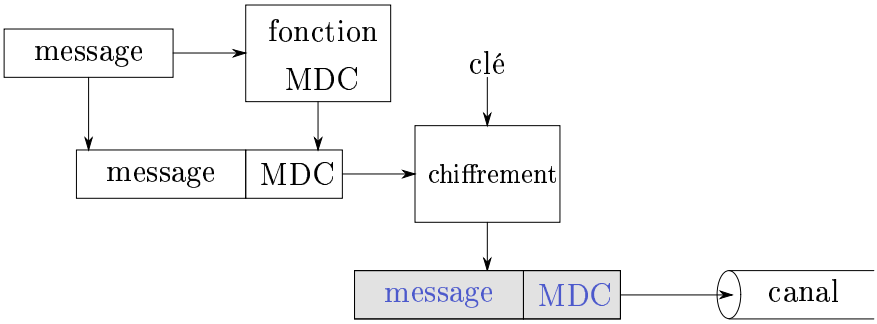


FIGURE B.3 : Intégrité *via* une fonction MDC et du chiffrement

sécurisé, pendant que le message passe par le canal normal. Le canal sécurisé peut prendre différentes formes : le téléphone (reconnaissance vocale de l'interlocuteur), un support physique (cdrom, disquette ou autres), un VPN (*Virtual Private Network*), ...

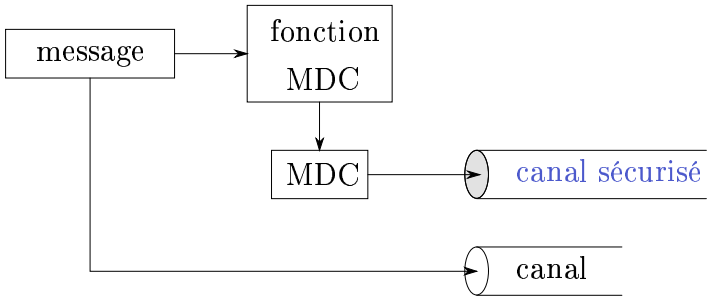


FIGURE B.4 : Intégrité *via* une fonction MDC et un canal sécurisé

B.4 Identification et authentification d'entité

L'*identification* (ou *authentification d'identité*) permet à une entité (le *prouveur*) de convaincre indéniablement de son identité une autre personne (le *vérifieur*). Cela protège des attaques fondées sur l'*impersonnification*. Généralement,

le vérifieur doit s'assurer de la validité d'un message provenant du prouveur, validité qui dépend en fait d'un secret connu uniquement du prouveur.

La différence majeure avec l'authentification de message (par MAC ou signatures numériques) vient de ce que cette dernière ne fournit aucune indication temporelle sur la date de création du message. En revanche, dans le cas de l'authentification d'entité, celle-ci est réalisée pendant la communication courante.

Il existe deux types d'authentification :

1. *authentification faible* : elle repose sur l'utilisation de mots de passe. Si l'entité le connaît, alors elle est identifiée ;
2. *authentification forte* : le vérifieur soumet le prouveur à plusieurs défis fondés sur des échanges questions/réponses.

Nous ne nous intéressons ici qu'à la version forte. Il existe différents types de challenges :

- *défi symétrique* : il repose sur des algorithmes de chiffrement symétrique ou encore des fonctions à sens unique avec clé, ce qui nécessite que le prouveur et le vérifieur partagent au préalable un même secret ;
- *défi asymétrique* : avec un algorithme de chiffrement à clé publique, le prouveur démontre au vérifieur qu'il possède bien la clé privée correspondante en deux étapes :
  1. le prouveur déchiffre un message chiffré par sa clé publique ;
  2. le prouveur signe ce message.
- *preuve à divulgation nulle de connaissance* : le prouveur parvient à convaincre le vérifieur qu'il connaît un secret sans révéler aucune information sur ce secret.

Dans la suite, nous ne présentons que ce dernier type de protocoles.

B.4.1 Preuve à divulgation nulle de connaissance

Dans la suite, nous noterons V le vérifieur, P le prouveur.

Les preuves à divulgation nulle de connaissance (*zero-knowledge proofs*) imposent une contrainte supplémentaire aux protocoles d'identification : à l'issue du défi, le vérifieur V ne doit disposer d'**aucune** information sur le secret détenu par le prouveur P. Le premier protocole de cette sorte a été introduit dans [GMW86] et [MR89].

De tels schémas se révèlent très confortables pour l'identification puisque le secret est choisi de manière à ce que seul le prouveur le connaisse. De cette façon, lorsque P persuade le vérifieur qu'il connaît le secret, il l'assure également de son identité.

Ces schémas reposent sur des problèmes qui sont difficiles à résoudre sans la connaissance d'un secret, comme par exemple des problèmes d'arithmétique (les protocoles de Guillou-Quisquater et Fiat-Shamir, voir [Sch96]). Nous en présentons ici deux autres fondés sur l'emploi de graphes.

B.4.1.1 Isomorphisme de graphes

Les auteurs de [GMW86] furent les premiers à proposer un tel schéma fondé sur la difficulté à retrouver un isomorphisme entre deux graphes. Deux graphes sont publiquement connus : G_0 et $G_1 = \sigma(G_0)$, où σ est un isomorphisme secret. Le prouveur doit convaincre le vérifieur qu'il connaît cet isomorphisme σ.

Le protocole repose sur l'échange résumé dans le tableau B.2. Il est très difficile, à partir de deux graphes isomorphes, de retrouver l'isomorphisme qui les lie.

Le prouveur génère aléatoirement une nouvelle permutation π et calcule $H = \pi(G_0)$. Il donne ce graphe au vérifieur. Celui-ci choisit au hasard la valeur d'un bit b et retourne ce bit au prouveur. Selon la valeur de ce bit, le prouveur fournit au vérifieur l'isomorphisme entre le graphe G_b et H :

- si $b = 0$, la permutation est π ;
- si $b = 1$, il s'agit de $\pi o \sigma^{-1}$ puisque $\pi o \sigma^{-1}(G_1) = \pi(G_0) = H$.

Cette transaction est répétée autant de fois que nécessaire. Si P est malhonnête, la probabilité qu'il donne la bonne réponse à chaque répétition est égale[4] à $1/2$. Ainsi, après n itérations, la probabilité que P donne n réponses valides est de $1/2^n$. Donc, pour n suffisamment grand, si P a donné une réponse exacte à chaque fois, le vérifieur V est convaincu que P connaît σ avec une probabilité élevée, de $1 - 1/2^n$.

Prouveur		**Vérifieur**
Choisit un isomorphisme π puis calcule $H = \pi(G_0)$	$\xrightarrow{H}$	
	$\xleftarrow{b}$	Choisit aléatoirement $b \in \{0, 1\}$
Si $b = 0$ renvoie π, sinon $\pi o \sigma^{-1}$	Isomorphisme φ_b entre G_b et H $\xrightarrow{\quad}$	Vérifie que $\varphi_b(G_b) = H$

TABLEAU B.2 : *zero-knowledge* et isomorphisme de graphes

B.4.1.2 Calcul d'un circuit Hamiltonien

Ce schéma a été proposé par M. Blum dans [Blu86]. Un graphe G est publiquement connu. Il contient un circuit Hamiltonien, gardé secret. Le prouveur cherche à convaincre le vérifieur qu'il connaît ce circuit.

Le défi est résumé dans le tableau B.3. Tout comme le précédent, il doit être répété plusieurs fois. Le prouveur génère aléatoirement la permutation π et le graphe H tel que $H = \pi(G)$. Il transmet H au vérifieur, qui lui renvoie un bit

[4]Puisque G_0 et G_1 sont publics, un attaquant peut choisir de calculer H par rapport à l'un des graphes, mais il ne pourra répondre qu'une fois sur deux au défi, lorsque celui-ci porte sur le graphe qu'il a sélectionné.

dont la valeur est déterminée par tirage au sort. Si celui-ci vaut 0, le prouveur donne au vérifieur l'isomorphisme π entre G et H. S'il vaut 1, il révèle le circuit Hamiltonien contenu dans H, obtenu en appliquant la permutation au circuit contenu dans G.

Prouveur		**Vérifieur**
Choisit un isomorphisme π puis calcule $H = \pi(G)$	$\xrightarrow{H}$	
	$\xleftarrow{b}$	Choisit aléatoirement $b \in \{0, 1\}$
si $b = 0$:	$\xrightarrow{\text{Isomorphisme entre } G \text{ et } H}$	
si $b = 1$:	$\xrightarrow{\text{Circuit Hamiltonien de } H}$	

TABLEAU B.3 : *zero-knowledge* et circuit Hamiltonien

B.4.2 Attaques

Il existe de nombreuses attaques contre les schémas d'identification. Nous ne citerons ici que les trois plus importantes :

- *impersonnification* : l'attaquant parvient à faire croire au vérifieur qu'il est le prouveur ;
- *attaque par rejeu* : l'attaquant rejoue un défi qu'il a intercepté auprès d'un vérifieur (le même ou un autre qui celui qui a lancé le défi), ce qui conduit aussi à une impersonnification ;
- *attaque à clairs choisis* : l'attaquant choisit méticuleusement les défis qu'il lance au prouveur afin d'extraire de l'information sur son secret. Il peut ensuite essayer de se faire passer pour lui.

B.5 Signature numérique

Les signatures numériques et manuscrites poursuivent le même but : lier une identité à un document. Cependant, comme la reproduction de données numériques est plus facile et parfaite que celle de données «analogiques», des protections supplémentaires sont requises.

B.5.1 Introduction et terminologie

Les schémas de signature se décomposent en deux étapes : la signature en elle-même et sa vérification. Définissons quelques notations avant de détailler ces fonctions :

- soit $\mathcal{M}$ l'ensemble des messages ;

- soit $\mathcal{M}_{\mathcal{S}}$ l'ensemble des messages qui peuvent être signés par le schéma ;
- soit $\mathcal{S}$ l'ensemble des *signatures*, en général une chaîne binaire d'une longueur donnée ;
- soit S_B une fonction de $\mathcal{M}_{\mathcal{S}}$ dans $\mathcal{S}$, appelée *fonction de signature* pour l'entité B (Bob). Cette fonction est gardée secrète par Bob ;
- soit V_B une transformation de $\mathcal{S} \times \mathcal{M}_{\mathcal{S}}$ dans $\{vrai, faux\}$, appelée *fonction de vérification* pour B. Elle est publique et permet à tout le monde de vérifier les signatures émises par B.

Bob		**Alice**
m	$\longrightarrow$	$V_B(S_B(m)) = m$?
	$(m, S_B(m))$	

TABLEAU B.4 : Schéma de signature

Les fonctions S_B et V_B fournissent un *schéma de signature pour B*. Le fonctionnement général est décrit dans le tableau B.4. Bob rend public sa fonction de vérification V_B afin que tout le monde puisse contrôler la validité des signatures qu'il émet. En revanche, il ne divulgue pas sa clé privée S_B qui lui permet de signer les messages.

La propriété recherchée essentiellement pour les schémas de signature est la *non répudiation*. Elle stipule qu'une fois le document signé, son signataire ne peut plus nier en être l'auteur. Pour que ce mécanisme soit fiable, il est alors nécessaire que la signature d'un document altéré ne soit plus valide. Par extension, la signature d'un document ne doit pas pouvoir être utilisée avec un autre document(*non réutilisable*). En effet, une signature manuscrite est facilement reproductible. On souhaite ainsi éviter de voir une personne «copier» la signature d'une autre sur un document différent. Pour la sécurité du schéma, il est également indispensable que personne d'autre que le signataire ne puisse reproduire la signature. Dans le cas contraire, deux personnes pourraient alors revendiquer en être l'auteur. Enfin, une fonctionnalité supplémentaire veut que tout le monde puisse vérifier la signature d'un document.

Certains schémas de chiffrement asymétrique fournissent un moyen efficace de construire un schéma de signature :

- la fonction de signature S_B est construite à partir de la fonction de déchiffrement D_B ;
- la fonction de vérification V_B est construite à partir de la fonction de chiffrement E_B ;

B.5.2 Schémas de signature

Les schémas de signature se divisent en deux catégories :

1. *avec annexe* : la fonction de vérification nécessite le message initial ;

2. *avec recouvrement* : le message est reconstruit à partir de sa signature.

B.5.2.1 Signature avec annexe

Ce schéma permet de signer des messages de longueur donnée. Pour éviter cette contrainte, une fonction de hachage est préalablement appliquée au message afin de le réduire à la taille désirée. La signature est ensuite calculée sur le haché du message. Pour cela, la fonction de signature S (privée) est utilisée sur le haché, considéré alors comme un chiffré, ce qui produit la signature. Cela suppose que le haché produit correspond bien à un élément de l'ensemble des chiffrés pour la fonction de déchiffrement du schéma asymétrique (voir fig. B.5).

La signature est vérifiée à l'aide de la fonction de chiffrement (publique). La signature est assimilée à un message en clair pour servir d'argument à la fonction de chiffrement du schéma asymétrique. Le haché du message est de nouveau calculé, et ces deux éléments comparés : en cas de différence, la signature n'est pas valide (voir tab. B.5 page suivante).

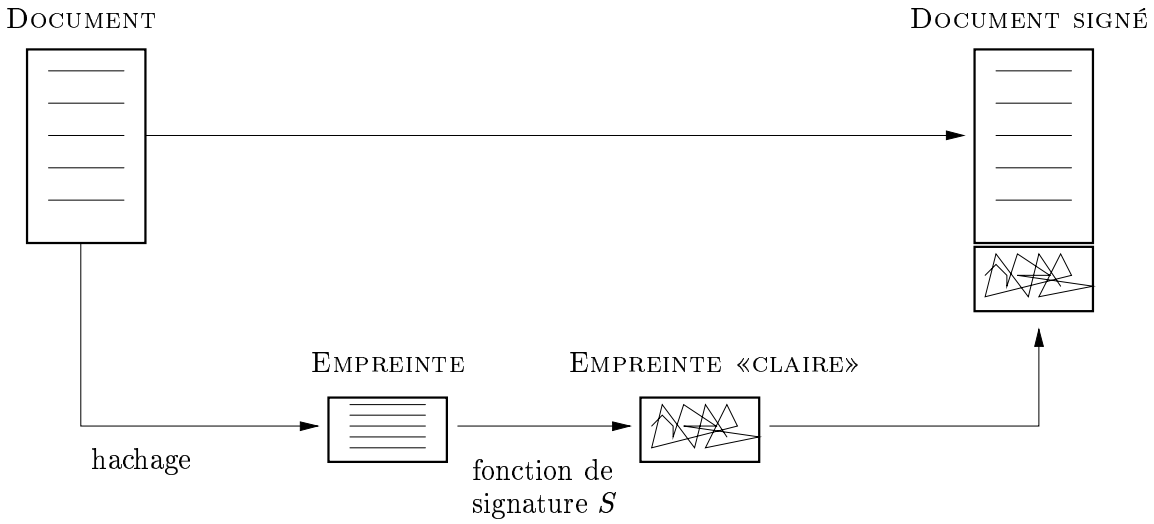


FIGURE B.5 : Signature avec annexe : hacher et chiffrer pour signer

Pour éviter qu'un attaquant remplace m par un autre message, la fonction de hachage h doit donc être sans collision.

B.5.2.2 Signature avec recouvrement

Avec ces schémas, la signature contient le message lui-même qui est alors extrait après la vérification. Ainsi, il n'est pas nécessaire de connaître le message pour en contrôler la signature.

Contrairement aux signatures avec annexe, la fonction intermédiaire n'est plus une fonction de hachage, mais une *fonction de redondance* r bijective de $\mathcal{M}_S$

Génération de la signature

1. Bob calcule le haché du message m : $h(m) = \tilde{m}$
2. Bob calcule la signature du haché : $S_B(\tilde{m}) = s_{\tilde{m}}$
3. Bob rend public la paire $(m, s_{\tilde{m}})$

Vérification de la signature

1. Alice récupère la paire $(m, s_{\tilde{m}})$ et connaît la fonction publique de vérification de Bob V_B
2. Alice calcule le haché du message m : $h(m) = \tilde{m}$
3. Alice accepte la signature ssi $V_B(\tilde{m}, s_{\tilde{m}}) = vrai$

TABLEAU B.5 : Schéma de signature avec annexe

dans $\mathcal{S}$. Cette fonction, publiquement connue, autorise le vérifieur à récupérer le message lorsque la signature est valide.

Génération de la signature

1. Bob calcule $\tilde{m}$: $r(m) = \tilde{m}$
2. Bob calcule la signature de $\tilde{m}$: $S_B(\tilde{m}) = s_{\tilde{m}}$
3. Bob rend public la signature $s_{\tilde{m}}$

Vérification de la signature

1. Alice récupère la signature $s_{\tilde{m}}$ et connaît la fonction publique de vérification de Bob V_B
2. Alice calcule $\tilde{m} = V_B(s_{\tilde{m}})$
3. Alice vérifie que $\tilde{m} \in \mathcal{M}_{\mathcal{S}}$
4. recouvrement de m par $m = r^{-1}(\tilde{m})$

TABLEAU B.6 : Schéma de signature avec recouvrement

B.5.3 Attaques

La terminologie des attaques à l'encontre des algorithmes de signature numérique est la même que celles définies pour les fonctions de hachage (cf. B.3.3) :

1. *existence de forge* : un attaquant parvient à construire une signature pour (au moins) un message, sans avoir de réel contrôle sur celui-ci ;

2. *forge sélective* : une signature valide est obtenue sur une classe de messages maîtrisée par l'attaquant ;
3. *cassage complet* : soit l'attaquant parvient à récupérer la clé privée du signataire, soit il met en place un algorithme équivalent qui produit des signatures valides ;

Les attaques contre les schémas de signature sont de deux types :

1. *attaque à clés seuls* : l'attaquant ne dispose que de la clé publique de sa victime pour mener son action ;
2. *attaque à messages* : l'attaquant parvient à examiner la signature correspondant soit à un message connu (*attaque à messages connus*), soit à un message choisi (*attaque à messages choisis*).

L'attaque des anniversaires de Yuval présentées précédemment s'applique aux schémas de signature avec annexe. D'un point de vue pratique, cette attaque est applicable de deux manières (on note x et x' deux messages tels que $h(x) = h(x')$) :

1. si le signataire est malhonnête, il peut répudier le message x et prétendre avoir signé x' à la place ;
2. un vérifieur convainc un tiers de signer le message préparé x puis prétend ultérieurement que c'est x' qui a été signé.

B.6 Partage de secret

Le premier schéma de partage de secret a été présenté par A. Shamir dans [Sha79]. Il est présenté ci-après en section B.6.3. Les recherches en ce domaine sont encore très actives et s'orientent plus vers l'utilisation de codes correcteurs ([MS81]).

B.6.1 Problématique

Un exemple tiré de [Liu68] permet d'appréhender rapidement la problématique liée au *partage de secret* (*secret sharing*) :

> Eleven scientists are working on a secret project. They wish to lock up the document in a cabinet so that the cabinet can be opened if and only if six or more of the scientists are present.
>
> What is the smallest number of locks needed ? What is the smallest numbers of keys to the locks each scientist must carry ?

Si cette situation n'est pas très réaliste, elle correspond néanmoins à des cas concrets : une entité souhaite partager un secret entre plusieurs dépositaires de sorte que seules certaines coalitions soient autorisées à accéder à ce secret.

En cryptographie, le partage de secret sert principalement pour la gestion des clés. Par exemple, quand le propriétaire d'une clé secrète, ou privée, désire la sauvegarder à plusieurs endroits, il peut alors choisir de confier «intégralement»

sa clé à différentes entités, mais plus celle-ci est divulguée, moins elle devient sûre. Une autre solution est de la découper en morceaux et d'en confier un extrait à chaque entité. On parle alors de *partage de secret* (*secret sharing*).

B.6.2 Schémas de partage de secret

Les schémas de partage de secret reposent sur la notion de *structure d'accès*. Pour un ensemble d'utilisateurs U, cette structure est construite à partir de l'ensemble $\mathcal{P}(U)$ des parties de U : certains des éléments de $\mathcal{P}(U)$ sont autorisés à accéder au secret, d'autres pas.

Cette disposition définit un schéma général. Toutefois, dans la pratique, les *schémas à seuil* sont plus facilement réalisables. Ils constituent un cas particulier du modèle général où parmi les n utilisateurs de U, il en faut k, avec $k \leq n$, ensembles pour dévoiler le secret.

Après les structures à seuil, des solutions plus souples ont vu le jour, qui autorisent l'accès à certains éléments donnés de $\mathcal{P}(U)$, et donc à des entités précises, comme un utilisateur privilégié par exemple ([Mas93]).

Enfin, on dit d'un schéma qu'il est *parfait* lorsque la réunion de parts provenant d'une structure qui n'est pas autorisée à accéder au secret ne révèle aucune information sur le secret lui-même.

B.6.3 Exemple : le schéma de Shamir

Le protocole de partage de secret proposé par Shamir repose sur l'interpolation de polynômes et le fait qu'un polynôme $P(x) = y$ de degré $t-1$ est défini de manière unique par t points distincts $(x_i, y_i), 1 \leq i \leq t$.

Soit S un nombre secret qu'une entité (Bob, par exemple) souhaite partager entre n différents utilisateurs de sorte à ce qu'un groupe de t puisse quand même retrouver S. Bob va alors construire aléatoirement un polynôme P de degré t tel que $P(0) = S$ puis distribuer à chacun des participants un point défini par $(x_i, P(x_i)), 1 \leq i \leq n$.

Si un groupe de k utilisateurs ($k < t$) rassemble ses parts, ils ne peuvent accéder au secret car il existe une infinité de polynômes passant par k points lorsque $k < t$, ce qui montre que ce schéma est *parfait*. Au contraire, s'ils sont t ou plus, on dispose alors d'un système de t équations à t inconnues : il n'existe qu'une unique solution, qui permet de retrouver le secret $S = P(0)$.

Construction des parts

1. Bob dispose du secret S qu'il souhaite partager entre n utilisateurs avec un seuil de t
2. Bob choisit un nombre premier $p > \max(S, t)$
3. Bob pose $P(0) = a_0 = S$
4. Bob choisit aléatoirement $t-1$ coefficients $a_1, \ldots, a_{t-1}$ tels que $0 \leq a_i \leq p-1$, ce qui définit le polynôme $P(x) = \sum_{i=0}^{t-1} a_i x^i$
5. Bob distribue $S_j = P(j) \mod p, 1 \leq j \leq n$ à chacun des utilisateurs

TABLEAU B.7 : Schéma de partage de secret de Shamir

Bibliographie

[Ala00] Alattar (A. M.). – Smart images using digimarc's watermarking technology. *In : Electronic Imaging, Security and Watermarking of multimedia content II*. The Society for Imaging Science and Technology (IS&T) and the International Society for Optical Engineering (SPIE), pp. 264–273. – San Jose, CA, USA, janvier 2000. ISSN 0277-786X. ISBN 0-8194-3598-9.

[ANS98] Anderson (R.), Needham et Shamir (A.). – The steganographic file system. *In : IWIH : International Workshop on Information Hiding*. – 1998.

[AP98] Anderson (R. J.) et Petitcolas (F. A. P.). – On the limits of steganography. *IEEE Journal of Selected Areas in Communications*, vol. 16, mai 1998, pp. 474–481. – Special issue on copyright & privacy protection.

[Bar93] Barnsley (M. F.). – Fractal modelling of real world images. *In : Fractals Everywhere*, pp. 219–239. – New York, NY, USA, Academic Press, 2ème édition, 1993.

[Bas00] Bas (P.). – *Méthodes de Tatouage d'Images Fondées sur le Contenu*. – Thèse de doctorat, Institut National Polytechnique de Grenoble, octobre 2000.

[BCD98] Bas (P.), Chassery (J.) et Davoine (F.). – Using the fractal code to watermark images. *In : Proceedings of the IEEE International Conference on Image Processing (ICIP'98)*, pp. 469–473. – Chicago, IL, USA, 1998.

[BCL+99] Bloom (J.A.), Cox (I.J.), Linnartz (J.-P.M.G.), Miller (M.L.) et Traw (C.B.S.). – Copy protection for DVD. *In : Proceedings IEEE*, pp. 1267–1276. – juillet 1999.

[BD85] Barnsley (M.) et Demko (S.). – Iterated function system and the global construction of fractals. *Proceedings of the Royal Society*, vol. A 399, 1985, pp. 243–245.

[BEHL86] Barnsley (M.), Ervin (V.), Hardin (D.) et Lancaster (J.). – Solution of an inverse problem for fractals and other sets. *Proceedings of National Academic Science*, vol. 83, 1986.

[BGML96] Bender (W.), Gruhl (D.), Morimoto (N.) et Lu (A.). – Techniques for data hiding. *IBM Systems Journal*, vol. 35, 1996, pp. 313–336.

[Bla93] Blaze (M.). – A cryptographic file system for UNIX. *In : ACM Conference on Computer and Communications Security*, pp. 9–16. – 1993.

[BLM99] Brassil (J. T.), Low (S.) et Maxemchuk (N. F.). – Copyright protection for electronic distribution of text documents. *In : Proceedings of the IEEE (USA)*, pp. 1181–1196. – 1999.

[Blu86] Blum (M.). – How to prove a theorem so no one else can claim it. *In : Proceedings of the international congress of mathematicians*, pp. 1444–1451. – 1986.

[BPBC97] Bartolini (F.), Piva (A.), Barni (M.) et Capellini (V.). – DCT-based watermark recovering without resorting to the uncorrupted image. *In : IEEE International Conference on Image Processing (ICIP)*, pp. 520–523. – 1997.

[Bra98] Braudaway (G. W.). – Results of attacks on a claimed robust digital image watermark. *In : Electronic imaging, security and watermarking of multimedia contents*. The Society for Imaging Science and Technology (IS&T) and the International Society for Optical Engineering (SPIE), pp. 122–131. – San Jose, CA, USA, janvier 1998.

[BS95] Boneh (D) et Shaw (J). – Collusion-secure fingerprinting for digital data. *In : 15th Annual International Cryptology Conference*, pp. 452–465. – Santa Barbara, CA, USA, 27–31 1995.

[BS01] Bœuf (J.) et Stern (J.). – An analysis of one of the SDMI candidates. *In : Proceedings of the Information Hiding Workshop*, LNCS. – Pittsburgh, PA, USA, avril 2001.

[BSC01] Bo (Xiaochen), Shen (Lincheng) et Chang (Wensen). – Evaluation of the image degradation for a typical watermarking algorithm in the block-dct domain. *In : Proceedings of the Third International Conference on Information and Communications Security (ICICS01)*. – Xian, China, novembre 2001.

[Cac98] Cachin (C.). – An information-theoretic model for steganography. *In : Proceedings of the Workshop on Information Hiding*, pp. 306–318. – 1998.

[Can98] Canus (C.). – Robust large deviation multifractal spectrum estimation. *In : Proceedings of International Wavelets Conference (IWH98)*. – Tangier, Morroco, avril 1998.

[CEL01] Cohen (G.), Encheva (S.) et Litsyn (S.). – Intersecting codes and partially identifying codes. *In : Proceedings of the International Workshop on Coding and Cryptography*, pp. 139–147. – Paris, France, 2001.

[CEZ99] Cohen (G.), Encheva (S.) et Zemor (G.). – Copyright protection for digital data. *In : Proceedings of the Workshop on Watermarking and Copyright Enforcement.* – Paris, France, octobre 1999.

[CK01] Craver (S.) et Katzenbeisser (S.). – Security analysis of public-key watermarking schemes. *In : Mathematics of Data/Image Coding, Compression, and Encryption IV.* The International Society for Optical Engineering (SPIE), pp. 172–182. – juillet 2001.

[CKLS96] Cox (I. J.), Kilian (J.), Leighton (T.) et Shamoon (T.). – Secure spread spectrum watermarking for multimedia. *In : Proceedings of the Workshop on Information Hiding.* – University of Cambridge, UK, mai 1996. Copyright (c) 1996 by NEC Research Institute.

[CKLS97] Cox (I. J.), Kilian (J.), Leighton (T.) et Shamoon (T.). – Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing*, vol. 6, 1997, pp. 1673–1687.

[CL96] Canus (C.) et Lévy Véhel (J.). – Change detection in sequences of images by multifractal analysis. *In : Proceedings of ICASSP*, pp. 2178–2181. – Atlanta, GA, USA, mai 1996.

[Cla99] Clarke (I.). – *A distributed decentralised information storage and retrieval system.* – Rapport de recherche, University of Edinburgh, 1999.

[CLRS99a] Collet (P.), Lutton (E.), Raynal (F.) et Schoenauer (M.). – Individual GP : an alternative viewpoint for the resolution of complex problems. *In : Proceedings of the Genetic and Evolutionary Computation Conference.* pp. 974–981. – Orlando, FL, USA, juillet 1999.

[CLRS99b] Collet (P.), Lutton (E.), Raynal (F.) et Schoenauer (M.). – *Polar IFS + individual genetic programming = efficient IFS inverse problem solving.* – Rapport de recherche, INRIA, 1999.

[CLRS00] Collet (P.), Lutton (E.), Raynal (F.) et Schoenauer (M.). – Polar IFS+parisian genetic programming=efficient IFS inverse problem solving. *Genetic Programming and Evolvable Machines*, vol. 1, octobre 2000, pp. 339–361.

[CLT98] Canus (C.), Lévy Véhel (J.) et Tricot (C.). – Continuous large deviation multifractal spectrum : Definition and estimation. *In : Proceedings of Fractals 98.* – Malta, octobre 1998.

[CMYY98] Craver (S.), Memon (N.), Yeo (B.-L.) et Yeung (M. M.). – Resolving rightful ownerships with invisible watermarking techniques : Limitations, attacks, and implications. *IEEE Journal of Selected Areas in Communications*, vol. 16, 1998, pp. 573–586.

[Com95] Comes (S.). – *Les traitements perceptifs d'images numérisées.* – Thèse de doctorat, Université catholique de Louvain, Belgium, 1995.

[CP98] CATTANEO (G.) et PERSIANO (G.). – *TCFS - Transparent Cryptographic File System.* – Rapport de recherche, DIA, Universita Degli Studi Di Salerno, 1998. http://tcfs.dia.unisa.it.

[Cra00] CRAVER (S.). – Zero knowledge watermark detection. *In : Proceedings of Information Hiding (IH99)*, Lecture Notes in Computer Science. pp. 101–116. – Springer, 2000.

[CSWH00] CLARKE (I.), SANDBERG (O.), WILEY (B.) et HONG (T.W.). – Freenet : A distributed anonymous information storage and retrieval system. *In : Designing Privacy Enhancing Technologies : International Workshop on Design Issues in Anonymity and Unobservability*, Lecture Notes in Computer Science. – Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2000.

[CTL98] COLLBERG (C.), THOMBORSON (C.) et LOW (D.). – Manufactiruing cheap resilient and stealthy opaque constructs. *In : Proceedings of the ACM Symposium on the Principles of Programming Languages*, pp. 194–196. – 1998.

[CZ94] COHEN (G.) et ZEMOR (G.). – Intersecting codes and independent families. *IEEE Transactions on Information Theory*, vol. 40, 1994, pp. 1872–1881.

[Dav00] DAVOINE (F.). – Comparaison of two wavelet based image watermarking schemes. *In : IEEE International Conference on Image Processing (ICIP).* – Vancouver, Canada, septembre 2000.

[DDM98] DELAIGLE (J.-F.), DE VLEESCHOUWER (C.) et MACQ (B.M.). – A psychovisual approach for digital picture watermarking. *Journal of Electronic Imaging*, 1998.

[DeJ75] DEJONG (K. A.). – *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* – Thèse de doctorat, University of Michigan, Ann Arbor, MI, USA, 1975.

[DH76] DIFFIE (W.) et HELLMAN (M.). – New directions in cryptography. *IEEE Transactions on Information Theory*, vol. IT-22, novembre 1976, pp. 644–654.

[DS96] DAVERN (P.) et SCOTT (M.). – Fractal based image steganography. *In : Information Hiding, 1st Int. Workshop, Cambridge.* pp. 279–294. – Springer (Lecture Notes in Computer Science), 1996.

[DS01] DESIGNER (SOLAR) et SONG (DUG). – Passive analysis of ssh (secure shell) traffic, août 2001. http://www.openwall.com/advisories/OW-003-ssh-traffic-analysis.txt.

[EFS98] Encrypting file system for windows 2000, 1998. http://www.microsoft.com/windows/server/Technical/security/encrypt.asp.

[EM92] EVERTSZ (C.) et MANDELBROT (B.). – *Chaos and Fractals : New Frontiers in Science*, chap. Multifractal Measures. – H.-O. Petgen and H. Jurgens and D. Saupe, 1992, springer verlag édition.

[ESB00] Eggers (J.), Su (J.) et B.Girod. – Public key watermarking by eigenvectors of linear transforms. *In : EUSIPCO.* – Tampere, Finland, septembre 2000.

[Eur00] European Broadcasting Union and Union Européenne de Radio Télévision. – *Watermarking-call for systems*, mai 2000.

[Fal90] Falconer (K. J.). – *Fractal Geometry.* – Chichester : John Wiley & Sons, 1990.

[FD00] Furon (T.) et Duhamel (P.). – An asymmetric public detection watermarking technique. *In : Proceedings of Information Hiding (IH99)*, Lecture Notes in Computer Science. pp. 88–100. – Springer, 2000.

[Fei73] Feistel (H.). – Cryptography and computer privacy. *Scientific American*, vol. 228, mai 1973, pp. 15–23.

[Fei74] Feistel (H.). – Block cipher cryptographic system, mars 1974. US Patent 3 798 359.

[FG00] Fridrich (J.) et Goljan (M.). – Robust hash functions for digital watermarking. *In : Multimedia Security.* IEEE International Conference on Information Technology : Coding and Computing (ITCC'2000). – Las Vegas, NV, USA, mars 2000.

[FIP01] Furon (T.), I. Venturini et P. Duhamel. – Unified approach of asymmetric watermarking schemes. *In : Electronic imaging, security and watermarking of multimedia contents.* Society for Imaging Science and Technology (IS&T) and International Society for Optical Engineering (SPIE). – San Jose, CA, USA, janvier 2001. ISSN 0277-786X.

[Fis95] Fisher (Y.). – *Fractal Image Compression : Theory and Application.* – Berlin, Germany / Heidelberg, Germany / London, UK / etc., Springer Verlag, 1995.

[FOW66] Fogel (L. J.), Owens (A. J.) et Walsh (M. J.). – *Artificial Intelligence through simulated Evolution.* – John Wiley & Sons, 1966.

[FR02] Fontaine (C.) et Raynal (F.). – About the links between cryptography and data hiding. *In : A paraître dans Proceedings of the SPIE.* – San Jose, CA, USA, janvier 2002.

[Fri00] Fridrich (J.). – Visual hash for oblivious watermarking. *In : Electronic Imaging, Security and Watermarking of multimedia content II.* The Society for Imaging Science and Technology (IS&T) and the International Society for Optical Engineering (SPIE). – San Jose, CA, USA, janvier 2000. ISSN 0277-786X. ISBN 0-8194-3598-9.

[GDDM97] Goffin (F.), Delaigle (J.-F), De Vleeschouwer (C.) et Macq (B.M.). – Low-cost perseptive digital picture watermarking method. *Proceedings of the SPIE*, vol. 3022, 1997, pp. 264–277. – Storage and Retrieval for Image and Video Databases V.

[GMW86] GOLREICH (O.), MICALI (S.) et WIGDERSON (A.). – Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. *In : Proceedings of the 27th IEEE symposium on the foundations of computer science*, pp. 174–187. – 1986.

[Gol89] GOLDBERG (D. E.). – *Genetic Algorithms in Search, Optimization, and Machine Learning.* – Reading, MA, USA, Addison-Wesley Publishing Company, Inc., 1989.

[GR87] GOLDBERG (D. E.) et RICHARDSON (J.). – Genetic algorithms with sharing for multi-modal function optimisation. *In : Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications.* pp. 41–49. – Lawrence Erlbaum, 1987.

[Gut96] GUTMANN (P.). – Secure filesystem (SFS) for DOS/Windows, 1996. http://www.cs.auckland.ac.nz/~pgut001/sfs/.

[Har85] HARDIN (D.P.). – *Hyperbolic Iterated Function Systems and Applications.* – Thèse de doctorat, Georgia Institute of Technology, 1985.

[Hay82] HAYES (M. H.). – The reconstruction of a multidimensional sequence from the phase or magnitude of its fourier transform. *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-30, 1982, p. 140.

[HI96] HANDEL (T. G.) et II (M. T. SANDFORD). – Hiding data in the OSI network model. *In : Information Hiding*, pp. 23–38. – 1996.

[Hol62] HOLLAND (J. H.). – Outline for a logical theory of adaptive systems. *Journal of the Association of Computing Machinery*, vol. 3, 1962, pp. 297–314.

[Hol75] HOLLAND (J. H.). – *Adaptation in natural artificial systems.* – Ann Arbor, MI, USA, University of Michigan Press, 1975.

[Hol86] HOLLAND (J. H.). – Escaping Brittleness : The possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. *In : Machine learning, an artificial intelligence approach. Volume II*, pp. 593–623. – Morgan Kaufmann, 1986.

[HOP+98] HERRIGEL (A.), O'RUANAIDH (J.), PETERSEN (H.), PEREIRA (S.) et PUN (T.). – Secure copyright protection techniques for digital images. *In : International Workshop on Information Hiding*, pp. 169–190. – Portland, OR, USA, avril 1998.

[HS88] HARRIS (C.) et STEPHENS (M.). – A combined corner and edge detector. *In : Proceedings of the 4th Alvey Vision Conference*, pp. 147–151. – Manchester, UK, 1988.

[HSG00] HARTUNG (F.), SU (J. K.) et GIROD (B.). – Spread spectrum watermarking : Malicious attacks and counterattacks. *In : Electronic Imaging, Security and Watermarking of multimedia content II.* The

Society for Imaging Science and Technology (IS&T) and the International Society for Optical Engineering (SPIE), pp. 147–158. – San Jose, CA, USA, janvier 2000. ISSN 0277-786X. ISBN 0-8194-3598-9.

[Hut81] HUTCHINSON (J.). – Fractals and self-similarity. *Indiana University Journal of Mathematics*, 1981, pp. 713–747.

[HVR01] HERRIGEL (A.), VOLOSHYNOVSKIY (S.) et RYTSAR (Y.). – The watermark template attack. *In : SPIE Photonics West, Electronic Imaging 2001, Security and Watermarking of Multimedia Contents III.* – San Jose, CA, USA, janvier 2001.

[Int97] International Federation of the Photographic Industry, London, UK. – *Request for proposals-embeded signalling systems*, 1997.

[Jac93] JACQUIN (A. E.). – Fractal image coding : A review. *Proceedings of the IEEE*, vol. 81, octobre 1993, pp. 1451–1465.

[JJS93] JAYANT (N.), JOHNSTON (J.) et SAFRANEK (R.). – Signal compression based on models of human perception. *Proceedings of the IEEE*, vol. 81, n10, 1993, pp. 1385–1422.

[Kah96] KAHN (D.). – *The codebreakers.* – Scribner, 1996.

[ker] The international kernel patch. http://www.kerneli.org/.

[KH98] KUNDUR (D.) et HATZINAKOS (D.). – Digital watermarking using multiresolution wavelet decomposition. *In : International Conference on Acoustics, Speech and Signal Processing (ICASSP).* Institute for Electrical and Electronix Engineers (IEEE), pp. 2969–2972. – Seattle, WA, USA, mai 1998.

[KH99] KUNDUR (D.) et HATZINAKOS (D.). – Digital watermarking for telltale tamper proofing and authentication. *Proceedings of the IEEE, Special Issue on Identication and Protection of Multimedia Information*, vol. 87, juillet 1999, pp. 1167–1180.

[KK00] KINOSHITA (H.) et KOBAYASHI (T.). – An improvement on safety of a digital signature system with zkip for the graph isomorphism. *In : Proceedings of Systemics, Cybernetics and Informatics (SCI2000)*, pp. 288–294. – juillet 2000.

[Koz92] KOZA (J. R.). – *Genetic Programming : On the programming of computers by means of natural selection.* – Cambridge, MA, USA, MIT Press, 1992.

[KP99a] KATZENBEISSER (S.) et PETITCOLAS (F.) (édité par). – *Information hiding : techniques for steganography and digital watermarking.* – Artech House Books, 1999.

[KP99b] KUTTER (M.) et PETITCOLAS (F. A. P.). – A fair benchmark for image watermarking systems. *In : Electronic imaging, security and watermarking of multimedia content II*, pp. 226–239. – San Jose, CA, USA, janvier 1999. ISSN 0277-786X. ISBN 0-8194-3128-1.

[KVH00] KUTTER (M.), VOLOSHYNOVSKIY (S.) et HERRIGEL (A.). – Watermark copy attack. *In : Electronic Imaging, Security and Watermarking of multimedia content II.* Society for Imaging Science and Technology (IS&T) and International Society for Optical Engineering (SPIE), pp. 371–380. – San Jose, CA, USA, janvier 2000. ISSN 0277-786X. ISBN 0-8194-3598-9.

[KW97] KIRBY (D.) et WATANABE (K.). – Subjective testing of MPEG-2 NBC multichannel audio coding. *In : Proceedings of the International Broadcasting Convention (IBC'97)*, pp. 482–487. – Amsterdam, Netherlands, septembre 1997.

[LC98a] LIN (C.Y.) et CHANG (S.F.). – Issues and solutions for authenticating MPEG video. *In : SPIE Storage and Retrieval for Image and Video Database.* – San Jose, CA, USA, janvier 1998.

[LC98b] LIN (C.Y.) et CHANG (S.F.). – A robust image authentication method surviving JPEG lossy compression. *In : SPIE Storage and Retrieval for Image and Video Database.* – San Jose, CA, USA, janvier 1998.

[LC00] LIN (C.) et CHANG (S.). – Semi-fragile watermarking for authenticating JPEG visual content. *In : Electronic Imaging, Security and Watermarking of multimedia content II.* The Society for Imaging Science and Technology (IS&T) and the International Society for Optical Engineering (SPIE), pp. 140–151. – San Jose, CA, USA, janvier 2000. ISSN 0277-786X. ISBN 0-8194-3598-9.

[LDL94] LÉVY VÉHEL (J.), DAOUDI (K.) et LUTTON (E.). – Fractal modeling of speech signals. *Fractals*, vol. 2(3), 1994, pp. 379–382.

[Lev01] LEVY-DIT-VEHEL (F.). – Cours de cryptographie, 2001.

[LG97] LÉVY VÉHEL (J.) et GUIHENEUF (B.). – Multifractal image denoising. *In : Scandinavian Conference on Image Analysis (SCIA'97).* – Finland, 1997.

[Liu68] LIU (C. L.). – *Introduction to Combinatorial Mathematics.* – Mc Graw-Hill, 1968.

[LL93] LUTTON (E.) et LÉVY VÉHEL (J.). – *Optimization of fractal functions using genetic algorithms.* – Rapport de recherche, INRIA, juin 1993.

[LL01] LÉVY VÉHEL (J.) et LUTTON (E.). – Evolutionary signal enhancement based on hölder regularity analysis. *In : Proceedings of EVOIASP*, LCNS. – Lake Como, Italy, 2001.

[LLC+95] LUTTON (E.), LÉVY VÉHEL (J.), CRETIN (G.), GLEVAREC (P.) et ROLL (C.). – Mixed IFS : Resolution of the inverse problem using genetic programming. *Complex System*, vol. 9, octobre 1995, pp. 375–398.

[LM00] Lévy Véhel (J.) et Manoury (A.). – Wavelet packet based digital watermarking. *In : Proceedings of ICPR.* – Barcelona, Spain, septembre 2000.

[LvD98] Linnartz (J.-P. M. G.) et van Dijk (M.). – Analysis of the sensitivity attack against electronic watermarks in images. *In : Proceedings of Information Hiding Workshop*, LNCS. pp. 258–272. – Portland, OR, USA, 1998.

[Mas93] Massey (James L.). – Minimal codewords and secret sharing. *In : Proceedings of the Sixth joint swedish-russian workshop on Information Theory*, pp. 246–249. – Mölle, Sweden, 1993.

[MB99] Mintzer (F.) et Braudaway (G. W.). – If one watermark is good, are more better ? *In : International Conference on Acoustics, Speech and Signal Processing (ICASSP).* Institute for Electrical and Electronix Engineers (IEEE), pp. 2067–2070. – Phoenix, AR, USA, mai 1999.

[MEC98] Mayache (A. M.), Eude (T.) et Cherifi (H.). – A comparison of image quality models and metrics based on human visual sensitivity. *In : IEEE International Conference on Image Processing*, pp. 409–413. – octobre 1998.

[Mey92] Meyer (Y.). – *Les Ondelettes : algorithmes et applications.* – Armand Colin, 1992.

[Mit99] Mittelholzer (T.). – An information-theoretic approach to steganography and watermarking. *In : Proceedings of the Information Hiding Workshop*, LNCS 1768. pp. 1–16. – Springer Verlag, 1999.

[MK99] McDonald (A. D.) et Kuhn (M. G.). – Stegfs : A steganographic file system for linux. *In : International Workshop on Information Hiding*, pp. 462–477. – 1999.

[MLL99] Manoury (A.), Lévy Véhel (J.) et Lucas (M.-F.). – Watermarking d'images par paquets d'ondelettes. *In : Proceedings of GRETSI.* – Vannes, France, septembre 1999.

[MR89] Micali (S.) et Rackoff (C.). – The knowledge complexity on interactive proof systems. *SIAM journal on computing*, vol. 18, 1989, pp. 186–208.

[MS81] McEliece (R.J.) et Sarwate (D.V.). – On sharing secrets and reed-solomon codes. *Communications of the ACM*, vol. 24, septembre 1981, pp. 583–584.

[MS95] Miller (B. L.) et Shaw (M. J.). – *Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimization.* – Rapport de recherche, IlliGAL, décembre 1995.

[MU01] Meerwald (Peter) et Uhl (Andreas). – A survey of wavelet-domain watermarking algorithms. *In : SPIE Photonics West, Electronic Imaging 2001, Security and Watermarking of Multimedia Contents III.* – San Jose, CA, USA, janvier 2001.

[MvOV99] MENEZES (A. J.), VAN OORSCHOT (P. C.) et VANSTONE (S. A.). – *Handbook of applied cryptography.* – CRC Press, juillet 1999.

[Nob88] NOBLE (J.). – Finding corners. *Image and Vision Computing*, 1988, pp. 121–128.

[ODB96] O'RUANAIDH (J.), DOWLING (W.J.) et BOLAND (F.M.). – Phase watermarking of digital images. *In : IEEE-ICIP'96*, pp. 239–242. – Lausanne, Switzerland, octobre 1996.

[Ora01] ORAM (ANDY) (édité par). – *Peer-to-Peer : Harnessing the Power of Disruptive Technologies.* – O'Reilly, mars 2001.

[PA01] PICARD (J.) et A. ROBERT. – Neural networks functions for public key watermarking. *In : 4th Workshop on Information Hiding*, éd. par MOSKOWITZ (IRA S.), Lecture Notes in Computer Science. pp. 142–156. – Pittsburgh, PA, USA, avril 2001.

[PAK98] PETITCOLAS (F. A. P.), ANDERSON (R. J.) et KUHN (M. G.). – Attacks on copyright marking systems. *In : Lecture Notes in Computer Science.* Workshop on Information Hiding, pp. 218–238. – Portland, OR, USA, avril 1998.

[Pet00] PETITCOLAS (F. A.P.). – Watermarking schemes evaluation. *IEEE Signal Processing*, vol. 17, septembre 2000, pp. 58–64. – ISSN 1053-5888.

[PH02] PROVOS (N.) et HONEYMAN (P.). – Detecting steganographic content on the internet. *In : Proceedings of the ISOC NDSS 2002.* – San Diego, CAUSA, février 2002.

[Pit96] PITAS (I.). – A method for signature casting on digital images. *In : IEEE International Conference on Image Processing (ICIP'96)*, pp. 215–218. – Lausanne, Switzerland, September 1996.

[PJ96] PUATE (J.) et JORDAN (F.). – Using fractal compression scheme to embed a digital signature into an image. *In : Proceedings of SPIE Photonics East'96 Symposium.* – Boston, MA, USA, novembre 1996.

[Pra78] PRATT (W. K.). – *Digital Image Processing.* – New York, USA, John Wiley and Sons, 1978.

[Pro01] PROVOS (N.). – Defending against statistical steganalysis. *In : Proceedings of the 10th USENIX Security Symposium.* – Washington DC, USA, août 2001.

[PS94] PROAKIS (J.) et SALEHI (M.). – Communication systems engineering, 1994.

[PSR+01] PETITCOLAS (F. A. P.), STEINEBACH (M.), RAYNAL (F.), DITTMAN (J.), FONTAINE (C.) et FATÈS (N.). – A public automated web-based evaluation service for watermarking schemes : Stirmark benchmark. *In : Electronic imaging, security and watermarking of multimedia contents.* Society for Imaging Science and Technology (IS&T) and International Society for Optical Engineering (SPIE). – San Jose, CA, USA, janvier 2001. ISSN 0277-786X.

[PVM+01] Pereira (S.), Voloshynovskiy (S.), Madueño (M.), Marchand-Maillet (S.) et Pun (T.). – Second generation benchmarking and application oriented evaluation. *In : Proceedings of Information Hiding Workshop*, LNCS. – Pittsburgh, PA, USA, avril 2001.

[RD98] Roche (S.) et Dugelay (J.-L.). – Image watermarking based on the fractal transform. *In : IEEE MMSP'98*. – Los Angeles, USA, décembre 1998.

[RD00] Rey (Christian) et Dugelay (Jean-Luc). – Blind detection of malicious alterations on still images using robust watermarks. *In : IEEE Secure Images and Image Authentication Colloqium*. – London, UK, avril 2000.

[Rec73] Rechenberg (I.). – *Evolutions Strategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. – Stuttgart, Germany, Frommann-Holzboog, 1973.

[RLCS99] Raynal (F.), Lutton (E.), Collet (P.) et Schoenauer (M.). – Manipulation of non-linear IFS attractors using genetic programming. *In : Proceedings of the Congress of Evolutionary Computation*. pp. 1171–1177. – Washington DC, USA, juillet 1999.

[Row96] Rowland (C. H.). – Covert channels in the tcp/ip protocol suite, mars 1996. http://www.firstmonday.dk/issues/issue2_5/rowland/.

[RPF01] Raynal (F.), Petitcolas (F. A. P.) et Fontaine (C.). – évaluation automatique des méthodes de tatouage. *A paraître la revue Taitement du signal*, 2001.

[RSA77] Rivest (R. L.), Shamir (A.) et Adelman (L. M.). – *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. – Rapport de recherche, MIT, 1977.

[RSA78] Rivest (R.), Shamir (A.) et Adlemann (L.). – A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, vol. 21, décembre 1978, pp. 120–126.

[SC96] Schneider (M.) et Chang (S.F.). – A robust content based digital signature for image authentication. *In : Proceedings of IEEE-ICIP'96*, pp. 227–230. – Lausanne, Switzerland, 1996.

[Sch96] Schneier (B.). – *Applied cryptography*. – J. Wiley and sons, 1996.

[SD99] Smith (J.) et Dodge (C.). – Developments in steganography. *In : Proceedings of the 3rd Workshop on Information Hiding*. pp. 77–87. – Dresden, Germany, septembre 1999.

[SG00] Su (J. K.) et Girod (B.). – Fundamental performance limits of power-spectrum condition-compliant watermarks. *In : Electronic imaging, security and watermarking of multimedia content II*. Society for Imaging Science and Technology (IS&T) and International Society for Optical Engineering (SPIE), pp. 314–325. – San Jose, CA, USA, janvier 2000. ISSN 0277-786X. ISBN 0-8194-3598-9.

[Sha48] SHANNON (C. E.). – A mathematical theory of communication. *Bell System Tech. J.*, vol. 27, 1948, pp. 379–423, 623–656.

[Sha49a] SHANNON (C. E.). – Communication in the presence of noise. *Proceedings of the IRE*, vol. 37, 1949, pp. 10–21.

[Sha49b] SHANNON (C. E.). – Communication theory of secrecy systems. *Bell Systen Technicl Journal*, vol. 28, 1949, pp. 656–715.

[Sha79] SHAMIR (A.). – How to share a secret. *Communications of the ACM*, vol. 22, 1979, pp. 612–613.

[Sim84] SIMMONS (G. J.). – The prisoners' problem and the subliminal channel. *In : Advances in Cryptology.* CRYPTO '83, pp. 51–67. – Plenum Press, 1984.

[Smi83] SMITH (S. F.). – Flexible learning of problem solving heuristics through adaptive search. *In : Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI-83).* pp. 421–425. – Los Altos, CA, USA, 1983.

[Soc98] The Society for Imaging Science and Technology (IS&T) and the International Society for Optical Engineering (SPIE). – *Optical Security and Counterfeit Deterrence Techniques II.* – San Jose, CA, USA, janvier 1998. ISSN 0277-786X, ISBN 0-8194-2556-7.

[Soc99] The Society for Imaging Science and Technology (IS&T) and the International Society for Optical Engineering (SPIE). – *Security and Watermarking of Multimedia Contents.* – San Jose, CA, USA, janvier 1999. ISSN 0277-786X, ISBN 0-8194-3128-1.

[SPR+01] STEINEBACH (M.), PETITCOLAS (F. A. P.), RAYNAL (F.), DITTMAN (J.), FONTAINE (C.), SEIBEL (C.) et FATÈS (N.). – Stirmark benchmark : audio watermarking attacks. *In : Multimedia Security.* IEEE International Conference on Information Technology : Coding and Computing (ITCC'2001). – Las Vegas, NV, USA, avril 2001.

[SSL] SOMMERFELD (B.), SCHILLER (J.) et LEECH (M.). – Normalisation du protocole ssh. http://www.ietf.org/html.charters/secsh-charter.html.

[ST01] STERN (J.) et TILLICH (J.-P.). – Automatic detection of a watermarked document using a private key. *In : Proceedings of Information Hiding Workshop*, LNCS. – Pittsburgh, PA, USA, avril 2001.

[Ste01] STERN (J.). – *Contribution à une théorie de la protection de l'information.* – Thèse de doctorat, Paris XI Orsay, mars 2001.

[SWT01] SONG (D.), WAGNER (D.) et TIAN (X.). – Timing analysis of keystrokes and timing attacks on ssh. *In : Proceedings of the 10th USENIX Security Symposium.* – août 2001. http://paris.cs.berkeley.edu/~dawnsong/ssh-timing.html.

[Tri95] TRICOT (C.). – *Curves and fractal dimensions.* – Springer-Verlag, 1995.

[TRvS+93] TIRKEL (A. Z.), RANKIN (G. A.), VAN SCHYNDEL (R. M.), HO (W. J.), MEE (N. R. A.) et OSBORNE (C. F.). – Electronic watermark. *In : Digital Image Computing, Technology and Applications (DICTA'93)*, pp. 666–673. – Macquarie University, Sidney, Australia, 1993.

[Tur89] TURNER (L. F.). – Digital data security system, 1989. Patent IPN WO 89/08915.

[VPIP01] VOLOSHYNOVSKIY (S.), PEREIRA (S.), IQUISE (V.) et PUN (T.). – Attack modelling : Towards a second generation benchmark. *Signal Processing*, vol. 81, June 2001, pp. 1177–1214. – Special Issue : Information Theoretic Issues in Digital Watermarking, 2001. V. Cappellini, M. Barni, F. Bartolini, Eds.

[VPP+01] VOLOSHYNOVSKIY (S.), PEREIRA (S.), PUN (T.), EGGERS (J.) et SU (J.). – Attacks on digital watermarks : Classification, estimation-based attacks and benchmarks. *IEEE Communications Magazine (Special Issue on Watermarking)*, 2001. – F. Pérez-González, Ed. Invited paper. (to appear).

[Vrs90] VRSCAY (E. R.). – Moment and collage methods for inverse problem of fractal cosntruction with IFS. *In : proceedings of Fractal 90.* – Lisbone, Portugal, juin 1990.

[vSTO94] VAN SCHYNDEL (R. G.), TIRKEL (A. Z.) et OSBORNE (C. F.). – A digital watermark. *In : IEEE International Conference on Image Processing (ICIP)*, pp. 86–90. – Austin, TX, USA, 1994.

[Wat87] WATSON (ANDREW B.). – Efficiency of an image code based on human vision. *Journal of the Optical Society of America A*, vol. 4, n12, 1987, pp. 2401–2417.

[Way92] WAYNER (P.). – Mimic functions. *Cryptologia*, vol. 16, 1992, pp. 193–214.

[Way95] WAYNER (P.). – Strong theoretical steganography. *Cryptologia*, vol. 19, 1995, pp. 285–299.

[WD96] WOLFGANG (R.B.) et DELP (E.J.). – A watermark for digital images. *In : IEEE-ICIP'96*, pp. 219–222. – Lausanne, Switzerland, 1996.

[WG89] WILSON (S. W.) et GOLDBERG (D. E.). – A critical review of classifier systems. *In : Proceedings of the 3rd International Conference on Genetic Algorithms.* pp. 244–255. – Morgan Kauffman, 1989.

[WL98] WU (M.) et LIU (B.). – Watermarking for image authentication. *In : Proceedings of the IEEE International Conference on Image Processing (ICIP'98).* – Chicago, IL, USA, 1998.

[WP00] Westfeld (A.) et Pfitzmann (A.). – Attacks on steganographic systems. *In : Proceedings of Information Hiding (IH99)*, Lecture Notes in Computer Science. – Springer, 2000.

[YG93] Yin (X.) et Germay (N.). – A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. *In : Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*. pp. 450–457. – Springer Verlag, 1993.

[YM97] Yeung (M.M.) et Mintzer (F.C.). – An invisible watermarking technique for image verification. *In : Proceedings of International Conference on Image Processing*, pp. 680–683. – 1997.

[Yuv79] Yuval (G.). – How to swindle Rabin. *Cryptologia*, vol. 3, 1979, pp. 187–189.

[ZFK+98] Zöllner (J.), Federath (H.), Klimant (H.), Pfitzmann (A.), Piotraschke (R.), Westfeld (A.), Wicke (G.) et Wolf (G.). – Modelling the security of steganographic systems. *In : IWIH : International Workshop on Information Hiding*. – 1998.

[ZK98] Zhao (J.) et Koch (E.). – A generic digital watermarking model. *Computers and Graphics*, vol. 22, juillet 1998.