Projet de veille technologique
- La Stéganographie -

Rémy ALLAIS
François DUMONT
Alexandre PELTIER

23 juin 2004

Résumé

Ce rapport a été rédigé dans le cadre du projet de veille Technologique de Maîtrise informatique et réseaux à l'Institut Universitaire Professionnel de Blois.

La connaissance est un élément constitutif et déterminant du progrès de l'humanité. Tantôt une épée "le renseignement" garantissant l'accès ou le maintien au pouvoir, tantôt un bouclier "remède" contribuant à l'amélioration de notre bien être, l'information est recherchée et convoitée. L'intégrité et l'authenticité de l'information doivent être garanties. Vouloir l'acquérir ou l'échanger nécessite sa protection. La confidentialité de l'information et des échanges repose sur des mécanismes visant à dissimuler/soustraire l'information aux yeux de tous ceux qui ne doivent pas en avoir connaissance. Le chiffrement de ces données est un moyen, mais il en existe au moins un second.

La stéganographie est l'art de faire passer de l'information en dissimulant son existence même. Le but de la stéganographie est d'éviter d'attirer l'attention sur la transmission d'un message caché. On rend anonyme l'information utile dans un flot d'informations anodines. Si des soupçons existent, alors l'objectif n'est pas atteint. La stéganalyse est l'art de découvrir et de rendre inutile la dissimulation des messages.

Ce rapport a pour but d'identifier les caractéristiques de la stéganographie, puis de procéder à un état de l'art des logiciels disponibles enfin, des limites inhérentes à la mise en oeuvre de cette technique.

Table des matières

Chapitre 1

Introduction

Ce rapport a pour but de présenter l'état de l'art de la stéganographie ainsi que l'ensemble des logiciels actuellement disponibles. Avant toute chose, il est nécessaire de maîtriser différents concepts pour appréhender la stéganographie. Après avoir dressé l'historique du domaine, nous définirons les notions utilisées. Une étude des méthodes et algorithmes utilisés sera ensuite effectuée. Un comparatif des logiciels disponibles sera établie. Les applications et les limites de la stéganographie seront ensuite expliquées. Enfin, l'évolution et les enjeux de la stéganographie seront évoqués.

Pour réaliser l'objectif fixé du rapport, nous avons développé une méthodologie. Afin de fixer un cadre réaliste à notre propos, une définition claire du domaine d'étude a été effectuée.

1.1 Méthodologie

Ce projet a été découpé en trois parties : collecte d'informations, étude des applications disponibles, rédaction du rapport.

La recherche et collecte d'informations s'est avérée la partie la plus longue. En effet, la stéganographie est liée à la sécurité informatique. Mais, paradoxalement, assez peu d'informations se sont révélées disponibles sur ce sujet. Par ailleurs, aucun livre n'a été publié en français. De plus, les documents traitant de sécurité évoquent à peine la stéganographie. L'exhaustivité de notre recherche repose essentiellement sur les papiers de recherche et Internet.

Ce rapport est rédigé sous une forme synthétique et à titre purement informatif. Le cadre de notre étude qui a été défini a induit des contraintes qui ont conduit à opérer des sélections précises dans certaines parties du domaine d'étude. Certaines connaissances sont requises pour faciliter la compréhension des concepts liés aux algorithmes employés. Afin de mieux comprendre la stéganographie, des exemples de systèmes

ont été étudiés. Les tutoriaux des logiciels considérés comme les plus efficaces ont été placés en Annexe.

1.2 Terminologie

La terminologie définie dans ce chapitre sera conservée tout au long du rapport, afin de faciliter la distinction des différents éléments intervenant dans la dissimulation d'informations.

Le support au sein duquel les informations sont dissimulées est nommé "**medium de couverture**", ou "**medium**". Une fois les informations insérées, nous utilisons alors l'expression "**stégo-medium**" ou "**stégo-objet**". De façon générale, les "**données**" qualifient les informations cachées dans le "**stégo-medium**".

Le processus complet de dissimulation d'informations repose sur deux processus :

1. la dissimulation : consistant à insérer une information dans le medium ;
2. l'extraction : restitution de l'information cachée. Le vocable "**détection**" est également utilisé lorsqu'il s'agit de vérifier la présence d'une information dans le stégo-medium, sans procéder à son extraction.

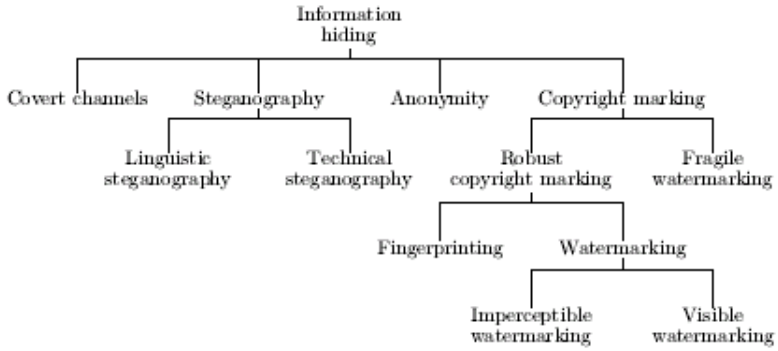
1.3 Qu'est-ce que la stéganographie ?

Dans cette section, nous allons d'abord expliquer ce qu'est la dissimulation d'information, puis distinguer brièvement la stéganographie de la cryptographie.

1.3.1 La dissimulation d'information

La stéganographie fait partie du domaine de la dissimulation d'information comme l'illustre la figure 1.1

FIG. 1.1 – La dissimulation d'information et ses sous-ensembles



Selon les buts poursuivis, trois méthodes principales assurent la dissimulation d'informations : la stéganographie, le tatouage et le fingerprinting. Deux méthodes secondaires permettent aussi la dissimulation d'informations : les canaux cachés et l'anonymat.

1.3.1.1 La stéganographie

La stéganographie vise à cacher un message, dans un medium afin que nul ne puisse distinguer le medium vierge du stégo-medium. La nature de l'information dissimulée ne revêt pas d'importance : il peut s'agir d'un texte en clair ou de sa version chiffrée. Ce message n'a à priori aucun lien avec le stégo-medium qui le véhicule.

Lorsqu'une personne non autorisée - l'attaquant - tente uniquement de déceler la présence d'un message transitant dans un medium sur le canal de communication, elle est dite passive. La plupart des solutions de stéganographie considèrent exclusivement ce type d'attaquant. Toutefois, il peut aussi être actif : il sait que le stégo-medium contient l'information. Il tente de la modifier ou de l'extraire.

1.3.1.2 Le tatouage numérique

Le tatouage a une portée commerciale. Il tente de prévenir les contournements des droits d'auteurs. Un client tente de détecter la présence possible d'une marque dans un medium, puis dans l'affirmative, vérifie si l'utilisateur a bien acquitté les droits (généralement en payant une licence). Il s'agit de dissimulation d'information car, pour y parvenir, on insère un tatouage (ou marque, ou filigrane) dans le medium spécifique au propriétaire. Comme celui-ci souhaite protéger son medium et non une version clonée déformée. L'insertion du tatouage doit limiter les modifications subies par le

medium. Par suite, chaque copie du stégo-medium contient une marque identique : celle du propriétaire légal. Il ne s'agit pas de dissimulation à proprement parler. La présence du tatouage dans le stégo-medium est connue. Cependant cette connaissance est insuffisante. L'effet a obtenir est préventif. Les modifications apportées au stégo-medium tatoué attestent d'une contrefaçon.

1.3.1.3 Le fingerprinting

Le fingerprinting assure la détection des copies illégales d'un stégo-medium. Chaque utilisateur authentifié reçoit sa propre copie du medium qui contient une empreinte : l'identifiant. Ainsi, lorsqu'une copie illégale est découverte, la lecture de l'empreinte indique la source de la fuite. A la différence du tatouage où l'origine du medium est déterminante, le fingerprinting se préoccupe du destinataire . Chaque copie du medium contient une information différente, relative à son utilisateur, rendant alors chaque stégo-medium unique.

1.3.1.4 Autres domaines

On trouve deux autres méthodes pouvant assurer la dissimulation d'informations :

- Les canaux cachés (*covert channels*) sont des canaux utilisés que l'on trouve parfois dans des programmes non éprouvés pouvant permettre la fuite d'informations. Défaut de conception ou mécanisme d'évasion intentionnel, leur connaissance peut assurer le transport d'une information. Dans ce cas, le vecteur et sa charge sont dissimulés.
- L'anonymat est utilisé lorsque l'on veut cacher l'émetteur et le destinataire d'un message. Le message proprement dit n'est quant à lui pas nécessairement protégé.

Comme nous venons de le voir, l'art de la dissimulation d'informations est vaste. Le cadre de notre étude se limite uniquement à la stéganographie linguistique et technique.

1.3.2 Cryptographie et stéganographie

L'une des principales difficultés lorsque l'on aborde la stéganographie est de la distinguer de la cryptographie. La stéganographie (du grec *"steganos"*, dissimulé et *"graphein"*, écrire) est une sous-discipline de la dissimulation d'informations. Cette dernière, déjà très ancienne, consiste à dissimuler un message, que l'on désire transmettre confidentiellement, dans un flux de données d'apparence anodine, de façon que sa présence soit imperceptible. Comme dans le cas de la cryptographie, la technique permet d'échanger des messages avec un correspondant sans que des personnes non autorisées ne puissent en prendre connaissance. Mais alors qu'avec la cryptographie traditionnelle, la sécurité repose sur le fait que le message est incompréhensible, en matière de

stéganographie, la sécurité repose sur la remise en question même de m'existence du message.

Le tableau suivant permet d'évaluer rapidement les différences entre la stéganographie et la cryptographie :

CRYPTOGRAPHIE	STÉGANOGRAPHIE
Le message n'est pas caché	Le message est caché
L'interception est rendu possible	L'ennemi va tenter de découvrir le médium de couverture
L'ennemi va tenter de déchiffrer le message	

La stéganographie a aujourd'hui pris un autre sens. Une définition plus appropriée pourrait consister à :

Assurer la dissimulation d'une information dans un flux de données numériques, tel qu'un fichier numérique image ou son. Ce type de fichiers est habituellement considéré comme inoffensif, incapable de contenir des informations autres que celles normalement prévues.

Chapitre 2

Pré-requis

Cette section a pour but d'introduire une série de notions nécessaires à la bonne compréhension du rapport. La stéganographie s'appuie principalement sur le traitement du signal, sur les formats d'images et de sons numériques.

2.1 Le bit de poids faible (LSB)

Toute donnée numérique est stockée dans un octet (byte) de 8 bits.

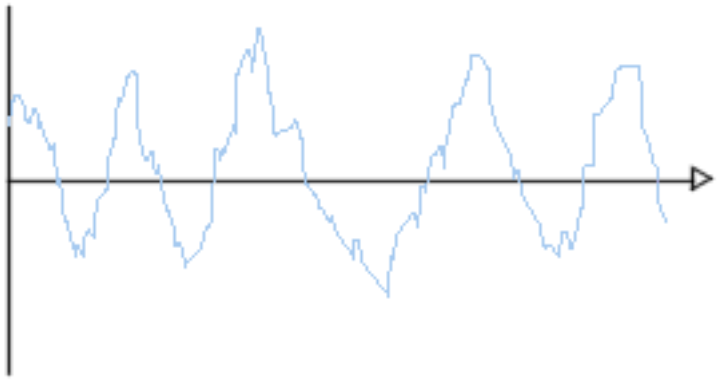
7	6	5	4	3	2	1	0
msb							lsb
0	1	1	0	1	0	1	1

Le LSB est le bit de poids faible (Least Significant Bit). En changeant de 0 à 1 ou de 1 à 0, ce bit fait varier de très peu la valeur globale de l'octet.

2.2 Le codage du son numérique

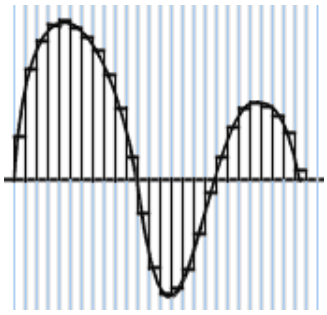
L'onde sonore se représente dans le temps. Une illustration est donnée en figure 2.1.

FIG. 2.1 – Représentation de l'onde sonore



Le son est divisé en échantillons, dont l'amplitude est codée sur 8 ou 16 bits. Cela s'appelle la profondeur de l'échantillon (voir figure 2.2).

FIG. 2.2 – Représentation numérique du son



2.2.1 Le format WAV

Le format WAV est un format numérique de codage du son, identique à celui utilisé pour les CD. La profondeur des échantillons est codée généralement sur 16 bits. La modification du bit de poids faible dans ce cas impliquera un changement d'amplitude d'un échantillon inaudible par l'oreille humaine. La fréquence 44100 échantillons par seconde correspond à la qualité CD.

2.3 Les formats d'images numériques

La stéganographie s'appuie souvent sur des fichiers numériques de type images. Les formats sont connues, et depuis la généralisation d'Internet, certains sont très employés.

2.3.1 L'image

Une image est une matrice composée de cellules appelées pixels. Il existe plusieurs manière de coder un pixel.

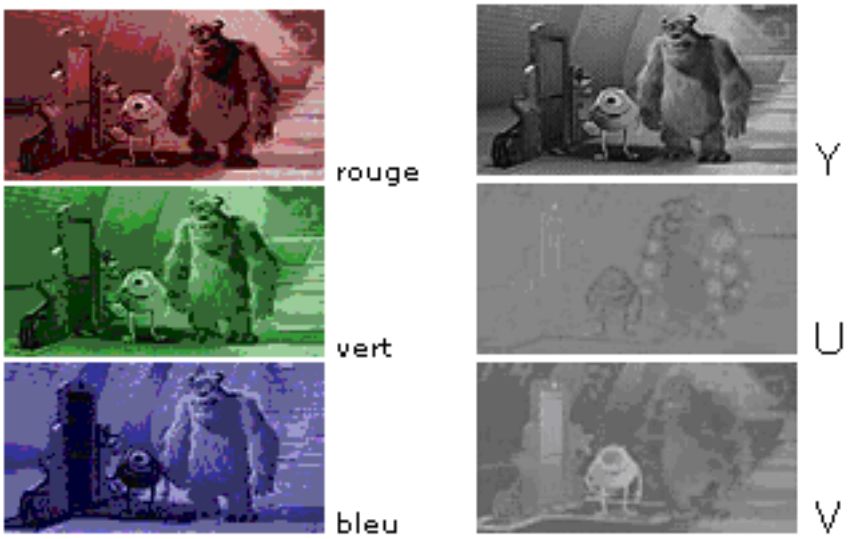
2.3.1.1 Le codage RVB (ou RGB)

RVB (Rouge Vert Bleu) permet de décomposer la couleur des pixels des images numériques en 3 sous-composantes.

2.3.1.2 Le codage couleur YUV

Y représente la luminance et U, V, les composantes Rouge et Bleu (chrominances). Ces 3 informations permettent de restituer au final les composantes RVB et apporte l'avantage de permettre une compression facile des couleurs et de fiabiliser le transport du signal (voir figure 2.3).

FIG. 2.3 – Représentation numérique du son



2.3.2 Le format BMP

L'image est formée de pixels (points élémentaires) qui ont chacun leur propre couleur. Le codage de l'image se fait en écrivant successivement les bits correspondant à chaque pixel, ligne par ligne en commençant par le pixel en bas à gauche.

- Les images en 2 couleurs utilisent 1 bit par pixel, ce qui signifie qu'un octet permet de coder 8 pixels ;
- Les images en 16 couleurs utilisent 4 bits par pixel, ce qui signifie qu'un octet permet de coder 2 pixels ;
- Les images en 256 couleurs utilisent 8 bits par pixel, ce qui signifie qu'un octet code chaque pixel ;
- Les images en couleurs réelles utilisent 24 bits par pixel, ce qui signifie qu'il faut 3 octets pour coder chaque pixel, en prenant soin de respecter l'ordre de l'alternance bleu, vert et rouge.

Chaque ligne de l'image doit comporter un nombre total d'octets qui soit un multiple de 4 ; si ce n'est pas le cas, la ligne doit être complétée par des 0 afin de respecter ce critère.

2.3.3 Le format GIF

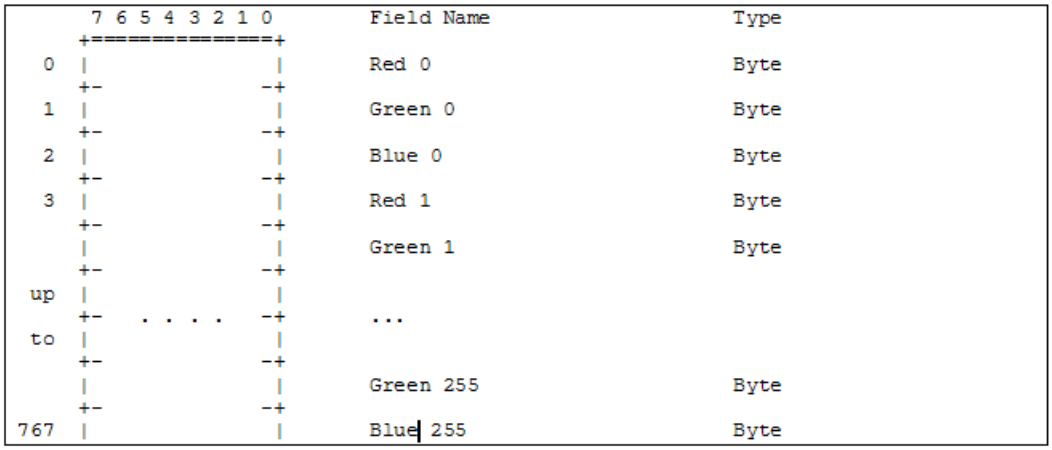
Une image GIF (Graphic Interchange Format) peut contenir de 2 à 256 couleurs (2, 4, 8, 16, 32, 64, 128 ou 256) parmi 16.8 millions dans sa palette. Ainsi grâce à cette palette limitée en nombre de couleurs (et non limité en couleurs différentes), les images obtenues par ce format ont une taille généralement très faible.

La palette d'une image référence toutes les couleurs utilisées par les pixels de l'image (voir figure 2.4).

FIG. 2.4 – Une image GIF et sa palette de couleur

Le format d'un fichier GIF est le suivant :

FIG. 2.5 – Format GIF



	7 6 5 4 3 2 1 0	Field Name	Type
0		Red 0	Byte
1		Green 0	Byte
2		Blue 0	Byte
3		Red 1	Byte
		Green 1	Byte
up			
to		...	
		Green 255	Byte
767		Blue 255	Byte

2.3.4 Le format JPEG

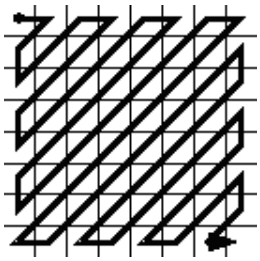
La compression JPEG est une compression avec pertes, ce qui lui permet, en dépit d'une perte de qualité, un des meilleurs taux de compression (20 :1 à 25 :1 sans perte notable de qualité).

Cette méthode de compression est beaucoup plus efficace sur les images photographiques (comportant de nombreux pixels de couleurs différentes) plutôt que sur les images géométriques (on utilise la compression LZW) car sur ces dernières les différences de nuances dues à la compression sont très visibles.

Les étapes de la compression JPEG sont les suivantes :

1. Rééchantillonnage de la chrominance, car l'oeil ne peut discerner de différences de chrominance au sein d'un carré de 2x2 points ;
2. Découpage de l'image en blocs de 8x8 points, puis application de la fonction TCD qui décompose l'image en somme de fréquences ;
3. Quantification de chaque bloc, c'est-à-dire qu'il est appliqué un coefficient de perte (permettant de déterminer le ratio taille/qualité) qui "annulera" ou diminuera des valeurs de hautes fréquences, afin d'atténuer les détails en parcourant le bloc intelligemment avec un codage RLE (en zig-zag pour enlever un maximum de valeurs nulles, voir figure 2.6) ;
4. Encodage de l'image puis compression avec la méthode d'Huffman.

FIG. 2.6 – Illustration de la méthode zig-zag

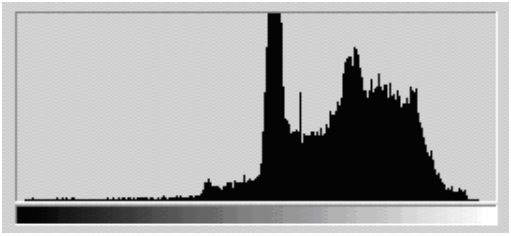


Le format de fichier embarquant un flux codé en JPEG est en réalité appelés JFIF (JPEG File Interchange Format, soit en français Format d'échange de fichiers JPEG), mais par déformation le terme de "fichier JPEG" est couramment utilisé.

2.4 Transformée en cosinus discret (TCD)

Cette opération mathématique permet de traiter l'image comme un signal numérique. Cela traduit la répartition des couleurs dans le domaine fréquentiel (voir figure 2.7).

FIG. 2.7 – Exemple d'histogramme représentant les fréquences contenues dans une image



Il résulte de cette transformée une matrice de coefficients TCD, qui permettent de ne conserver que les bandes de fréquence utiles. C'est ce qui donne toute de l'efficacité de la compression JPEG.

Chapitre 3

Histoire de la stéganographie

En 1996 s'est déroulé la première conférence académique sur le sujet[1], mais cette discipline est très ancienne et a subi de multiples évolutions. On distingue actuellement deux types de stéganographie : la stéganographie linguistique et la stéganographie technique. La stéganographie technique trouve ses origines dans l'antiquité alors que la stéganographie linguistique est relativement récente.

3.1 Antiquité

L'apparition de la stéganographie remonte à l'époque des grecs mais la Chine a également innovée dans le domaine de la dissimulation d'informations.

3.1.1 En Grèce

Hérodote raconte comment Dematurus au 5ème siècle avant J.-C. fit parvenir à ses compatriotes Grecs le plan d'invasion des Perses. Il gratta la cire sur une paire de tablettes de bois, il inscrivit sur le bois son message, puis couvrit à nouveau le message de cire. Les tablettes, en apparence vierges, purent ainsi tromper la vigilance des gardes le long du parcours. Hérodote raconte également l'histoire de Histaïaeus qui rasa la tête de son messager, écrivit le message sur son crâne, puis attendit que les cheveux de l'agent repoussent.

Les Grecs mettront en place plusieurs mécanismes dédiés à la stéganographie : des trous sur un disque représentant des lettres et des fils de couleurs différentes, permettant de lire un mot ; une autre technique consistait à percer un petit trou sur les lettres d'un document et constituer ainsi un message.

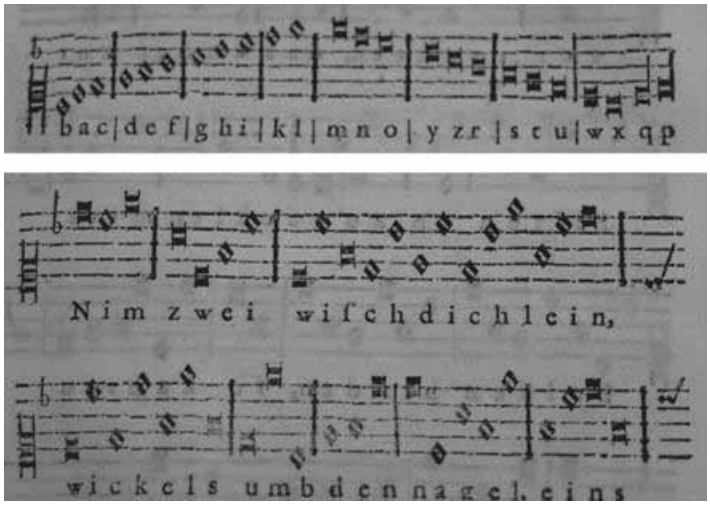
[1] R. J. Anderson, *Information Hiding : first international workshop*, vol. 1174 de Lecture Notes in Computer Science, Isaac Newton Institute, Mai 1996.

3.1.2 En Chine

En Chine on écrivait le message sur de la soie, qui ensuite était ensuite placée dans une petite boule recouverte de cire. Le messager avalait ensuite cette boule.

3.2 Améliorations des techniques

FIG. 3.1 – G. Schott utilise une simple correspondance entre les lettres de l'alphabet et notes.



Au cours des 16ème et 17ème siècles, une littérature importante sur la stéganographie existait déjà. Dans son livre Schola Steganographica, Gaspar Schott (1608-1666) explique comment cacher des messages dans les partitions de musique (voir figure 3.1) : chaque note correspond à une lettre. Schott a également étendu le code "Ave Maria" proposé par L'abbé Trithème dans Polygraphioe (1516), l'un des premiers livres connus traitant de cryptographie et de stéganographie. Le code étendu utilise 40 tables, chacune contenant 24 entrées (une pour chaque lettre de l'alphabet) dans quatre langues : latin, allemand, italien et français. Chaque lettre du texte clair est remplacée par un mot ou une phrase apparaissant dans la table à la ligne correspondant à la lettre choisie. Après l'opération, le texte résultant ressemble à une prière, une simple lettre de correspondance, ou encore une formule de magie.

Enée le Tacticien proposa de cacher un message dans un autre texte en changeant la hauteur des lettres ou en perçant de petits trous au dessus et en dessous des lettres du message de couverture. Wilkins améliora cette technique en utilisant des encres "invisibles", souvent du jus de citron, d'oignon ou de chlorure d'ammoniac pour inscrire

des points. Les espions allemands reprirent cette technique durant les deux guerres mondiales. Une adaptation moderne est toujours utilisée pour sécuriser les documents électroniques et se contente d'imprimer des points minuscules pour coder un nom d'utilisateur, etc. L'utilisation d'encre invisible fut aussi utilisée pendant la guerre de sécession entre George Whashington et des agents secrets nordistes.

En 1857, Brewster suggérait de cacher des messages secrets dans un espace à peine plus grand qu'un point d'encre. En 1860, le problème de réduction des photographies était résolu par René Dragon (1819-1900). Ainsi les messages sur micro-film firent leur apparition, d'abord durant la guerre de 1870 puis à l'occasion de la première guerre mondiale. Ils étaient ensuite apposés sous forme de points ou point-virgule dans des magazines par exemple.

Avec les progrès réalisés en chimie, l'utilisation d'encre invisible devint plus efficace. Par exemple, les photocopieurs émettent un fort rayonnement d'ultraviolet. Pour éviter la reproduction d'un billet par un photocopieur, certains motifs sont dessinés à l'aide d'une encre qui devient visible lors de l'exposition aux ultraviolets.

En 1883, Auguste Kerckhoffs a énoncé le premier principe de la cryptographie moderne. Il affirma qu'il fallait supposer la méthode de chiffrement connue de l'ennemi et que par conséquent, la sécurité du système ne devait résider que dans le choix de la clé utilisée pour le chiffrement. En appliquant ce principe à la stéganographie, on obtient une tentative de définition d'un stégo-système sécurisé.

3.3 Histoire de la stéganographie linguistique

Une méthode très utilisée en stéganographie linguistique est l'**acrostiche** (chaque première lettre de la phrase forme une autre phrase). L'exemple le plus long jamais écrit est l'*Amorosa visione* de Giovanni Boccaccio (1313-1375). Boccaccio écrivit d'abord trois sonnets contenant à eux trois environ 1500 lettres, puis écrivit un poème de façon que la première lettre des tercets[2] successifs corresponde exactement aux lettres des sonnets.

Une extension de la méthode porte le nom de "*messages enchâssés*". Il s'agit d'images et/ou des mots qui sont discrètement dissimulés dans un message plus grand. Un autre exemple, plus contemporain, découvert dans le magazine français PC Soluces de Noël 1998 permettait de lire relatif à la qualité de la suite du jeu Tomb Raider.

Dans *The Code Breakers*, D. Kahn raconte comment des prisonniers de guerre cachaient des messages dans des lettres envoyées à leur famille en utilisant les points et barres sur les lettres i, j, t, et f à la façon du code Morse.

[2] Un *tercet* est une strophe qui comporte trois vers. Dans un sonnet, on trouvera deux tercets et deux quatrains.

Chapitre 4

Etat de l'art

Ce chapitre a pour but de présenter les techniques les plus significatives de la stéganographie et le principe général qu'elle adopte. Plusieurs supports numériques peuvent être utilisés pour dissimuler des données : le texte, l'image, l'audio, les exécutables et les supports physiques.

Il y a plusieurs approches pour classifier les techniques des systèmes stéganographiques. La plus simple consiste à les regrouper par type de medium. Une autre méthode possible réside dans les modifications apportées au medium de couverture. C'est cette dernière qui a été retenue et nous avons dénombré six catégories :

1. Les **systèmes de substitution** qui substituent les parties redondantes d'un medium par le message secret ;
2. Les **techniques dans le domaine transformé** qui encapsulent le message secret dans le domaine des fréquences ;
3. Les **techniques à étalement de spectre** qui imitent les techniques existantes dans le domaine des télécommunications ;
4. 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 ;
5. Les **techniques de distorsion** qui altèrent le support, la différence avec le support initial constituant alors le message ;
6. Les **méthodes de génération de couverture** qui créent un support autour du message pour le dissimuler.

Pour chaque algorithme abordé, il sera également évoqué l'efficacité et la résistance aux attaques.

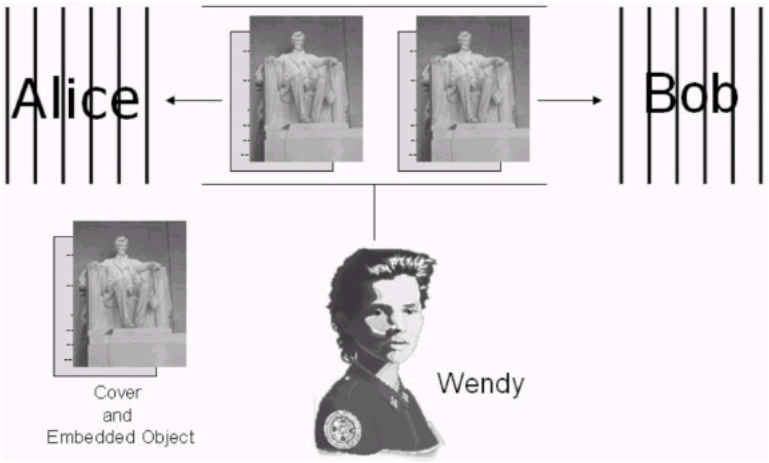
4.1 Principe de la stéganographie

Le principe général de la stéganographie a été clairement défini avec le modèle dit “du prisonnier”. Un schéma typique d’une communication secrète est ensuite établie. Enfin, ses caractéristiques sont décrites.

4.1.1 Le problème du prisonnier

Le modèle classique d’une communication invisible a été proposé pour la première fois par Simmons[1] avec le *“problème du prisonnier”*(figure 4.1).

FIG. 4.1 – Problème du prisonnier



Alice et Bob sont arrêtés pour un crime et sont placés dans des cellules distinctes. Ils souhaitent établir un plan d’évasion, mais malheureusement toutes communication entre eux sont arbitrées par une gardienne nommée Wendy. L’utilisation de cryptage est impossible, ils se verraient placer en isolement cellulaire (donc aucun échange possible). Les deux parties doivent donc communiquer de manière invisible pour ne pas attirer l’attention de Wendy. La solution : cacher une information dans un message anodin. Bob pourrait, par exemple, dessiner une vache bleue couchée dans une prairie verte et envoyer cette oeuvre d’art moderne à Alice. Wendy ne sait pas que les couleurs de l’image ont une signification[2].

Malheureusement, des problèmes peuvent entraver les intentions d’Alice et Bob. Wendy peut altérer le message que Bob a envoyé à Alice. Par exemple, elle pourrait changer la

[1] Simmons, G.J., « The Prisoners problem and the subliminal channel », in Advances in Cryptology, Proceedings of CRYPTO 83, Plenum Press, 1984, p.51-67

[2] On retiendra l’hypothèse qu’Alice et Bob ont accès à un ordinateur dans leur cellule et qu’ils peuvent s’envoyer des messages dans différents formats (texte, images, son, etc.).

couleur de la vache en rouge et ainsi détruire l'information cachée ; elle devient ainsi une gardienne active. Elle pourrait agir avec malveillance et contrefaire les messages. Elle pourrait ainsi envoyer un message à l'un en se faisant passer pour l'autre.

Ce modèle est applicable dans de nombreuses situations en stéganographie. Alice et Bob représentent les *deux parties de la communication* voulant échanger une information secrète de manière invisible. Wendy la gardienne est attentive. Elle est susceptible de lire les messages envoyés et capable de les altérer.

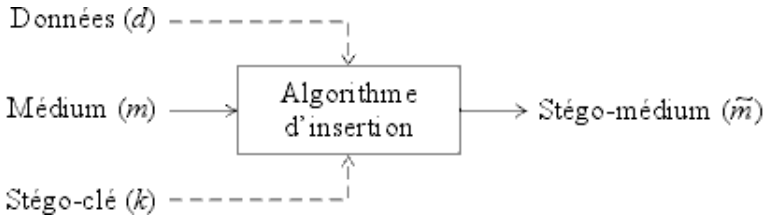
Tandis que les techniques de chiffrement visent à cacher le contenu d'un message, la stéganographie va plus loin en essayant de dissimuler la nature même de la communication. Deux personnes peuvent communiquer en secret en échangeant des messages banals contenant des informations confidentielles. Les deux parties doivent prendre en compte la présence d'un attaquant passif, actif ou malveillant.

4.1.2 Structure d'une communication secrète

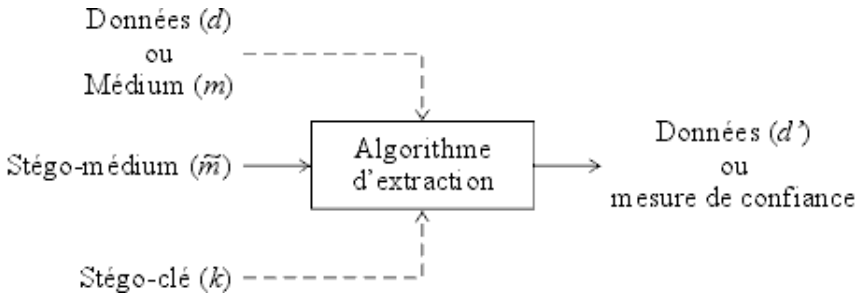
Le processus complet de la stéganographie repose sur deux opérations (figure 4.2) :

1. La dissimulation, qui consiste à insérer l'information dans le medium ;
2. L'extraction, qui récupère cette information. Le mot détection est également utilisé lorsqu'il s'agit de vérifier la présence d'une information (représentée grâce à un signal, une caractéristique particulière du medium) dans le stégo-medium, sans pour autant vouloir l'extraire.

FIG. 4.2 – Principe général



(a) Insertion des données dans le médium



(b) Extraction des données du médium

4.1.3 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 **simple** : 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 permet d'insérer puis extraire le message du stégo-medium ;
- 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, extrait le message à l'aide de sa clé privée.

4.1.4 Caractéristiques

La stéganographie repose sur deux principes essentiels :

– Un fichier numérique peut être altéré jusqu'à un certain degré sans perdre ses fonctionnalités.
– A certain degré, les sens de l'être humain ne sont pas suffisamment précis pour discerner les changements mineurs des fichiers.

Cependant, pour mesurer l'efficacité d'un système stéganographique d'une technique donnée, il existe trois critères :

* L'IMPERCEPTIBILITÉ : les données ne doivent pas être "perceptibles" dans le stégo-medium. Pour le tatouage ou le fingerprinting, l'objectif est de ne pas détériorer le stégo-medium 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 ;
* la CAPACITÉ : quantité de bits significatifs dissimulés dans le stégo-medium par unité d'accès (varie en fonction du medium) ;
* la ROBUSTESSE : aptitude de préservation des données cachées face aux modifications du stégo-medium.

Après avoir présenté les principes généraux et les caractéristiques caractérisant cette discipline, les techniques utilisées sont abordées. Notre but est de dissimuler de l'information.

4.2 Les systèmes de substitution

Utiliser la substitution revient à remplacer certaines données déjà présentes dans un fichier par l'information à cacher. Cette méthode peut sembler simpliste, mais il faut faire attention à ne pas supprimer de données importantes qui rendraient impossible la lecture du stégo-objet final. Typiquement, on substitue les données non primordiales par notre message. Le destinataire extrait l'information s'il a connaissance des positions où le message a été substitué. Puisque seules des modifications mineures ont été apportées dans le processus d'encapsulation, l'émetteur présume qu'elles ne seront pas détectées par un attaquant passif.

4.2.1 Substitution du bit de poids faible (LSB)

Le procédé de base consiste à modifier les bits de poids faible des octets constituant le medium. Cette méthode peut être appliquée sur tout type de support.

La figure 4.3 illustre le procédé d'encapsulation du message secret et son extraction. Pour encapsuler un message, on choisit dans un ensemble $\{j_1, ..., j_{l(m)}\}$ un medium de couverture et on exécute la substitution des LSB de c_μ avec m_i(c = couverture,

m = message). A l'extraction, il est juste nécessaire de connaître les bits qui ont été modifiés.

On peut citer StegoDos[3], S-Tools[4] (voir utilisation en annexe), EzStego[5], Hide and Seek[6], Hide4PGP[7], White Noise Storm[8], et Steganos[9] qui utilisent cette méthode.

4.2.1.1 ...dans une image

Un exemple typique de ce procédé est l'application d'une image 24-bits au format BMP non compressée (voir chapitre 2). Une clé secrète k détermine, dans une zone de l'image, l'emplacement des pixels.

La luminance de chaque pixel est codée par une valeur comprise entre 0 (soit en binaire : 00000000), et 255 (soit en binaire : 11111111). Le bit de poids faible de

[3]StegoDos, Black Wolf's Picture Encoder v0.90B, 1993

[4]A. Brown, "S-Tools for Windows", 1996

[5]R. Machado , "EzStego, Stego Online, Stego", http ://www.stego.com, 1997

[6]C. Maroney, "Hide and Seek", http ://www.rugeley.demon.co.uk/security, 1994-1997

[7]H. Repp, "Hide4PGP", 1996

[8]R. Arachelian, "White Noise Storm", 1994

[9]F. Hansmann, "Steganos, Deux Ex Machina Communications", http ://www.steganography.com, 1996

FIG. 4.3 – Algorithme général de la méthode de substitution de poids faible

```
for i = 1,...,l(c) do
  s_i ← c_i
end for
for i = 1 ...,l(m) do
  compute index j_i where to store ith message bit
  s_ji ← c_ji ⇋ m_i
end for
```

(a) Encapsulation du message

```
for i = 1,...,l(M) do
  compute index j_i where the ith message bit is stored
  m_i ← LSB(c_ij)
end for
```

(b) Extraction du message

l'écriture binaire, le dernier, est celui qui a le moins d'influence sur la valeur de la luminance. On cache l'information en introduisant un biais dans cette proportion : on impose une valeur aux bits de poids faible des pixels choisis selon la clé.

Par exemple, la lettre A peut être cachée dans 3 pixels (qui n'ont pas subi de compression). Soit le code binaire de 3 pixels (9 octets) suivant :

(00100111 11101001 11001000)

(00100111 11001000 11101001)

(11001000 00100111 11101001)

La valeur binaire de A est 10000011. En insérant la valeur binaire de A dans les 3 pixels, on obtiendrait le résultat suivant :

(0010011***1*** 1110100***0*** 1100100***0***)

(0010011***0*** 1100100***0*** 1110100***0***)

(1100100***0*** 0010011***1*** 1110100***1***)

Parmi les 8 bits utilisés, les bits en gras sont ceux qui ont été modifiés et en italique les bits de poids faible nécessaire pour comprendre le message entier. En moyenne, seule la moitié des bits utilisés sont modifiés. Cette modification est imperceptible pour l'oeil humain, car la luminance de ces pixels aura varié d'au plus 1. On peut aussi cacher des données dans le deuxième plus petit bit significatif sans que l'oeil humain ne discerne le changement. Par contre, la modification du bit de poids faible est difficilement applicable sur des images de 8 bits (256 couleurs) en raison de la limitation du nombre de couleurs.

Il existe plusieurs logiciels qui utilisent cette méthode. On peut notamment citer InPlainView[10] et InThePicture[11].

4.2.1.2 ...dans un fichier audio

De la même manière, le LSB peut également être appliqué à un fichier audio. Le bit de poids fort est le plus perceptible pour l'oreille humaine à l'inverse du bit de poids faible. Modifier le bit de poids faible aura un impact limité sur la qualité globale du son. S-Tools et StegoWav[12] sont des exemples de logiciel appliquant cette méthode à un medium au format WAV (voir aussi section 2.2.1).

[10] Justin Weiler, InPlainView v1.0, http ://www.softpile.com/Utilities/Encryption/Review_05300_index.html, 1999

[11] Intar, InThePicture v2.2, http ://www.intar.com/ITP/itpinfo.htm

[12] Peter Heist, "StegoWav", 1996

4.2.1.3 Amélioration de la méthode LSB

Une approche plus sophistiquée de cette méthode réside dans l'utilisation d'un générateur de nombres pseudo-aléatoires pour étaler le message secret sur le medium de couverture.

Si les deux participants de la communication partage une stégo-clé k utilisable comme graine pour un générateur de nombres aléatoires, ils peuvent créer une séquence aléatoire. Ainsi, la distance entre deux bits encapsulés est déterminée aléatoirement. Puisque le récepteur a accès à la graine k et a connaissance du générateur de nombres pseudo-aléatoires, il peut reconstruire la séquence de départ (voir figure 4.4).

FIG. 4.4 – Méthode à intervalle aléatoire

```
for i = 1...,l(c) do
  s_i ← c_i
end for
generate random sequence k_i using seed k
n ← k_1
for i = 1,...,l(m) do
  s_n ← c_n ⇋ m_i
  n ← n + k_i
end for
```

(a) Encapsulation

```
generate random sequence k_i using seed k
n ← k_1
for i = 1,...,l(m) do
  m_i ← LSB(c_n)
  n ← n + k_i
end for
```

(b) Extraction

PGMStealth[13] est un exemple de logiciel du domaine public qui implémente cet algorithme.

[13] J. Poskanzer, PGMStealth, 1991

4.2.2 Dégradation d'image

Un cas particulier des systèmes de substitution est la technique de dégradation d'image ou "*image downgrading*". Cette technique, découverte en 1992 par Kurak et McHugh, était utilisée dans les systèmes de sécurité - notamment militaires - à plusieurs niveaux. Par exemple, lorsque l'on souhaite réduire la classification d'une information confidentielle (une image satellite classée top-secret qui passe au domaine public).

Cette technique met en jeu deux images : une image de couverture et une image secrète ayant les mêmes dimensions. Pour chaque pixel :

– Pour classifier l'information, on échange les 4 bits de poids faible du medium avec les 4 bits de poids fort significatifs de l'image secrète ;
– Pour déclassifier, on extrait les 4 bits de poids fort significatifs du stégo-medium pour obtenir l'image secrète.

La dégradation de l'image n'est pas - au plan visuel - significative dans la plupart des cas, 4 bits sont suffisants pour transmettre une image secrète correcte (voir figure 4.6 et 4.7).

FIG. 4.5 – Exemple de l'image downgrading

(a) Médium

(b) Image secrète

FIG. 4.6 – Exemple de l'image downgrading

(a) Stégo-image

(b) Image extraite de la stégo-image

Ce procédé a récemment été amélioré par K. Knox[14]. Il utilise la diffusion d'erreur

[14] *Reversible digital images*, d'Electronic Imaging99 (SPIE)

pour améliorer la qualité des deux images.

4.2.3 Régions de couverture et bit de parité[15]

En divisant le medium en plusieurs régions disjointes, on peut stocker un bit d'information dans chaque région de couverture plutôt que dans un seul élément. Le bit de parité d'une région I peut être calculé par :

$$p(I) = LSB(c_j) \, mod(2)$$

Appliquer cette méthode a une image consiste à définir dans un premier temps, des blocs de pixels puis, dans un deuxième temps, à modifier la valeur des pixels de ces blocs en fonction de l'information que l'on souhaite insérer. Pour un bloc donné, on augmente par exemple les valeurs des pixels lorsque l'on veut inscrire 1, et on les diminue pour inscrire 0.

Le choix de l'emplacement des blocs dans l'image peut être fixé ou bien paramétré par une clé secrète. L'introduction d'une telle clé est particulièrement importante si l'on utilise des marques qui sont des empreintes numériques. En effet, si les blocs sont pris aux même endroits pour tous les stégo-media, alors on peut, par la simple soustraction de deux versions marquées d'une même image, déterminer l'emplacement d'un certain nombre de ces blocs. Il suffirait alors de modifier légèrement la valeur des pixels dans ces blocs pour rendre le message caché illisible.

En revanche, on remarque que si la position des blocs ainsi que les valeurs binaires qui leur sont associées, forment un motif bien défini, alors la présence ou l'absence d'un message secret peut être détecté sans que l'on ait besoin du fichier original.

Cette technique permet d'incruster une "marque" qui résiste aux compressions avec perte ainsi qu'à la numérisation après impression. Elle ne permet cependant que l'inscription d'un minimum d'informations, de l'ordre d'une centaine de bits en moyenne (puisqu'on inscrit un bit unique par bloc).

4.2.4 Les palettes d'images

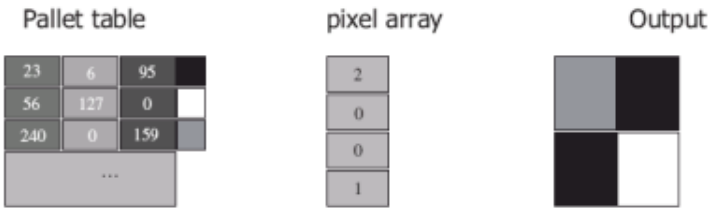
Une image basée sur une palette est une image colorisée avec un sous-ensemble de couleurs appartenant à un espace de couleurs spécifiques. Autrement dit, la palette est un tableau de 3 octets des valeurs RVB. Chaque pixel est représenté comme un index à cette table plutôt que la valeur des couleurs. Si peu de couleurs sont utilisées, cette approche réduit nettement la taille du fichier. Les deux formats les plus populaires sont le Graphics Interchange Format (GIF) et le format bitmap (BMP) (voir section 2.3.3).

[15] *Cover-regions and parity bit*, C. Dautzenberg, F.M. Boland, G. Caronni

Cependant leur utilisation s'efface au profit des formats utilisant des techniques de compression sophistiquées.

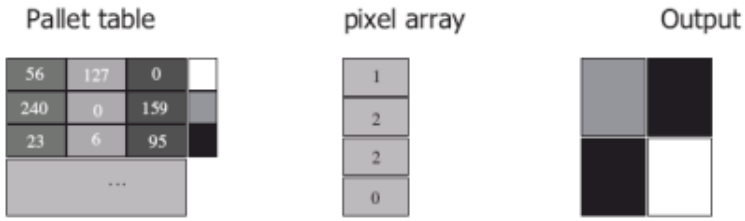
Il existe deux façons d'encoder de l'information dans une image basée sur une palette : on manipule les données ou la palette de couleurs. Le LSB des pixels peut encore être une fois utilisé comme nous l'avons présenté précédemment. En ce qui concerne les palettes, changer l'ordre de la palette ne modifie pas l'image comme le montre la figure 4.7 et 4.8.

FIG. 4.7 – Exemple de palette, pixel et de l'affichage correspondant



L'image résultante n'est pas modifiée car les valeurs des pixels ont aussi été modifiées pour compenser. Il existe $N!$ permutations de l'ordre de la palette, ce qui est suffisant pour coder un petit message. Cette méthode est très vulnérable puisqu'il suffit de modifier l'ordre de la palette pour détruire le message.

FIG. 4.8 – Même exemple avec la palette modifiée



Le logiciel EzStego[16] tri la palette de manière à regrouper les couleurs visuellement les plus proches. Les valeurs des couleurs peuvent par exemple, être stockées selon leur distance euclidienne dans l'espace RVB : $d = \sqrt{R^2 + V^2 + B^2}$.

Puisque l'oeil est plus sensible au changement dans la luminance d'une couleur, une autre approche serait de trier les entrées de la palette selon la composante de luminance.

[16]Machado, R., "EzStego, Stego Online, Stego," http ://www.stego.com, 1997.

Une autre méthode a été établit par les chercheurs coréens C.H. Tzeng, Z.F. Yang et W.H Tsai.

Cette méthode est basée sur le modèle des couleurs RVB. On définira la couleur d'un pixel en faisant la moyenne de ces 3 octets. Soit c_1 et c_2 deux couleurs avec respectivement les valeurs (r_1, v_1, b_1) et (r_2, v_2, b_2). Les relations entre c_1 et c_2 peuvent être définit de la façon suivante :

$$c_1 > c2, \text{ si } (r_1+v_1+b_1)/3 > (r_2, v_2, b_2)/3\,;$$

$$c_1 = c2, \text{ si } (r_1+v_1+b_1)/3 = (r_2, v_2, b_2)/3\,;$$

$$c_1 < c2, \text{ si } (r_1+v_1+b_1)/3 < (r_2, v_2, b_2)/3\,;$$

La méthode proposée consiste en un parcours de tous les pixels de l'image. Pour préserver la qualité de l'image, des pixels jugés capables de cacher une information sont sélectionnés durant le processus d'encodage. Un pixel x est jugé capable de cacher une information si le nombre de couleurs distinctes (α), parmi ces quatre voisins est plus grand que Tc, et si la différence maximum entre la couleur de x et celle de ces voisins (β) est inférieure à Td.

La figure suivante illustre un pixel x et ces quatre pixels voisins utilisés pour déterminer s'il est capable de cacher une information.

P1	**P2**	**P3**	
P4	x		

La définition de "l'habilitation" d'un pixel peut s'exprimer telle que :

X est capable de cacher une donnée si $\alpha > Tc$ et $\beta < Td$

Tc est utilisé pour vérifier si x est situé dans une région complexe où l'oeil humain n'est pas sensible à la modification d'un pixel. Si le nombre maximum de couleurs distinctes est plus grand que Tc, il se peut que ce soit un bon endroit pour cacher une donnée. De plus, si la différence maximum de couleur est supérieure à Td, il est probable qu'on se situe dans une région avec un grand contraste de couleur. Dans ce cas, il est préférable de laisser le pixel x intact. Td est utilisé pour déterminer si la couleur de x est très différente de celle de ces voisins.

Dès qu'un pixel capable de cacher des données est rencontré, une fonction f est définie dynamiquement par une clé privée K. Cette fonction est basée sur les relations entre x et la couleur de ses pixels voisins. La clé privée K est utilisée pour générer une

séquence binaire aléatoire et le résultat de f dépend de cette séquence. Pour simplifier, la fonction f prend comme paramètres l'indice du pixel x et les couleurs de ces 4 pixels voisins. Par exemple, le pixel x a des pixels voisins avec des couleurs distinctes (c_1, c_2, c_3, c_4). La fonction f peut être définie des façons suivantes :

$f(x, c_1, c_2, c_3, c_4) = 0$, si $c(x) \geq c_1$;

$f(x, C) = 1$, si $c_2 \leq c(x) \leq c_1$;

$f(x, C) = 0$, si $c_3 \leq c(x) \leq c_2$;

$f(x, C) = 1$, si $c_4 \leq c(x) \leq c_3$;

$f(x, C) = 1$, si ...

Si le résultat de la fonction est égal au bit b que l'on doit cacher, x demeure inchangé. Par contre s'ils ne correspondent pas, on change la couleur de celui-ci de manière à ce que la nouvelle couleur reste proche de celle initiale et que $f(x, c_1, c_2, c_3, c_4) = b$.

Pour une meilleure compréhension, il est proposé l'exemple suivant. On pose $Tc = 2$ et $Td = 10$. On a :

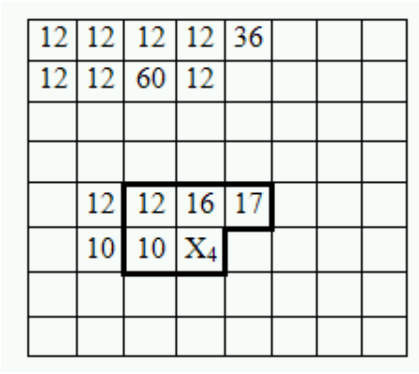
FIG. 4.9 – Etape 1

12	12	12	12	36			
12	X_1	60	X_2				
	12	12	16	17			
	10	X_3	X_4				

Etape 1

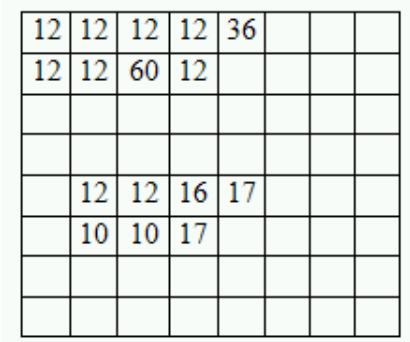
– X1 n'est pas idéal pour cacher des données : $\alpha = 1$ couleur (12) et α < Tc
– X2 n'est pas idéal pour cacher des données : $\beta = 60 - 12 = 48$ et β >Td
– X3 est idéal pour cacher des données : $\alpha = 3$ couleurs (10, 12, 16) et $\beta = 16 - 10 = 6$
  $f(X3, 10, 12, 16) = 0 = b$
  Le bit à cacher est 0. X3 restera inchangé.

FIG. 4.10 – Etape 2



Etape 2
- La couleur du pixel X3 est inchangée.
- X4 est idéal pour cacher des données : $\alpha = 4$ couleurs (10, 12, 16, 17) et $\beta = 17 - 10 = 7$
  $f(X3, 10, 12, 16, 17) = 0 \neq b = 1$
  Le bit à cacher est 1. X3 sera modifié.

FIG. 4.11 – Etape 3



Etape 3
- La couleur du pixel X4 est changée.
- $f(X4, 10, 12, 16, 17) = 1 = b$

Pour l'extraction du message caché, il suffit de parcourir l'ensemble de l'image à la recherche d'un pixel susceptible de loger des informations. Lorsqu'on rencontre un de ces pixels, on génére la fonction f grâce à la clé privée K. Si la fonction renvoie

un 0, le bit extrait est 0. De la même façon, si elle renvoie un 1, le bit extrait est 1. L'extraction est donc simple et rapide. Pour conclure sur cette technique, voici un tableau des performances avec différentes valeurs de Tc et Td.

Tc	Td	Maximum de bits utilisés
1	30	1150
2	20	960
3	10	105

4.2.5 Utilisation des espaces réservés ou inutilisés

Utiliser les espaces réservés ou inutilisés pour contenir de l'information cachée donne un moyen de dissimulation qui ne dégrade pas le medium.

4.2.5.1 Utilisation du système de fichier

Comment cacher de l'information à l'intérieur de l'espace libre d'un disque dur ou d'un CD-ROM ? En exploitant la manière dont les données sont stockées sur un support et gérées par le système d'exploitation ou le système de fichiers.

Les systèmes de fichiers stéganographiques

Un support physique de données est constitué de blocs qui contiennent les informations. Un ensemble de blocs, éventuellement non contigus, constitue un fichier. Pour répertorier tous les fichiers d'un disque, le système d'exploitation accède et met à jour une table d'allocation des fichiers. Seuls les fichiers dont l'adresse physique figure sur cette table sont accessibles par le système d'exploitation. Lorsqu'on veut effacer un fichier, le système de fichiers supprime l'adresse des blocs concernés de la table d'allocation (ce qui explique qu'on puisse retrouver des données contenues dans des fichiers effacés, puisque les blocs ne sont pas effectivement effacés). L'idée consiste, à l'aide d'un logiciel spécialisé, à inscrire directement des blocs sur le disque dur des blocs sans que la table d'allocation en ait connaissance. De cette manière, on insère un message secret sur le disque à l'insu du système d'exploitation.

Cette méthode souffre cependant d'une faiblesse : si l'intercepteur connaît l'existence d'un fichier caché, il n'aura aucune difficulté à le retrouver directement sur le disque. C'est pourquoi les données sont chiffrées et le support physique ne contient que des données illisibles pour celui qui ne possède pas la clé. On peut encore aller plus loin et tenter de dissimuler les informations chiffrées. A priori, cela ne pose pas de difficultés, puisque les blocs d'un système de fichiers sont par défaut remplis aléatoirement et que le chiffrement de données transforme celles-ci en données aléatoires. Ainsi en remplaçant certains blocs, désignés par une clé secrète, par ceux du fichier chiffré,

personne ne pourra s'apercevoir de l'insertion du message secret dans le système de fichiers.

L'inconvénient majeur est que la conservation des données n'est pas garantie. L'écriture d'un nouveau fichier sur le disque peut entraîner la perte des données enregistrées, puisque le système d'exploitation ne distingue pas les blocs utiles et les blocs réellement libres. Pour limiter au maximum le risque de suppression ou d'endommagement, la solution consiste à placer plusieurs copies des informations secrètes. Quand bien même quelques données seraient détruites, le secret demeure disponible tant qu'une seule copie est disponible. Evidemment, le risque de voir les informations perdus reste toujours présent. Comme le dit Andrew Mc Donald, l'auteur du programme libre StegFS[17] qui utilise ce principe, "C'est le prix à payer pour utiliser la stéganographie".

Espace intersticiel

Une amélioration de la méthode précédente est possible via l'utilisation de "slack space" ou "espace intersticiel". Les blocs constituant le système de fichier ont une taille fixe qui peut être modifiée dans un environnement UNIX. Sous Linux la valeur par défaut est de 4 ko. Sous Windows 95/98, la taille des blocs est de 32 ko. Chaque fichier créé se voit allouer un multiple entier de blocs pour satisfaire les exigences de stockage. Si la taille à stocker est inférieure au multiple entier de blocs, il va subsister un espace de stockage non utilisé et non utilisable. C'est le "**slack space**" ou "**espace intersticiel**". Un programme qui ouvre un fichier s'allouant la totalité de l'espace disponible sur un disque affichera probablement des informations censées être détruites.

Le noyau Linux implémente des fonctions non dépendantes du système de fichiers permettant d'accéder directement aux pages disques. Il existe cependant une interface utilisateur peu connue et mal documentée qui est utilisée dans des outils développés par Scyld Computing Corp., BMAP et SLACKER (se référer à l'article de Christophe Le Cannelier dans le numéro de Juillet/Août 2002 de Linux Magazine pour l'utilisation de ces logiciels).

4.2.5.2 Utilisation d'un fichier exécutable

Le fichier binaire exécutable résulte de la compilation d'un programme. Ce fichier, qui contient les instructions du programme compréhensibles par le système d'exploitation, est composé de différentes sections, chacune ayant sa propre utilité. Lorsque le programme est exécuté, le système d'exploitation lit toutes les informations dont il a besoin dans ces différentes sections. En particulier, l'une d'elles comporte les instructions en langage machine, dont certaines sont inutilisées.

Par exemple, on exécute le programme suivant :

```
/* hello.c*/
```

[17] StegFS `http://www.mcdonald.org.uk/StegFS/`

```
main(){
printf("Bonjour\n");
}
```

La version assembleur du programme, c'est-à-dire l'ensemble des instructions du programme écrit en langage humain et transformée pour le rendre compréhensible par le processeur va contenir les instructions suivantes :

```
[kalex]$  gcc o hello hello.c
[kalex]$  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
0x80483d5 <main+8> : call 0x8048308 <printf>
0x80483d8 <main+16> : add $0x4,%esp
0x80483d9 <main+17> : leave
0x80483da <main+18> : ret
0x80483db <main+19> : nop
0x80483dc <main+20> : nop
0x80483dd <main+21> : nop
0x80483de <main+22> : nop
0x80483df <main+23> : nop
End       of assembleur dump.
```

On constate que la version assembleur contient des instructions inutiles qui ne font aucune opération (nop, pour no opération). Dans cette portion de "code mort", on peut remplacer ces octets par ceux de notre choix, si on a l'assurance que cette zone ne sera jamais exploitée par le programme.

En effet, les octets qui correspondent au message et qui sont placés dans cette zone ne correspondent quasiment jamais à des instructions valides, de sorte que si le programme atteint cette zone, il a de grandes chances de "planter"[18].

[18]Cette méthode est aussi connu sous le nom de heap overrun ou overflow. Elle est régulièrement utilisée dans le monde du piratage pour rendre un logiciel vulnérable à une attaque.

Une autre solution proposée en 1998 par Christian Colberg, de l'Université d'Arizona, consiste à transformer un programme *P* en un autre *P'* qui adopte le même comportement. Dans le nouveau programme, le dissimulateur introduit de nouvelles instructions sous la forme de branchement conditionnel dont les deux parties sont constituées de codes équivalents. Ainsi un utilisateur du programme ne constatera pas de différences de comportement. Ici le message caché est constitué de la suite d'instructions nouvelles introduites : une étude des lignes de codes du programme ne permettra que le constat d'un ensemble de branchements conditionnels qui forment un arbre. Cet arbre dissimule l'information, à la manière des "*grammaires algébriques*" utilisées pour cacher du texte (voir section 4.7.2).

4.2.5.3 Exploitation d'autres zones non utilisées

Les en-têtes des fichiers images ou audio peuvent aussi contenir des informations "supplémentaires". Certains formats d'image peuvent même contenir des commentaires qui ne sont pas affichés par les visualiseurs d'image (PNG, JPEG). Invisible Secrets 2002[19] repose sur cette technique, si le medium est au format PNG ou JPEG.

Les protocoles du modèle réseau OSI ont des caractéristiques qui permettent de cacher de l'information. Les paquets TCP/IP utilisés pour transporter de l'information à travers Internet contiennent de l'espace inutilisé dans leurs en-têtes. L'en-tête d'un paquet TCP dispose de 6 bits réservés (généralement inutilisés) et l'en-tête d'un paquet IP en possède deux. Des millions de paquets sont transmis dans chaque canal de communication, ce qui peut fournit un excellent canal de communication caché.

Craig H. Rowland[20] explique plusieurs variantes pour dissimuler de l'information et a développé un programme en C pour les exploiter. L'exploitation repose sur le fait que l'on encode en ASCII. On peut ainsi passer des données dans des paquets qui apparaissent comme étant des requêtes de connexion, des demandes d'établissement de transfert de données, ou des étapes intermédiaires. Ces requêtes peuvent aussi contenir des fausses adresses IP tout comme des faux ports de destinations et sources.

Par exemple, en manipulant le champ d'identification IP nous allons transmettre le mot "HELLO" :

Packet One :

18 :50 :13.551117 nemesis.psionic.com.7180 > blast.psionic.com.www : S 537657344 :537657344(0) win 512 (ttl 64, id 18432)

Decoding :...(ttl 64, id 18432/256) [ASCII : 72(H)]

Packet Two :

[19] NeoByte Solution, Invisible Secrets 2002, http ://www.invisiblesecrets.com.

[20] *Covet channels in the TCP/IP Protocol*, Craig H. Rowland

18 :50 :14.551117 nemesis.psionic.com.51727 > blast.psionic.com.www : S1393295360 :1393295360(0) win 512 (ttl 64, id 17664)

Decoding :...(ttl 64, id 17664/256) [ASCII : 69(E)]

Packet Three :

18 :50 :15.551117 nemesis.psionic.com.9473 > blast.psionic.com.www : S 3994419200 :3994419200(0) win 512 (ttl 64, id 19456)

Decoding :...(ttl 64, id 19456/256) [ASCII : 76(L)]

Packet Four :

18 :50 :16.551117 nemesis.psionic.com.56855 > blast.psionic.com.www : S3676635136 :3676635136(0) win 512 (ttl 64, id 19456)

Decoding :...(ttl 64, id 19456/256) [ASCII : 76(L)]

Packet Five :

18 :50 :17.551117 nemesis.psionic.com.1280 > blast.psionic.com.www : S 774242304 :774242304(0) win 512 (ttl 64, id 20224)

Decoding :...(ttl 64, id 20224/256) [ASCII : 79(O)]

Packet Six :

18 :50 :18.551117 nemesis.psionic.com.21004 > blast.psionic.com.www : S3843751936 :3843751936(0) win 512 (ttl 64, id 2560)

Decoding :...(ttl 64, id 2560/256) [ASCII : 10(Carriage Return)]

4.3 Techniques dans le domaine transformé

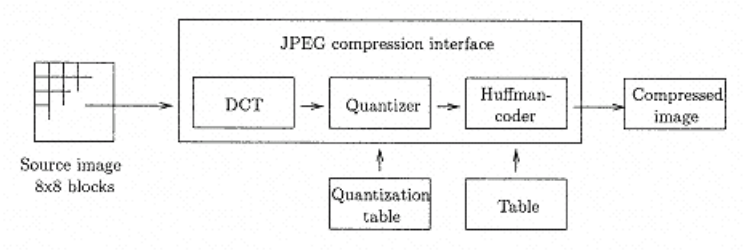
Nous avons vu que les techniques de substitution LSB sont des méthodes très simples à mettre en oeuvre, mais elles sont très vulnérables même aux modifications les plus minimes du medium de couverture. Un attaquant n'aura qu'à appliquer un traitement d'image quelconque pour détruire complètement l'information secrète contenu dans une image. Il a été noté très tôt qu'encapsuler l'information dans le domaine des fréquences plutôt que dans le temps était nettement plus robuste. Aujourd'hui les systèmes stéganographiques les plus robustes utilisent le domaine transformé (voir algorithme de F5 et Outguess en annexe).

En utilisant cette technique, nous allons pouvoir dissimuler l'information dans les zones significatives du medium.

4.3.1 Stéganographie dans le domaine de la Transformée en Cosinus Discret (TCD)

Avant de décrire les méthodes utilisées dans cette section, il est nécessaire d'avoir des notions sur la transformée de Fourier et la compression JPEG (voir section 2.3.4).

FIG. 4.12 – Algorithme de compression avec perte JPEG

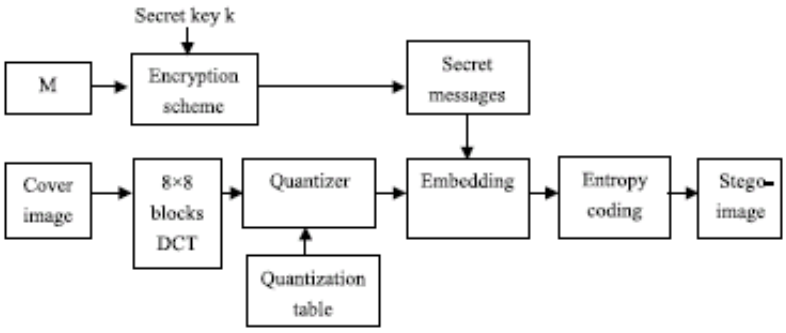


4.3.1.1 Modification des coefficients TCD

Les images JPEG utilisent la Transformée en Cosinus Discret (TCD) pour accomplir la compression. Les données compressées sont stockées en tant qu'entier ; cependant, les calculs pour le traitement de la quantification exigent des calculs en virgule flottante ce qui donne des arrondis. L'erreur introduite par l'arrondi définit le caractère de compression avec perte du format JPEG. Jpeg-Jsteg est un outil de stéganographie utilisant cette technique (voir algorithme en annexe). L'information est cachée dans l'image JPEG en modulant les choix d'arrondi dans les coefficients TCD.

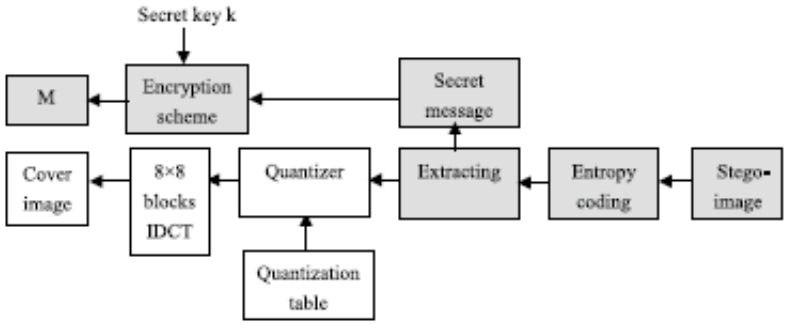
A l'encapsulation on suit le schéma de la figure 4.13.

FIG. 4.13 – Insertion d'information avec une image au format JPEG



A l'extraction du message on suit le schéma de la figure 4.14.

FIG. 4.14 – Extraction de l'information



4.3.1.2 Amélioration

E. Koch et J. Zhao proposent l'utilisation de la transformée en cosinus discret dans le schéma suivant :

1. Sélection et extraction de blocs de 8 x 8 pixels (les emplacements dépendent d'une clé secrète) ;
2. Pour chaque bloc :
   (a) modification de tous les pixels en fonction du bit à inscrire : leur valeur est augmentée ou diminuée selon que le bit vaut 0 ou 1 ;
   (b) normalisation des valeurs des pixels, de sorte qu'elles soient toute comprises dans un intervalle fixé (par exemple [-127..127]) ;
   (c) application de la transformée en cosinus discret ;
   (d) choix d'un certain nombre de coefficients ;
   (e) modification de ces coefficients de telle sorte qu'ils vérifient une certaine relation d'ordre ;
   (f) application de la transformée en cosinus discrète inverse ;
3. Réintégration des blocs dans l'image.

La clé secrète réside dans l'emplacement des blocs, mais aussi dans le choix des coefficients que l'on modifie. Cette technique permet d'inscrire environ dix fois plus d'informations que la technique de régions de couverture et bit de parité.

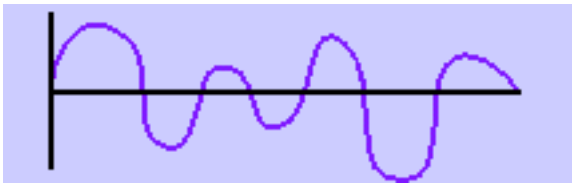
4.3.2 Dissimulation dans l'audio : Codage de phase

La phase est un décalage temporel entre 2 signaux périodiques. La phase absolue est le respect de la position temporelle de toutes les parties d'un son depuis son enregistrement jusqu'a sa reproduction.

Le codage de phase consiste à remplacer la phase d'un signal par une nouvelle phase qui code les données à cacher. Les phases relatives entre les différents segments du signal sont ensuite ajustées. En effet, c'est aux écarts de phase relative que l'oreille humaine est sensible.

On possède le signal audio de la figure 4.15.

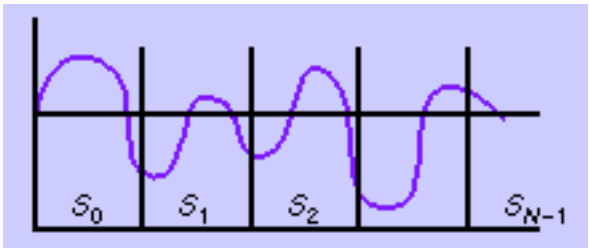
FIG. 4.15 – Signal audio initial



La procédure pour le codage de phase est la suivante :

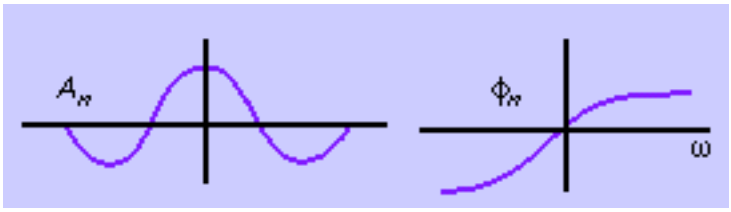
1. La séquence de son original est découpée en une série de N segments courts (figure 4.16) ;

FIG. 4.16 – Découpage



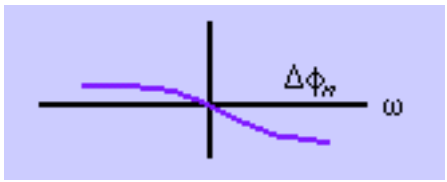
2. Une transformée de Fourier est appliquée sur chaque segment afin de créer une matrice de phase ϕ_n et d'amplitude A_n du signal (figure 4.17) ;

FIG. 4.17 – Transformée de Fourier



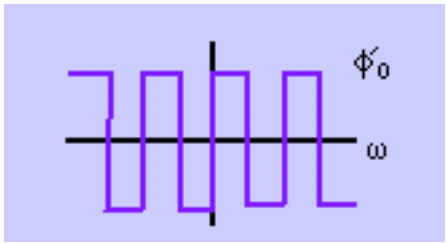
3. La différence de phase entre chaque segment adjacent est calculée (figure 4.18) ;

FIG. 4.18 – Différence de phase



4. Pour le premier segment, un segment artificiel est créé (figure 4.19).

FIG. 4.19 – Création d'un segment artificiel

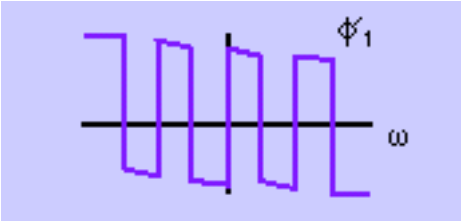


$$\Delta(\phi_0(k)) = \frac{\pi}{2} \text{ si } m_k = 0$$

$$\Delta(\phi_0(k)) = -\frac{\pi}{2} \text{ si } m_k = 0$$

Pour tous les autres segments, une nouvelle matrice de phase est créée en utilisant les différences de phases (figure 4.20).

FIG. 4.20 – Création des autres segments



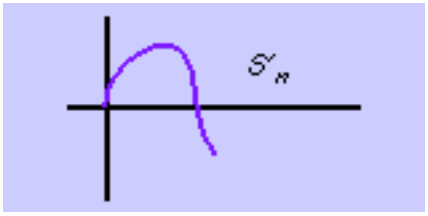
$$\Delta(\phi_1(k)) = \Delta(\phi_0(k)) + [\phi_1(k) - \phi_0(k)]$$

...

$$\Delta(\phi_N(k)) = \Delta(\phi_{N-1}(k)) + [\phi_N(k) - \phi_{N-1}(k)]$$

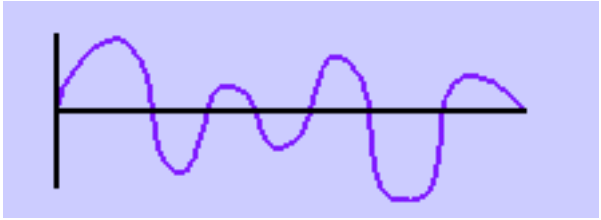
1. La nouvelle phase et l'amplitude originale sont combinées pour donner un nouveau segment Sn' (figure 4.21) ;

FIG. 4.21 – Sn'



2. Finalement, les nouveaux segments sont concaténés pour créer un résultat codé (figure 4.22).

FIG. 4.22 – Signal final



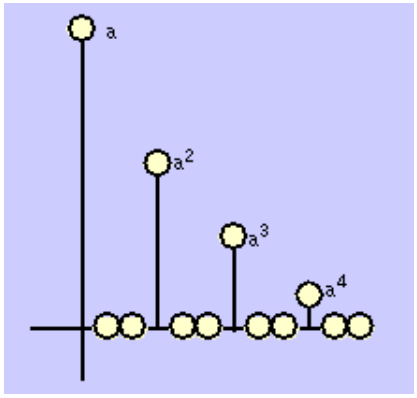
Depuis que la phase du premier segment a été modifiée, les phases absolues de tous les segments suivants sont changées. Mais leurs différences relatives sont préservées. Pour décoder le message, la longueur du segment, les coefficients de la transformée en cosinus discret et les intervalles de données doivent être connus. La valeur de la phase sous-jacente du premier segment est détectée comme 0 ou 1, qui représente le message binaire codé.

4.3.3 Ajout d'écho

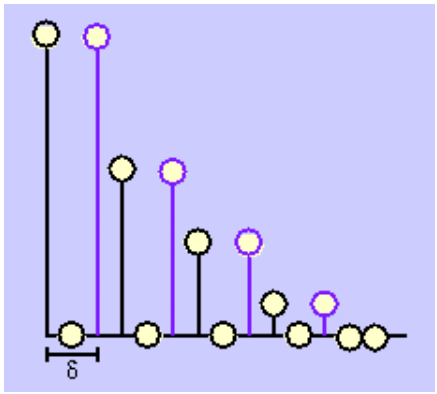
Cette technique est robuste et permet de cacher beaucoup de données. L'ajout d'un écho pour encoder des bits se fait en deux temps. Les bits codés sont très difficile à percevoir à l'oreille, mais on peut les distinguer comme des éléments qui enrichissent le son original. Cette méthode est la seule qui résiste à la compression. En ajoutant de l'écho, les données sont cachées en variant l'amplitude initiale du signal, en délabrant les fréquences et les offsets. Quand le délai entre le son original et l'écho diminue les signaux se mélangent et l'oreille humaine ne peut distinguer les deux.

L'information cachée est insérée en résonnant le son original. Un "1" binaire est résonné d'au plus une seconde et un "0" binaire d'au plus zéro seconde (voir figure 4.23).

FIG. 4.23 – Ajout d'écho



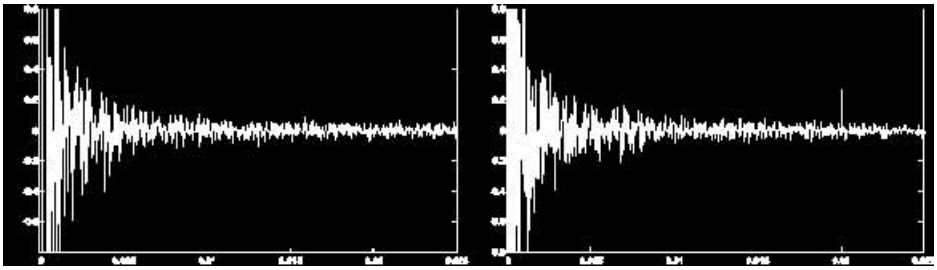
(a) Signal initial



(b) Signal après l'ajout d'écho

L'extraction de l'information cachée se fait en détectant les espaces entre les échos (figure 4.24).

FIG. 4.24 – Extraction de l'information



Ce procédé modifie très peu le son original. Si l'on parvient à percevoir un changement, on ne distinguera qu'un signal plus riche en son.

4.4 Techniques à étalement de spectre

Nous avons vu que l'objectif de la stéganographie est de transmettre une information à travers un support, bien que cette image subisse des transformations. Il est alors naturel d'utiliser des techniques qui ont fait leurs preuves dans les télécommunications. L'exemple que nous allons examiner est celui de l'étalement de spectre.

4.4.1 En télécommunication

En télécommunication, l'étalement de spectre était utilisé par les militaires pendant la seconde guerre mondiale. Le concept est relativement simple : multiplier les données binaires par un code[21] lors de l'émission puis à la réception pour retrouver le signal original transmis. Cette multiplication ayant pour effet d'étaler le spectre du signal au dessous du niveau du bruit, permettant ainsi de cacher l'existence de la communication (voir [18] pour plus de détails).

4.4.2 Modèle de l'étalement de spectre

L'idée est de plaquer sur l'image un signal bidimensionnel ressemblant à une "tôle ondulée". La transformée de Fourier de ce signal a la forme d'un pic centré autour de la fréquence d'ondulation et décroît fortement autour de ce pic. La clé est la structure de la "tôle", c'est à dire sa fréquence spatiale et la direction d'ondulation. Pour extraire le signal, on prend la transformée de Fourier de celle-ci et on extrait, grâce à la clé, le signal caché. Un exemple d'utilisation est donné à la figure 4.25.

[21] Code pouvant être une séquence pseudo-aléatoire (code Gold, séquence Kasami) ou code ayant des propriétés d'orthogonalité (fonction de Walsh, code de Gold orthogonaux)

Le signal à cacher est une grille-image constituée de carrés de pixels noirs ou blancs (a). On place celle-ci sur une porteuse de fréquence et de direction données par une clé secrète (b).

Cette marque est surperposée à l'image (c). En fait les pixels noirs correspondent à une augmentation de la luminance de l'image. De même les pixels blancs correspondent à une diminution. Lors de cette étape la grille est modifiée afin de préserver l'invisibilité du message, surtout dans les parties homogènes de l'image.

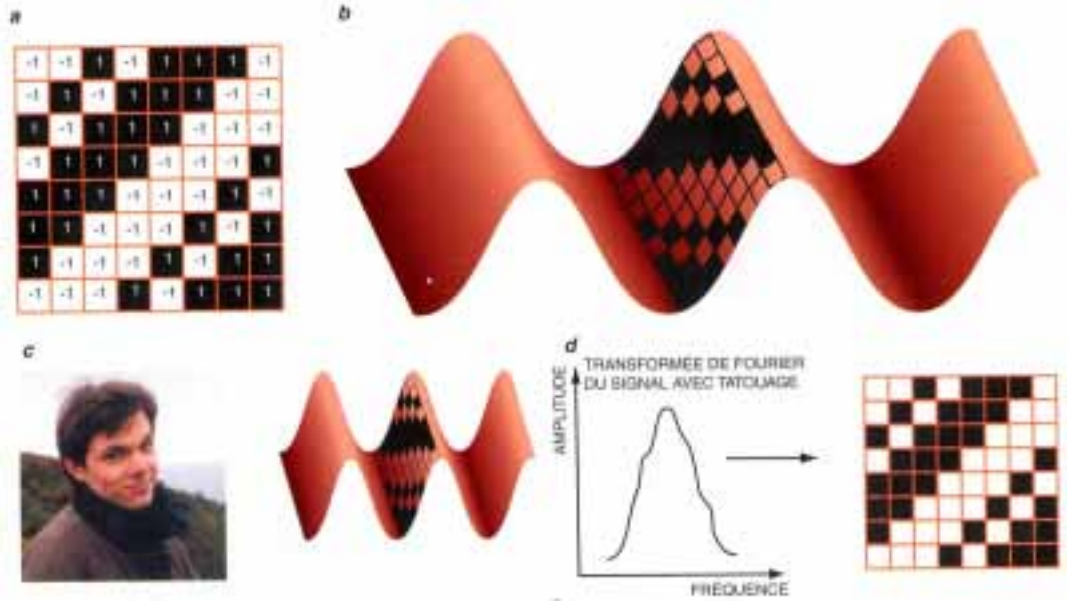
Pour récupérer le message, on extrait de la Transformée de Fourier de l'image le signal caché, localisé autour de la fréquence et de la direction données par la clef secrète. On prend la transformée de Fourier inverse et on obtient la grille de départ (d).

4.5 Les méthodes statistiques

Les techniques de stéganographie statistique utilisent l'existence du "bit 1". On change le medium de façon à ce que quelques caractéristiques statistiques soient modifiées si un "1" est transmis. Autrement le medium demeure inchangé. Ainsi, le récepteur sera capable de distinguer un medium non modifié d'un autre.

Pour pouvoir appliquer ce procédé à plusieurs bits il faut diviser le medium en zones. Un bit secret est inséré dans le ième bloc en plaçant un "1" si l'on a modifié le bloc

FIG. 4.25 – Modèle de l'étalement de spectre



dans le processus d'encapsulation. Sinon le bloc reste inchangé. La détection d'un bit spécifique se fait via une fonction de test qui discrimine les blocs modifiés des autres :

– $f(B_i) = 1$, le block B_i a été modifié pendant le processus d'encodage.
– $f(B_i) = 0$, le block B_i n'a pas été modifié pendant le processus d'encodage.

Le récepteur applique la fonction f sur tous les blocks B_i dans l'ordre pour trouver tous les bits secrets du message . En pratique, les méthodes statistiques sont difficiles à appliquer et de nombreuses hypothèses doivent être faites.

L'approche proposée par Bender, Gruhl et Morimoto [19] repose sur une étude statistique des points de l'image. Cette méthode repose sur la différence de luminance observée entre deux ensembles de pixels. Elle est fondée sur l'analyse statistique suivante : si l'on considère deux grands ensembles A et B de pixels répartis dans une image, alors la moyenne des différences de luminance entre les pixels de A et B est très proche de 0. On choisit des ensembles A et B à l'aide d'une clé secrète, puis on modifie leur luminance : on augmente celle des pixels de A d'une quantité c, et l'on diminue celle des pixels de B de la même quantité c.

Cette technique est facile à mettre en oeuvre et offre l'avantage d'une vérification sans l'image originale. Le message incrusté de cette manière résiste bien aux modifications des couleurs (passage de la couleur au gris, modification Gamma), mais ne résiste pas aux rotations et changements d'échelle ni à la compression avec perte.

4.6 Techniques de distorsion

Contrairement aux systèmes de substitution, les techniques de distorsion requiert la connaissance du medium original lors du décodage. Le principe général est le suivant :

– Alice applique une séquence de modification dans le but d'avoir un stégo-objet. Elle choisit ses modifications en fonction du message secret qu'elle veut transmettre.
– Bob mesure la différence par rapport au medium original pour reconstruire la série de modification qui correspond au message secret.

Pour utiliser cette méthode il faut envisager l'hypothèse que l'image originale a été transmise de manière sécurisée. La plupart des méthodes utilisant du texte comme medium utilise des techniques de distorsion. Cela peut être effectuer en faisant varier les positions des mots et des lignes ou en ajoutant des caractères invisibles ou des espaces.

4.6.1 Encodage dans du texte formaté

La dissimulation de données dans du texte est une chose particulière et comme nous allons le voir, elle est différente de l'image ou du son. Ceci résulte de plusieurs raisons :

on ne travaille pas avec le texte dans “l’a peu près”. On peut endommager légèrement l’image pour cacher de l’information. Tandis qu’avec du texte, l’original est modifié ou non. Nous allons donc voir comment pallier à ces contraintes.

4.6.1.1 Méthode des espaces en fin de ligne

Il s’agit d’une méthode très simple qui consiste à placer des espaces en fin de ligne. On définit un algorithme :

– 0 espace en fin de ligne correspond à 0 ;
– 1 espace en fin de ligne correspond à 1.

Nous disposons ainsi d’une base pour encoder un message.

EXEMPLE : (les espaces sont remplacés par des "_" pour plus de lisibilité).

Voici un message caché.
A vous de le lire._
*Je pense que vous commencez_*
à comprendre le principe.
*Cette méthode n’est pas_*
idéale.
*Nous parvenons quand même à_*
dissimuler un octet._

Comme on peut le constater dans cet exemple, nous avons encodé :

0 espace ; 1 espace ; 1 espace ; 0 espace ; 1 espace 0 espace ; 1 espace ; 1 espace.

Soit : **01101011**

Un octet a donc été dissimulé. Cette méthode est très simple à implémenter mais elle est loin d’être efficace. Il faut énormément de lignes pour coder peu de texte (8 lignes pour coder 1 octet). Pour coder une phrase de 20 mots (chaque mot faisant environ 4 caractères) et si l’on code chaque caractères sur 7 bits, il faudra environ 560 lignes.

On peut améliorer cette méthode en utilisant 3 espaces :

– 0 espace $\Leftrightarrow$ 00
– 1 espace $\Leftrightarrow$ 01
– 2 espaces $\Leftrightarrow$ 10
– 3 espaces $\Leftrightarrow$ 11

Il faut alors 4 lignes pour coder 1 octet (au lieu de 8).

4.6.1.2 Méthode des espaces entre les mots

Cette méthode est basée sur le même principe, mais cette fois-ci nous allons encoder le texte dans un nombre d’espaces entre chaque mot. Cette méthode est encore moins discrète, mais le rapport texte encodé sur texte hôte est plus important.

Dans un premier temps, on établit une convention :

– Un espace entre 2 mots suivit de deux espaces entre les 2 mots suivants $\Leftrightarrow 0$;
– Deux espaces entre 2 mots suivit d'un espace entre les 2 mots suivants $\Leftrightarrow 1$.

Pour mieux comprendre voici un exemple avec le texte suivant :

Ceci_est__essai__de_texte__caché_dans__un_texte_hôte.

Cette__méthode__peut_se_réveler_efficace__contre_un_analyseur_de__mot.

Soit : **01110110**

Un octet a été encodé dans cette phrase. En observant le texte tel qu'il va réellement apparaître, nous constatons que cette méthode est indiscrète. A propos du rapport texte à encoder sur texte hôte, il faut 2 mots pour un bit, donc pour une phrase de 20 mots (20*7=240 bits), il faut un texte hôte de 480 mots, en comptant 10 mots par ligne on arrive à environ 50 lignes. La différence donne un différentiel de rapport 10 en comparaison avec la méthode décrite plus haut. Snow[22] est un exemple de logiciel qui implémente cette technique (voir son utilisation en annexe).

4.6.1.3 Variantes

Si le message est transmis dans un format de fichier tel que HTML, LATEX ou PostScript, il devient alors possible de cacher le message. Il est possible d'enregistrer l'information secrète dans la taille de l'interligne. Si l'espace entre deux lignes est plus petit qu'un certain seuil, un 0 est encodé, sinon c'est un 1.

Les fichiers HTML sont de bons prétendants pour inclure des "*extra espaces*", tabulations, et retour à la ligne. Deogol[23] est un programme Perl implémentant la méthode d'encodage sur des fichiers HTML. Il permute les paramètres des tags. Par exemple :

Soit un tag HTML de la forme : `<tagname attribute1=value1 attribute2=value2 ... >`

On peut aussi coder en HTML : `<IMG SRC="picture.jpg" ALT="A picture">`

ou `<IMG ALT="A picture" SRC="picture.jpg">`

Ce qui permet de coder 1 bit si l'on définit l'un comme étant le bit 1 et l'autre comme étant le bit 0.

Donc sur la ligne de code :

`<TD NOWRAP ROWSPAN=1 COLSPAN=4 ALIGN=left VALIGN=top HEIGHT=40 WIDTH=40 id=col1>`

Nous aurons 8 ! possibilités de permuter les paramètres pour obtenir un tag équivalent. Soit $\log_2(8!) \simeq 15,3$ bits, soit presque 2 octets.

[22] M. Kwan, Snow Version 1.1, http ://www.darkside.com.au/snow, 1996
[23] http ://forrest.cx/code/deogol/

De nombreuses autres méthodes peuvent être utilisées. Par exemple, en encodant l'information afin que le système de traitement choisisse les retours à la ligne. Prenons le format TEX, il utilise un algorithme sophistiqué pour calculer les lignes et les sauts de pages ; en fait, il classe les retours à la ligne possibles en trois valeurs appelées : *mauvaise qualité, pénalité* et *démérite*. Simplement, la mauvaise qualité d'une ligne est la mesure du nombre d'espace TEX que l'on devrait insérer pour avoir un "style groupé". La pénalité est assignée à chaque retour à la ligne possible, en représentant le "coût esthétique" en brisant à un endroit spécifique. Le paramètre démérite d'une ligne est calculé en fonction de la mauvaise qualité et de la pénalité. TEX tente de choisir une séquence de retour à la ligne de façon à ce que la somme des démérites dans un paragraphe soit minimale. En ajustant des paramètres internes, il est possible de choisir des retours à la ligne sous-optimaux et stocker des informations supplémentaires.

4.6.1.4 Méthode des synonymes

La méthode des synonymes consiste à construire une table de correspondance entre des bits et des mots.

BIT 0	BIT 1
gros	obèse
petit	minuscule
riche	fortuné
beau	joli
nourriture	aliment
bateau	navire
femme	épouse
tranquille	calme

EXEMPLE :

Au loin je vois mon gros navire sur la mer tranquille. Oui je suis fortuné, j'ai une belle épouse et je mange de très bons aliments, et ma voiture est loin d'être minuscule.

Ce qui donne le message secret : **01010111**

Toutefois la sémantique du texte ainsi construit risque de ne pas leurrer un être humain.

4.6.2 Distorsion d'image numérique

Les techniques de distorsion peuvent facilement être appliquées aux images numériques. En utilisant une technique similaire aux mécanismes de substitution :

– L'émetteur choisit d'abord les pixels de couverture qu'il veut utiliser. On effectue cette sélection via l'utilisation de générateur de nombres pseudo-aléatoires ou de permutations pseudo-aléatoires ;

– Pour encoder un 0 dans un pixel, l'émetteur laisse le pixel inchangé ; pour encoder un 1, il ajoute une valeur aléatoire à la couleur du pixel. Cette valeur peut être choisie de manière à ne pas changer les propriétés statistiques du medium. La différence entre cette méthode et la méthode LSB est que ces valeurs ne sont pas nécessairement égales aux bits du message secret ;
– Le récepteur compare tous les pixels sélectionnés du stégo-objet avec les pixels correspondants de l'image originale. Si le pixel diffère, c'est un 1, sinon c'est un 0.

Une autre technique de distorsion dans une image a été introduite par Standford. Elle consiste à modifier l'ordre d'apparence des données redondantes dans le medium au lieu de modifier les valeurs proprement dites. Le procédé d'encapsulation maintient une liste de paires d'exemples dont la différence est inférieure à un seuil défini préalablement.

4.7 Méthode de génération de couverture

A l'opposé des méthodes présentées où l'information secrète est ajoutée à un medium spécifique grâce à un algorithme d'encapsulation, certaines applications stéganographiques génèrent un objet numérique qui devient medium de couverture.

4.7.1 Les fonctions mimétiques

L'explosion des échanges d'informations autorise l'émission de l'hypothèse qu'il est impossible pour un être humain sans moyens spécifiques d'analyser toutes les communications. Cela ne devient possible qu'à l'aide d'automates qui examinent et analysent les mots clés et le profil statistique d'un message. Il est pour l'instant possible de distinguer un message chiffré d'un message non chiffré au regard de ses propriétés statistiques. Les fonctions mimétiques proposées par Wayner [17] peuvent être utilisées pour cacher l'identité d'un message en changeant son profil statistique de manière à le faire passer pour un texte innocent.

La langue française possède des propriétés statistiques comme par exemple la fréquence de distribution de certain mots de deux ou trois lettres. Ce procédé a déjà été utilisé à de nombreuses reprises dans les techniques de compression (codage Huffman[24]). A l'aide d'un alphabet donnée et d'une technique s'appuyant sur la probabilité de distribution, on code une chaîne de mots à l'aide de la méthode Huffman. Si maintenant, on procède à l'inverse de cette méthode pour transformer un message qui a un profil statistique A en celui du profil statistique B, le profil statistique de B est "mimé".

[24]voir cours de télécommunication, Mr Girault, IUT GTR Blois

On peut donc utiliser le codage Huffman pour compresser un fichier avec une distribution A et obtenir un stégo-objet en appliquant le codage d'Huffman inverse au fichier précédemment créé (avec une distribution B). La fonction mimétique est :

$$g(x) = f_B^{-1}(f_A(x)).$$

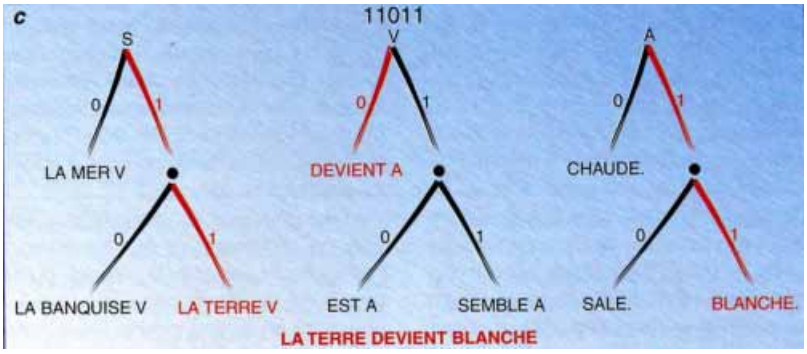
Cette méthode ne peut être utilisée que contre les détections automatiques. En effet, la sémantique est ignorée ce qui rend le message incompréhensible.

4.7.2 Les techniques basées sur la grammaire

Wayner utilise aussi les tables de Trithémius. Le texte de couverture est créé à partir de règle de grammaire et la (production) mise en scène en fonction du message secret. Ainsi le message secret n'est pas encapsulé, c'est le medium de couverture qui est le message caché. Si la grammaire est sans équivoque le récepteur peut extraire le message en utilisant un analyseur syntaxique.

La méthode fondée sur des "*grammaires algébriques*" consiste à cacher le message en suivant des règles de remplacement qui suivent des caractéristiques sémantiques, ici le sujet, verbe et attribut.

FIG. 4.26 – Exemple de grammaire algébrique



Dans cet exemple, on veut cacher le message binaire 11011 en partant du sujet S. On part de l'arbre qui indique les remplacements du sujet (à gauche) et on suit les branches. Le début du message 11 est remplacé par "La terre" suivi d'un verbe (V). On suit alors l'arbre de remplacement d'un verbe, ce qui aboutit à "devient" suivi d'un attribut (A) et, au final, on obtient "La terre devient blanche", phrase bien construite qui pourra abuser un lecteur.

Cependant en pratique on s'aperçoit très vite que le résultat final est susceptible d'attirer l'attention.

4.8 Classification des logiciels stéganographiques

Différentes techniques permettant d'utiliser de la stéganographie dans un medium numérique viennent d'être étudiées. Cette section décrit les critères d'évaluation d'un logiciel en fonction du type de méthode utilisée. En effet, de nombreux logiciels ne sont pas considérés comme sérieux. Les critères suivants ne prennent pas en compte l'option de chiffrement bien que généralement un logiciel proposant un algorithme de chiffrement solide donnerait plus de confiance. Notre classification s'échelonne de 1 à 5 :

1. Ajout de données à la fin du medium. (exemple : Camouflage, JpegX, SecurEngine for JPG, Safe&Quick Hide Files 2002, Steganography 1.50) ;
2. Insertion de données dans les champs de commentaire ou les en-têtes des structures de fichiers (exemple : Invisible Secrets 2002 for JPG and PNG, Steganozorus for JPG) ;
3. Encapsulation de données dans le medium de manière fixe, sequentielle et linéaire (exemple : InPlainView, InThePicture, Invisible Secrets 2002 for BMP, ImageHide, JSteg) ;
4. Encapsulation de données dans le flux de données du medium de manière pseudo aléatoire en fonction d'un mot de passe (CryptArkan, BMPSecrets, Steganos for BMP, TheThirdEye, JPHide) ;
5. Encapsulation de données dans le medium de manière pseudo aléatoire en fonction d'un mot de passe. Les propriétés statistiques du medium sont conservé par la modification des autres bits du medium. (exemple : Outguess, F5).

Les méthodes 1 et 2 sont fragiles et aisément détectables. Même les programmes les plus sérieux, Outguess et F5, tous les deux créés par des chercheurs du monde académique au code source libre ouvert à un examen minutieux de développeurs, sont désormais considérés comme "cassés" [10] grâce notamment, à l'équipe de Jessica Fridrich, Andreas Westtfeld. Rappelons que "casser" un logiciel de stéganographie ne signifie pas que l'on puisse retrouver les données, mais affirmer qu'un medium contient de l'information encapsulée et dont la taille peut être estimée.

Chapitre 5

Les limites

Dans la plupart des cas, les logiciels stéganographiques insèrent de l'information et manipulent les media de façon à ce qu'elle demeure invisible pour un être humain. Cependant toute modification introduit une distorsion ou une dégradation des propriétés du support de couverture par rapport au medium original. Les outils varient dans leur approche pour cacher de l'information. Sans connaître l'outil utilisé, détecter l'information peut devenir complexe. Cependant les logiciels introduisent des caractéristiques particulières qui peuvent agir comme des signatures pour les méthodes ou les outils utilisés.

5.1 Stéganalyse

La stéganalyse est une technique qui consiste à attaquer les méthodes stéganographiques par détection, destruction, extraction ou modification des données encapsulées. Le succès d'une stéganalyse dépend de l'objectif :

- Attaque **passive**, la détection suffit. Par exemple, la personne en charge de surveiller les données des employés ;
- Attaque **active**, on veut détecter mais aussi supprimer l'information encapsulée. Par exemple, le voleur de données souhaitant réutiliser une image qui a fait l'objet d'un tatouage numérique.

Plusieurs types d'attaques sont disponibles :

- Attaque sur le stégo-objet uniquement ;
- Attaque lorsque le medium original et le stégo-objet sont disponibles ;
- Attaque lorsque l'on connaît le message caché ; cela peut être utile pour l'analyse de motifs qui peuvent resservir lors d'une attaque future sur un système identique ;
- Attaque lorsque l'algorithme et le stégo-objet sont connus ;
- Attaque avec un message caché particulier ;

– Attaque lorsque l'algorithme est connu, le medium original et le stégo-objet sont disponibles.

5.2 Détecter l'information cachée

Les logiciels de stéganographie introduisent des caractéristiques particulières qui peuvent agir comme des signatures pour les méthodes ou les outils utilisés.

5.2.1 L'utilisation de chiffrement

En pratique, la stéganographie est couplée avec la cryptographie afin de rendre le message inintelligible s'il est découvert. Bien que la confidentialité soit assurée, cela pose des problèmes inhérents à l'utilisation du chiffrement.

5.2.1.1 Problème lié à la stéganographie à clé

La cryptographie est généralement couplée à un système stéganographique. Le principal problème est que lorsque l'on chiffre un fichier avec de la cryptographie forte, le flux de données se présente comme un flux aléatoire d'octets. Et curieusement, un flux de données aléatoires est rare dans le domaine de l'informatique. Du simple paquet TCP/IP qui circule sur le net à travers les routeurs, au film DivX sur notre disque dur, l'information est formatée en structures fixes, en types de fichier définis, en modèles hiérarchiques strictes. Typiquement, il n'y a pas de données aléatoires dans les ordinateurs, donc, un flux d'octets aléatoires apparaissant soudainement devient détectable.

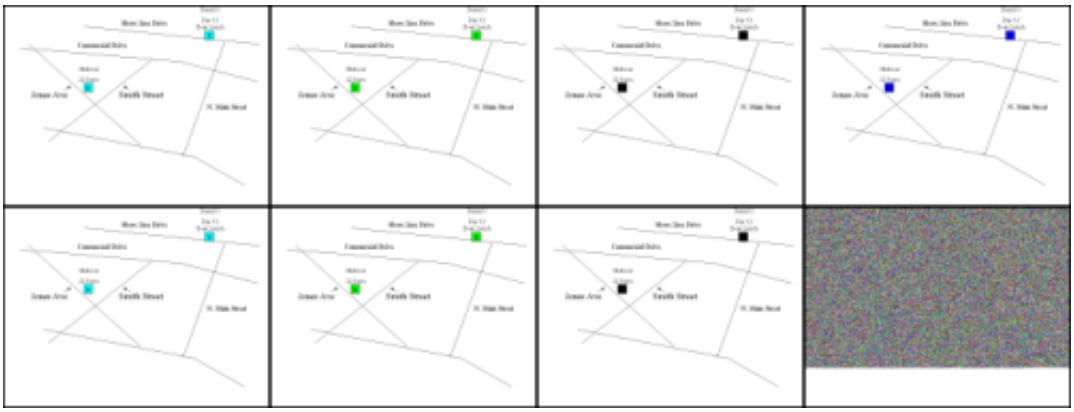
5.2.1.2 LSB et chiffrement

Nick DeBaggis du projet Honeynet[1], a décelé à l'aide d'un programme développé par ses soins la présence d'un message caché dans un fichier BMP. La figure 5.1 montre une étude de l'image. L'image est convertie en utilisant le bit 7 de chaque pixel, puis 6, etc. jusqu'au bit 0. On constate immédiatement la présence de bruit au bit de poids faible. Il s'agit d'une propriétés des données chiffrées et/ou compressées.

[1]Depuis plus de deux ans, le projet Honeynet propose régulièrement d'étudier des outils, des tactiques utilisées par des pirates lors d'intrusion pour démontrer la réalité de ces attaques, pour apprendre comment un pirate agit pour s'en protéger, et plus généralement pour améliorer les technologies et méthodes de collecte d'information.

Chaque mois, le *Scan of the Month Challenge* propose une analyse de trames réseaux suspectes. Le SOTM 24 propose de travailler sur l'analyse d'une disquette.

FIG. 5.1 – Etude d'une image BMP stéganographiée



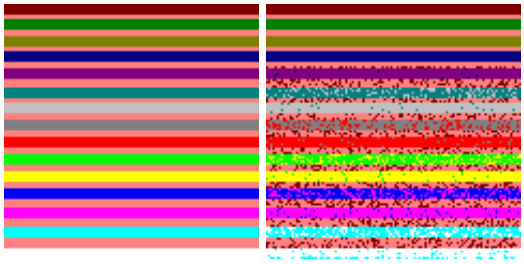
5.2.2 Problèmes liés aux techniques utilisées

L'utilisation de certaines méthodes ouvrent des failles pour la détection du stégo-objet, comme l'utilisation du LSB, ou des palettes ou des méthodes du domaine transformé.

5.2.2.1 Méthode LSB

Les techniques consistant à remplacer les bits les moins significatifs engendrent plusieurs inconvénients. Tout d'abord le niveau de sécurité de cette méthode est très faible. En effet, il suffit pour l'attaquant de vérifier les bits les moins significatifs pour détecter la présence ou non d'un message caché. Le deuxième inconvénient de cette méthode est que l'algorithme change les propriétés statistiques de l'image même si les bits sont choisis aléatoirement. Le troisième inconvénient est que si l'on utilise une image possédant de forts contrastes, on détectera la présence de modifications visuellement (figure 5.2).

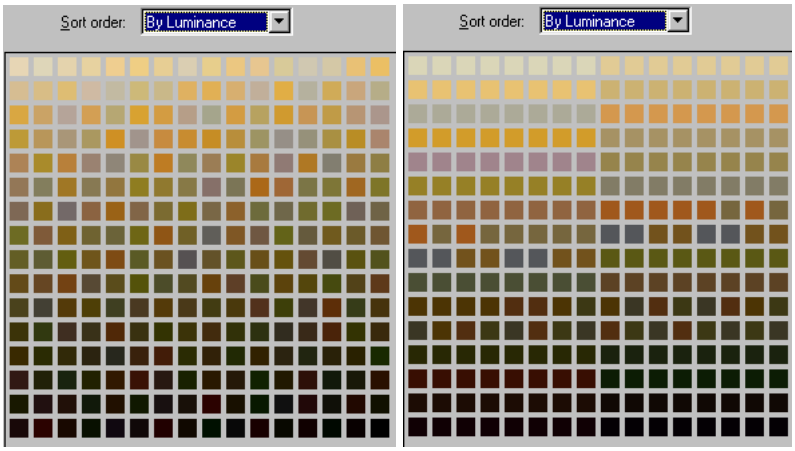
FIG. 5.2 – Cas d'une image avec de forts contrastes



5.2.2.2 Images basées sur les palettes

Neil F. Johnson et Sushil Jajodia, par exemple, ont montré que les systèmes stéganographiques pour les images basées sur une palette laissent des distorsions facilement détectables. On note aussi que les techniques basés sur la modification de l'ordre de la palette est immédiatement visible (figure 5.3).

FIG. 5.3 – Modification de l'ordre de la palette par S-tools

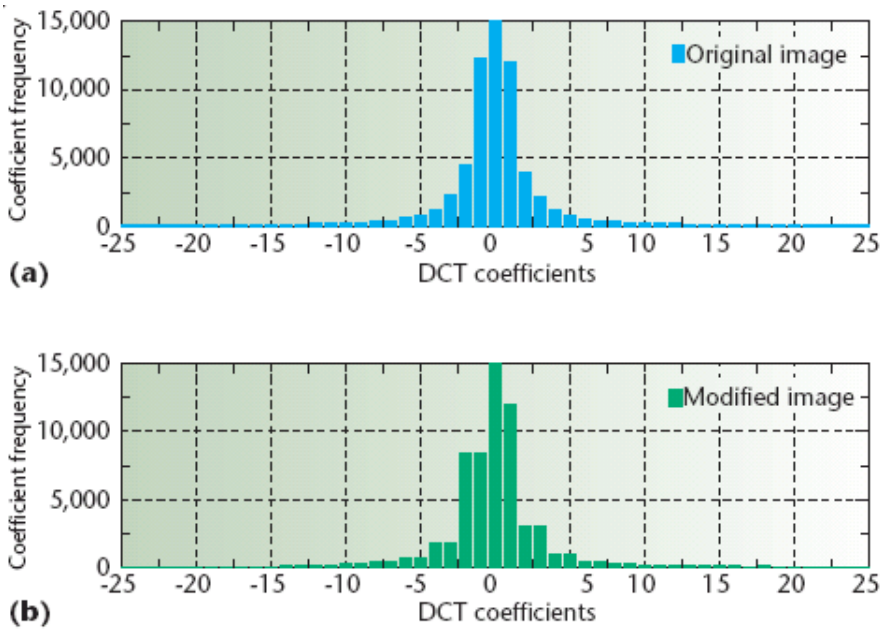


5.2.2.3 Transformée en cosinus discret

Un test statistique peut révéler si une image a été stéganographiquement modifiée. Westfield et Pfitzmann ont observé que pour une image donnée, l'encapsulation de données change l'histogramme des fréquences des couleurs d'une manière particulière. La différence de fréquence entre les couleurs adjacentes est réduite par le processus d'encapsulation. Dans un message chiffré, les 0 et les 1 sont distribués uniformément. Ceci est vrai pour le format JPEG mais au lieu d'observer les fréquences des couleurs, on constate les différences dans les fréquences des coefficients de la TCD.

La technique utilisée est nommée "*test de* χ^2", un exemple d'histogramme résultant est donnée dans la figure 5.4. Notons qu'il existe plusieurs améliorations de ce test et que de nombreux travaux sont menés dans ce domaine mais ils font appels à des notions compliquées d'analyse statistique. Citons entre autre les travaux de Jessica Fridrich et de son équipe, qui ont développé une méthode stéganalytique sur les deux logiciels considérés comme les plus fiables en matière de stéganographie appliquée à des fichiers au format JPEG : OutGuess et F5.

FIG. 5.4 – Histogramme des fréquences



5.2.2.4 Statistiques sur une image

De manière générale, toute stéganalyse massive passe par une étude statistique de l'image. Dans son ouvrage, Eric Cole [7] présente son outil de détection automatique qui utilise six différents tests statistiques sur des images.

Le logiciel de référence implémentant plusieurs tests statistiques est Stegdetect. C'est une "*detection framework*". Cet outil permet d'automatiser la détection d'informations dissimulées dans un medium image encapsulées via des logiciels spécifiques (voir l'exemple d'utilisation de Stegdetect en annexe).

5.3 Signatures

Un logiciel de stéganographie laisse une série d'octets caractéristiques ou *signature*. Parfois, le simple fait d'utiliser un format d'image particulier peut être une signature. Quelques exemples de ces signatures sont décrits dans cette section.

5.3.1 MandelSteg

Cet outil est unique dans le fait qu'il n'utilise pas d'images pré-existantes mais génére des fractales MandelBrot en tant que stégo-medium. En fonction des paramètres, l'image peut varier en couleur et en taille. Toutes les images MandelSteg générées ont une palette de couleur de 256 entrées et ont un motif récurrent détectable. On peut trouver dans chaque fractale 128 couleurs uniques avec deux entrées dans la palette pour chaque couleur.

5.3.2 Hide and Seek

Les versions 4.1 et 5.0 possèdent une caractéristiques communes à toutes les stégo-images résultantes. L'étude de l'histogramme révèle que toutes les palettes d'entrées sont divisibles par 4 pour toutes les valeurs de bit. De plus, l'image est rognée ou ajustée avec des bits de bourrage si elle n'a pas une taille valide.

5.3.3 Jsteg-Jpeg

En ajustant les coefficients TCD d'une image JPEG, les résultats attendus donnent un graphe des valeurs différentes de 0 homogène. Ce qui n'est pas le cas avec Jpeg-Jsteg. Il produit des graphes contenant plus d'erreurs, et donne des coefficients dupliqués à cause des erreurs d'arrondi exagérées. On constate donc une distorsion détectable pour certains coefficients.

5.4 Conclusion sur la stéganalyse

Les logiciels de stéganographie performants sont rares. Techniquement, l'exploitation d'un format croît avec sa complexité. En image, le format JPEG est très complexe, comparé à un format non compressé comme le bitmap. Pour visualiser une image de ce type il faut effectuer un nombre important d'opérations (voir 2.3.4). C'est pourquoi ceux qui "jouent" avec la stéganographie utilisent le format BMP. De plus, les programmes sérieux qui utilisent la stéganographie avec le format JPG, trichent. Invisible Secrets cache les données dans les commentaires de l'en-tête et SecureEngine ajoute les données cachées à la fin du fichier.

Chapitre 6

Les applications

Dans les chapitres précédents, l'état de l'art de la stéganographie et ses limites ont été étudiés. Qu'en est-il de ses applications ? Nous allons voir dans ce chapitre que la stéganographie peut trouver de nombreuses applications. Il reste encore à déterminer si ces applications relèvent uniquement de la technique de laboratoire.

6.1 La stéganographie inspire la terreur

La stéganographie à de multiples applications. Toutefois, les média véhiculent une image suspecte autour de la technique. Ceux qui cherchent à dissimuler n'ont ils rien à se reprocher ? Il existe de nombreux exemples. Selon les média, organisations terroristes, pédophiles, mafias et pirates informatiques utiliseraient très largement les nouvelles technologies et cette technique discrète pour communiquer entre eux.

6.1.1 Terrorisme sur Internet

En février 2001, *Wired News* mentionnait dans son article [1] "Secret Messages Come in .Wavs", l'existence de la stéganographie dans les images des sites d'eBay et Amazon. Des terroristes auraient eu recours à la stéganographie pour communiquer et Al-Qaida l'aurait même utilisée dans le cadre de ses préparatifs de l'attaque du 11 septembre.

6.1.1.1 Via les images

Relativisons tout de suite en indiquant les résultats de Niels Provos (figure 6.1). En effet, ce chercheur de l'université du Michigan a automatisé une recherche d'images

[1] `http://www.wired.com/news/print/0,1294,41861,00.html`

stéganographiées sur Internet. Il apparaît que sur 2 millions d'images téléchargés sur les forums USENET et le site de vente aux enchères eBay, seulement 1% de toutes les images semblaient dissimuler un contenu caché (environ 20000 images). Aucun message caché n'a cependant pu être déchiffré.

FIG. 6.1 – Résultat de Niels Provos

Table 2. Percentages of (false) positives for analyzed images.

TEST	EBAY	USENET
JSteg	0.003	0.007
JPHide	1	2.1
OutGuess	0.1	0.14

Quels que soient les résultats de cette étude, ce genre de scénario est possible. Utiliser ce genre de méthode est subtile au vu du nombre considérable d'images sur Internet. Dans notre cas, c'est une partie qui dépose un fichier pour une autre partie qui n'a pas connaissance de cette première. Cela revient à dire qu'un terroriste reçoit une information sans savoir de qui elle provient. Par conséquent, s'ils étaient pris ils ne connaîtraient pas l'identité de l'auteur du message.

6.1.1.2 Via l'audio

Les fichiers images ne sont pas les media uniques sur lesquels les techniques de stéganographie s'appuient. Les fichiers audio comme le WAV, MID, AU et le MP3 sont aussi idéaux car ils sont omniprésents. L'avènement des réseaux peer-to-peer a rendu aisé la dissimulation des fichiers MP3 sur un serveur donné.

Un autre avantage de l'utilisation des fichiers audio est qu'il peut être facilement transporté grâce aux lecteurs MP3. Si quelqu'un était interpellé en possession d'un baladeur MP3, cela n'attirerait pas l'attention. Mais les logiciels de stéganographie sont loin d'être parfaits comme nous l'avons vu. Ils laissent des signatures qui permettent aux observateurs avertis de déceler une anomalie.

6.1.2 Pédophilie

Les personnes possédant des données interdites, liées par exemple à la pédophilie, veulent les cacher. La stéganographie leur permet de cacher toute image dans un fichier, un son, une autre image "innocente", voire dans son propre disque dur.

6.1.3 Piratage informatique

Le magazine de sécurité informatique MISC évoque dans son édition de septembre[2], une utilisation malicieuse de la stéganographie par les hackers : un hacker peut par exemple coder un programme multi-threadé et distribué en mémoire. Ce type de code peut être fragmenté et envoyé à la victime par morceaux dissimulés via des procédés stéganographiques dans des images, le "réassemblage" se faisant localement, juste par un appel à l'endroit adéquat (en général, une pièce jointe et exécutée avec les droits d'écritures de l'utilisateur dans le répertoire temporaire du disque).

Un pirate peut également tenter de dissimuler un cheval de Troie ou tout autre code malveillant dans l'autorun d'un media amovible.

6.1.4 Espionnage industriel

Les entreprises ont des secrets à protéger (information stratégique, formule d'un nouveau médicament, code source d'un logiciel propriétaire). Voler ce genre d'information est possible voire probable.

Le vol d'information est estimé à 59 milliards de dollars par an aux états-unis, selon une récente étude de l'American Society for Industrial Security. Et Robert M. Wright du NIPCs Special Technology Application Unit, a évoqué l'utilisation de la stéganographie : "aujourd'hui ces voleurs d'information disposent de leurs propres outils : clé USB, email anonyme, logiciel, application peer-to-peer, infra-rouge et matériel sans fil, et stéganographie".

6.2 Protection des droits du citoyen

Quel est le but d'un citoyen lambda en utilisant cette technique ? On distingue l'attitude d'une personne située dans un pays démocratique et celle située dans un pays soumis à un régime non démocratique.

6.2.1 Dans les pays démocratiques

En Europe occidentale, l'usage de la cryptographie est autorisé et le contrôle est assuré uniquement dans le cadre de la fabrication de logiciels et de leur commercialisation (exportation notamment) ; la recherche universitaire est mondiale mais parfois classifiée. Il s'agit d'une arme. A ce titre, un corollaire de ce contrôle de la fabrication et exportation de ces produits peut aussi être considéré comme une garantie

[2] MISC n°9, sept-oct 2003, p.11.

(évaluation/certification) visant à limiter le sentiment de fausse protection décrit dans la section 4.8 sur certains logiciels “bidons” de stéganographie. Cela aussi appartient au rôle de l’Etat de protéger les citoyens. L’usage de la stéganographie dans le but de dissimuler l’existence d’un message chiffré de manière légitime n’est donc pas justifié. Le seul exemple concret d’une utilisation louable est celle du professeur David Touretzki. Il s’est spécialisé dans la stéganographie du code source du DeCSS. Il s’agit d’une méthode pour déchiffrer un flux DVD, ce qui permet :

– De copier un DVD sur un disque dur ;
– De pouvoir programmer des lecteurs de DVD notamment pour des OS tels que Linux.

Au nom du Millenium Act, les grands studios américains ont interdit la diffusion de ce code source : toute diffusion de celui-ci sur le territoire américain est un délit. Cet astucieux professeur ne diffuse pas le code source in-extenso[3] : il utilise des moyens divers et variés (dans plusieurs images, sous la forme d’un nombre premier gigantesque encryptant en son sein le code source, dans une séquence d’ADN, dans une partie de démineur, dans des codes-barre, etc.) faisant appel à la stéganographie et ne tombe ainsi pas sous le coup de la loi.

6.2.2 Dans les pays non démocratiques

Les pays non démocratiques, les dictatures sont encore les seuls à contrôler totalement les mécanismes cryptographiques. On peut donc voir la stéganographie comme une alternative. Bruce Schneier [12] déconseille cependant son utilisation dans ce cas précis. Etant donnée, que l’oreille indiscrète écoutant le message a elle aussi accès au même logiciel. Elle sera à l’alerte d’un message stéganographié.

6.3 Applications du tatouage

Avec l’avancée rapide des technologies, on constate une certaine inquiétude au sujet des droits “copyrights”. Le watermarking permet de :

– Protéger les possesseurs de copyright sur des documents numériques en cachant une signature dans l’information de sorte que même une partie modifiée du document conserve la signature.
– Découvrir l’origine de fuites en marquant de façon cachée et unique chaque copie d’un document confidentiel.

[3] http ://www.cs.cmu.edu/~dst/DeCSS/Gallery/Stego/

6.3.1 Caractéristique du tatouage numérique

Le tatouage numérique ou watermarking, est une application de la stéganographie. Elle a des caractéristiques qui diffèrent comme le montre le tableau 6.2.

TAB. 6.1 – Digital Watermarking vs Stéganographie

CARACTÉRISTIQUE	STÉGANOGRAPHIE	DIGITAL WATERMARKING
Quantité de donnée	Autant que possible	Très peu
Imperceptibilité	Très difficile	Pas important avec les marques visibles
Robustesse	Importante	Importante
But de l'attaquant	Détecter les données	Supprimer les données
But de l'utilisateur	Cacher de l'information qui ne doit pas être détectée	Encapsuler une signature qui prouve les droits de propriété
Utilisation courante	Espionnage industriel Canaux de communication caché Dealers, Terroriste	Protéger les droits du propriétaire d'une image numérique, d'une vidéo ou d'un contenu audio

6.3.2 Indexation

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

– Extraction des vecteurs caractéristiques (ou *feature vectors*) du medium ;
– Recherche des media similaires à un medium servant de requête.

Il existe un grand nombre d'attributs représentatifs d'un medium. 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. Le tatouage sert alors à ajouter une information supplémentaire, comme un mot clé par exemple, qui facilite la recherche. Cette application du tatouage est celle qui nécessite le moins de robustesse. Cette application ressemble 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 medium et les données ont un lien.

6.3.3 Tatouage faible

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

– Le tatouage fragile : si le medium a subi plus de changements que ceux autorisés, la détection de la marque échoue ;
– Le tatouage fragile évolué : si le medium a été altéré, la marque indique l'endroit où les modifications ont été effectuées.

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

6.4 Connaître l'existence de la stéganographie

Il est nécessaire de rester conscient quant à l'existence de la stéganographie. Il y a des précautions à prendre pour l'utiliser de manière sûre et du point de vue de l'attaquant, la détection est facilitée si l'on connaît les méthodes utilisées.

6.4.1 Pour l'utilisateur

Du point de vue de l'utilisateur, certaines précautions doivent être prises si l'on désire utiliser la stéganographie.

Un message contenant soudainement une image ou un son est suspect. Il faut établir un long processus de communication où l'on envoie régulièrement des images innocentes sur un thème donné. Ainsi, au bout d'un temps donné, cette attitude "endormira" l'attaquant. L'image stéganographiée passera inaperçue dans le flot habituel d'image.

Plus un message est petit, plus il est difficile de le détecter de manière statistique. La détection dépend aussi de la compression des images. Zollner propose une approche théorique pour résoudre le problème de la stéganographie sûre en employant une sélection non déterministe :

1. La clé secrète utilisée pour encapsuler le message n'est pas connue de l'attaquant.
2. L'attaquant ne connaît pas le medium de couverture.

En pratique ces deux conditions sont facilement réalisables, il suffit de créer un medium de couverture avec un appareil photo numérique ou en scannant des photos de vacances à condition que les photos d'origine ne soient pas publiquement disponibles.

6.4.2 Pour l'attaquant

Du point de vue de l'attaquant, une veille technologique sur ce domaine doit être faite, afin de pouvoir détecter l'utilisation de ce genre de technique. Un attaquant ayant accès physiquement à l'ordinateur peut appliquer la méthodologie suivante :

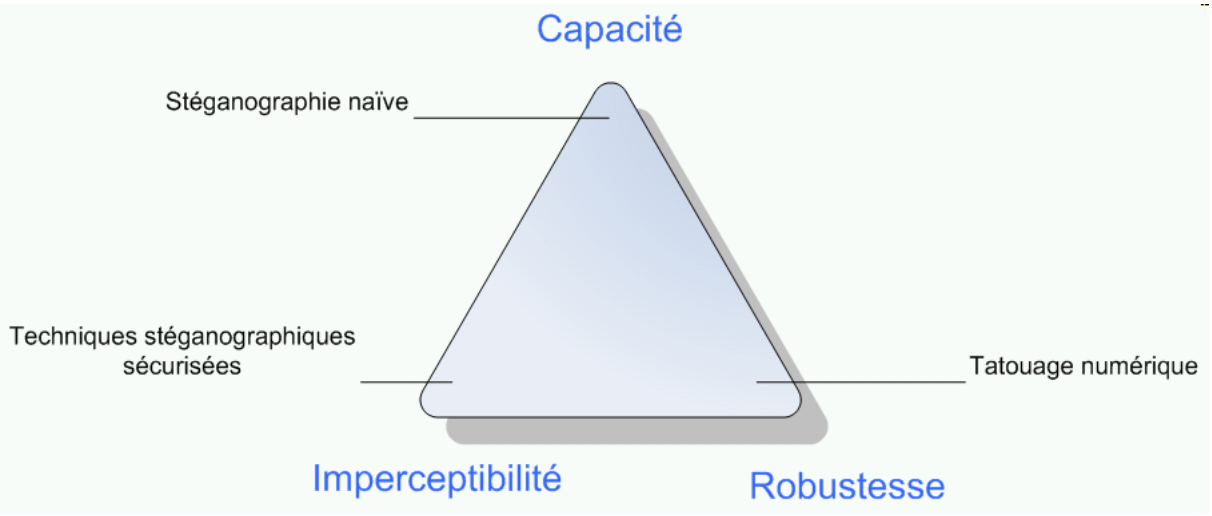
– Ecoute des mots clés ;
– Etude des fichiers internet temporaires et des sites favoris liés à la stéganographie ;
– Etude des logiciels de stéganographie installés.

Chapitre 7

Conclusion

Dans ce document, nous avons voulu présenter la problématique de la dissimulation d'informations. Selon les attentes, les contraintes d'imperceptibilité, de capacité et de robustesse varient. Cependant, comme ces besoins sont contradictoires, un compromis est nécessaire.

FIG. 7.1 – Le triangle "magique"



La stéganographie, 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'algorithmes d'insertion et de détection que de protocoles susceptibles de les encadrer afin d'en accroître la sécurité.

7.1 Un manque de maturité

La stéganographie ne semble pas encore assez mature ni assez sûre. Si on parle de sécurité totale bien sûr, ou impossibilité de détecter une quelconque information cachée. Le niveau de sécurité qui est demandé est proportionnel à l'importance des données à cacher.

Si un terroriste veut utiliser la stéganographie, il doit savoir que Wetstone (financé par le gouvernement des Etats-unis) développe des produits comme StégoWatch qui peut automatiquement détecter et faire des attaques par dictionnaire des informations cachées par de nombreux programmes. Ou encore StegDetect, logiciel open-source développé par Niels Provos.

7.2 Futur de la stéganographie

Selon une personne proche de la sécurité et du renseignement, la stéganographie est moins utilisée qu'il n'y paraît. Les organisations criminelles qui font l'objet d'une grande médiatisation semblent plutôt privilégier des moyens plus traditionnels (facteurs, chiffrement) ou s'achetant parfois les services d'experts.

La relance de popularité de ce domaine est principalement dû aux informations diffusées par des magazines après le 11 septembre. Le gouvernement américain, en particulier, pourrait avoir contribué à l'exagération du phénomène. Il existe trop peu de logiciels performants dans le monde de la stéganographie. Ils disposent d'une empreinte, véritable signature qui les rend détectables.

L'art de dissimuler le fait même de l'échange est devenu suspect, ce qui semble être une bonne raison pour que l'avenir de la stéganographie soit compromis (pour le domaine public). En matière de cryptographie, la recherche s'oriente vers la cryptographie quantique et la logique floue. Dans l'image, les travaux sont orientés sur la détection du contenu de l'image.

Les applications de la stéganographie sont actuellement sujettes à de constantes améliorations. Les chercheurs portent une attention particulière sur le tatouage numérique et le fingerprinting. Ces recherches qui s'effectuent dans un climat particulier et paradoxal. Le sentiment d'atteinte aux libertés fondamentales et le terrorisme tendent à réduire les tentatives de recherche dans ce domaine, tandis que la légiférération européenne sur la propriété intellectuelle, les droits d'auteurs, la brevetabilité logicielle pourraient contribuer à développer des mécanismes plus sûrs.

Bibliographie

[1] Fabien Petitcolas, *Information Hiding for Steganography and Digital Watermarking*, Artech House, inc., 2000.

[2] R.J. Anderson and F. Petitcolas, *On the limits of steganography*, IEEE Journal on Selected Areas in Communications, 16(4) :474481, 1998.

[3] Fabien A. Petitcolas, R. J. Anderson and Markus G. Kuhn, *Information Hiding - A Survey*, Proceedings of the IEEE, 87(7) :1062-1078, July 1999.

[4] Caroline Fontaine, *Tatouage d'images numériques*, dossier pour la science (Edition Francaise de Scientist American), (120) :7277, Dossier Hors série Juillet/Octobre 2002.

[5] Caroline Fontaine, *Marquage d'images en vue de la protection des droits d'auteur*, thèse de Doctorat de l'Université de Paris.

[6] Greg Kipper, *Investigators Guide to Steganography*, Auerbach Publication a CRC Press Company, 2004.

[7] Eric Cole, *Hiding in Plain Sight : Steganography and the Art of Covert Communication*, Willey Publising, Inc., 2003.

[8] Manuel Chesneau, *Quelques réflexions sur l'application du Tatouage dimages au domaine médical*, soutenance de projet, 2003.

[9] Chih-Hsuan TZENG, Zhi-Fang YANG, Wen-Hsiang TSAI, *A New Data Hiding Technique for Palette Images*, 2000.

[10] Jian ZHAO & Echard KOCH. Embedding Robust Labels Into Images For Copyright Protection. 1995.

[11] Bruce Schneier, *Applied Cryptography*, 2nd Ed, 1996 Wiley.

[12] Bruce Schneier, *Steganography : Truths and Fictions*, Cryptogram liste de diffusion créée par l'auteur, 2002

[13] Steganalysis, http ://guillermito2.net/stegano/ideas.html.

[14] Neils Provos, Stegdetect - Stegbreak, http ://www.outguess.org/detection.

[15] Niels Provos & Peter Honeyman, *Hide and Seek : An Introduction to Steganography*, IEEE SECURITY & PRIVACY, Mai-Juin 2003.

[16] Tom Kellen, *Hiding in Plain View : Could Steganography be a Terrorist Tool ?*, GSEC Practical Assignment v1.2f, 2001.

[17] Wayner, P., *Mimic Functions*, Cryptologia, vol. XVI/3, 1992, pp. 193214.

[18] A. Peltier, L. Renard, *Projet Etalement de Spectre*, IUT GTR de Blois, 2002.

[19] Bender, W., D. Gruhl, and N. Morimoto, *Techniques for data hiding*, IBM Systems Journal, vol. 35, no. 3/4, 1996, pp. 131-336.

[20] Jessica Fridrich, Miroslav Goljan, Dorin Hogea, *Attacking the Outguess*, 2000.

[21] http ://www.infosyssec.com/infosyssec/stendig1.htm, portail de sécurité regroupant tous les liens indispensables sur la stéganographie.

[22] Linux Magazine, Stéganographie ou l'art de cacher ses affaires, p.32, Avril 2001.

Annexes

Algorithme de logiciel de stéganographie dans le JPEG

FIG. 7.2 – Algorithme de JSTEG

```
Input: message, cover image
Output: stego image
while data left to embed do
        get next DCT coefficient from cover image
        if DCT ≠ 0 and DCT ≠ 1 then
                get next LSB from message
                replace DCT LSB with message LSB
        end if
        insert DCT into stego image
end while
```

Figure 3. The JSteg algorithm. As it runs, the algorithm sequentially replaces the least-significant bit of discrete cosine transform (DCT) coefficients with message data. It does not require a shared secret.

FIG. 7.3 – Algorithme de OUTGUESS

```
Input: message, shared secret, cover image
Output: stego image
initialize PRNG with shared secret
while data left to embed do
        get pseudo-random DCT coefficient from cover image
        if DCT ≠ 0 and DCT ≠ 1 then
            get next LSB from message
            replace DCT LSB with message LSB
        end if
        insert DCT into stego image
end while
```

Figure 6. The OutGuess 0.1 algorithm. As it runs, the algorithm replaces the least-significant bit of pseudo-randomly selected discrete cosine transform (DCT) coefficients with message data.

FIG. 7.4 – Algorithme de F5

Input: message, shared secret, cover image
Output: stego image
initialize PRNG with shared secret
permutate DCT coefficients with PRNG
determine k from image capacity
calculate code word length $n \leftarrow 2^k - 1$
while data left to embed **do**
    get next k-bit message block
    repeat
        $G \leftarrow \{n$ non-zero AC coefficients$\}$
        $s \leftarrow k$-bit hash f of LSB in G
        $s \leftarrow s \oplus k$-bit message block
        if $s \neq 0$ **then**
            decrement absolute value of DCT coefficient G_s
            insert G_s into stego image
        end if
    until $s = 0$ or $G_s \neq 0$
    insert DCT coefficients from G into stego image
end while

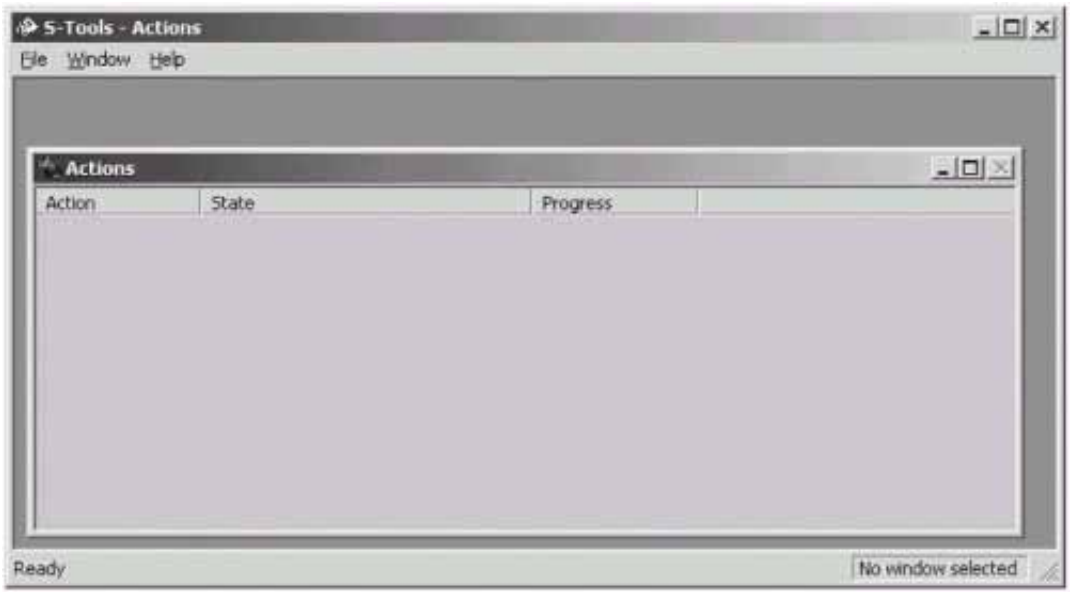
Figure 9. The F5 algorithm. F5 uses subtraction and matrix encoding to embed data into the discrete cosine transform (DCT) coefficients.

Utilisation d'un framework stéganographique : S-Tools

S-Tools est un programme écrit par Andy Brown. C'est peut être aujourd'hui l'outil stéganographique le plus populaire sur Internet. Les fichiers BMP, GIF et WAV peuvent être utilisés pour dissimuler des messages secrets. Il est facile à utiliser, en faisant simplement glisser les fichiers vers l'application. S-Tools cache les messages secrets dans le fichier couverture via des bits aléatoires. Les bits utilisés sont déterminés en utilisant un générateur de chiffres pseudo-aléatoires. Cette insertion rend la présence et l'extraction des messages secrets plus difficile.

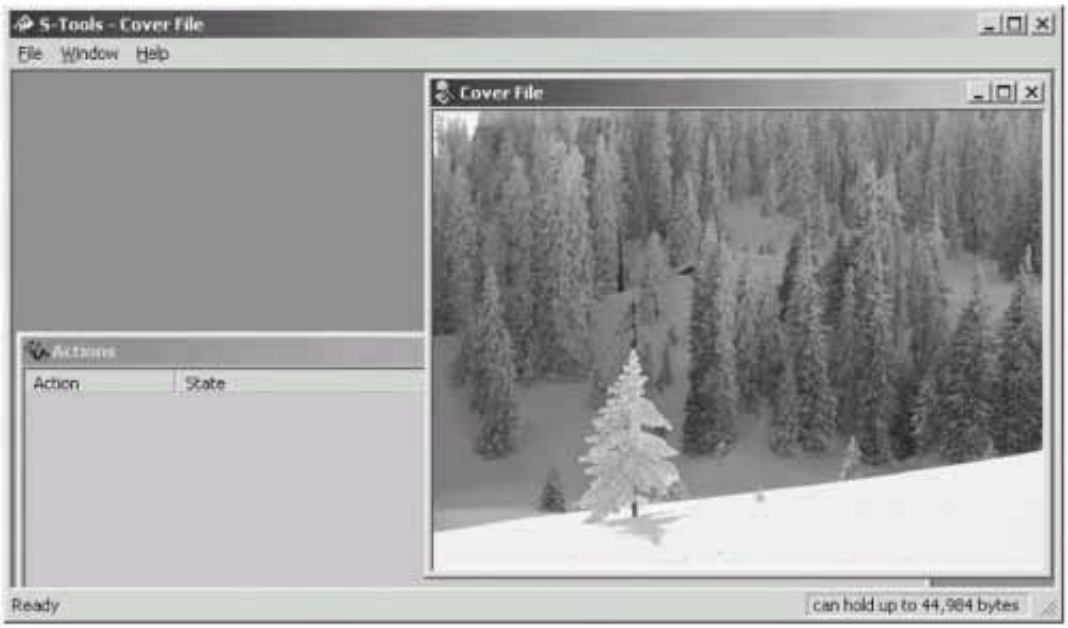
La figure suivante montre la fenêtre principale de S-Tools. Le fond en gris sombre est utilisé pour déplacer les images ou les sons dans l'application. Cet espace est réservé à l'insertion et extraction du message secret. La fenêtre "Actions" en gris clair montre le statut des tâches qui ont lieu.

FIG. 7.5 – Fenêtre principale



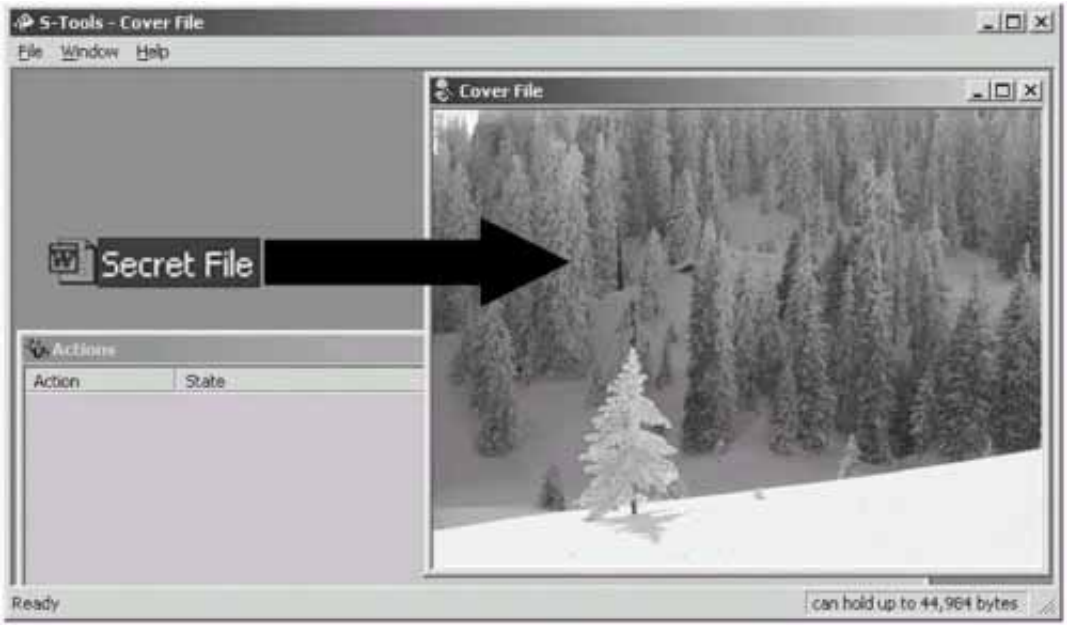
Après l'ouverture de S-Tools (S-Tools.exe) et de Windows Explorer, on déplace le fichier couverture Cover File.gif dans l'espace principale de travail de S-Tools. La figure suivante montre le fichier GIF dans l'espace principal de travail. Notons que le texte qui apparaît dans le coin en bas à droite est l'approximation du nombre de bits pour le stockage. Pour l'exemple, il y a 44.984 octets disponibles pour cacher des données.

FIG. 7.6 – Médium de couverture



Pour cacher le fichier secret, on glisse le fichier file.doc à l'intérieur de S-Tools dans le fichier de couverture comme montré dans la figure suivante :

FIG. 7.7 – Dissimulation d'informations



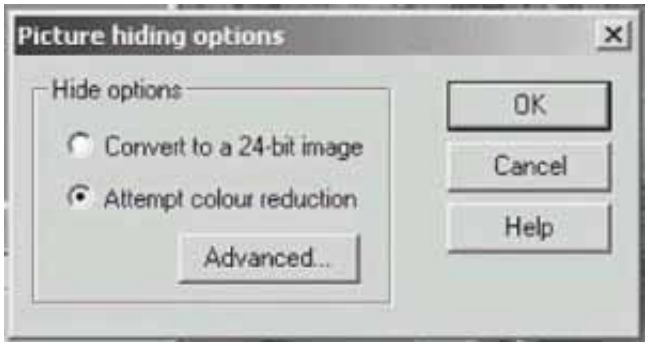
A partir de ce point, S-Tools demande un mot de passe (Passphrase). Le mot de passe est utilisé pour générer un nombre aléatoire qui est utilisé pour insérer les bits dans le fichier de couverture. S-Tools donne le choix entre IDEA, DES, TripleDES, et l'algorithme de cryptage MDC. La figure qui suit montre la boite de dialogue.

FIG. 7.8 – Chiffrement

Le mot de passe et l'algorithme de cryptage doivent être mémorisés de façon à extraire le fichier secret. Si IDEA était utilisé durant le processus d'insertion et que le DES est sélectionné durant le processus d'extraction, le fichier secret ne serait pas extrait. On entre un mot de passe et on sélectionne l'algorithme de cryptage IDEA.

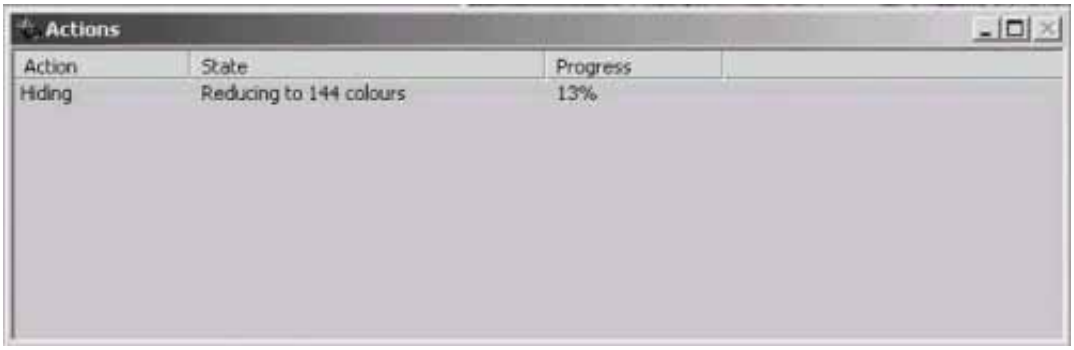
Une boite de dialogue apparaît, afin de sélectionner les choix du processus d'encodage. Le choix définit comment le fichier de couverture sera traité durant l'insertion du fichier secret. L'image de la boite de dialogue des options est montrée dans la figure suivante.

FIG. 7.9 – Options



On prend le choix par défaut et S-Tools débutera le processus d'insertion du fichier secret dans le fichier couverture. La fenêtre "Action" fera apparaître le statut du processus.

FIG. 7.10 – Fenêtre Action



Quand S-Tools finit d'insérer les données, l'image résultante sera affichée avec le titre "hidden data". Ce fichier GIF est le produit de l'insertion du fichier secret dans le fichier de couverture.

De la même façon que la méthode utilisée dans les systèmes de stégo-fichiers, S-Tools a dispersé les bits du fichiers partout dans l'espace du disque. C'est indétectable dans un visualiseur classique, mais le fichier est bien inséré.

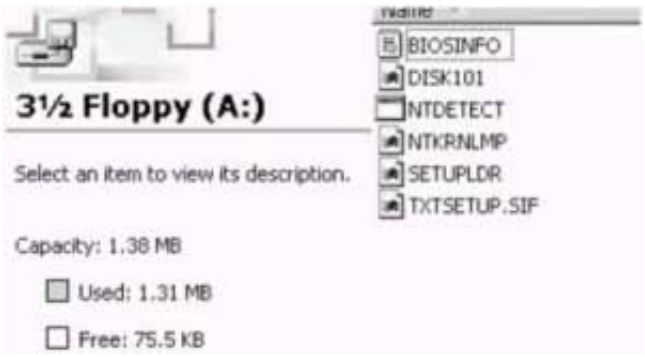
S-Tools Version 3 a la possibilité de cacher des informations dans une plage inutilisée sur une disquette. Tant que ce programme n'est pas disponible à grande échelle sur Internet, il est encore possible de le trouver.

S-Tools permet de cacher des fichiers dans des espaces inutilisés d'une disquette. Toute disquette, quand elle est formatée, est divisée en secteur. Chaque secteur sur un disque peut contenir 512 octets d'information. Sur une disquette de 1.44 Mo, il y a $1440 \times 1024/512 = 2888$ secteurs. Quand on charge un fichier sur une disquette, DOS calcule combien de secteurs on a besoin pour stocker le fichier et écrit ces informations dans la table d'allocation du fichier.

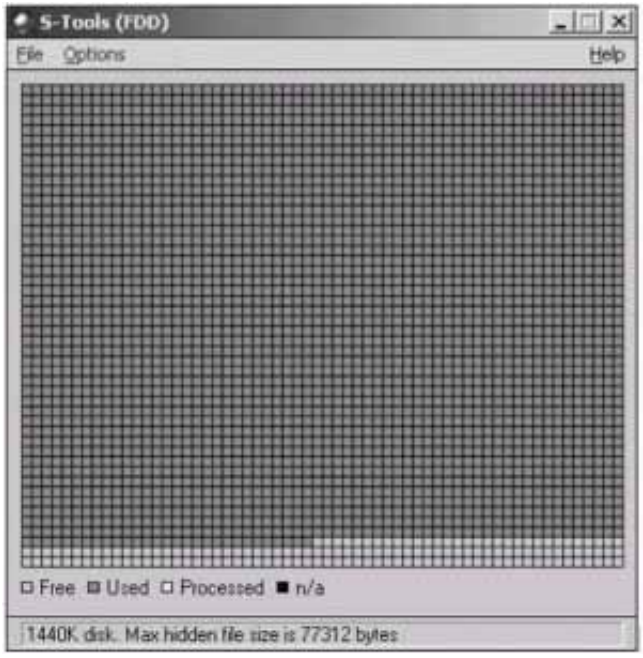
Le module FDD de S-Tools regardera la table d'allocation pour décider quels secteurs du disque n'ont pas été utilisés, et permettra de cacher des information dans ceux-ci. S-Tools ne cachera pas des informations dans des secteurs consécutifs car se serait trop facile à détecter. Au lieu d'utiliser un générateur de nombres aléatoires pour choisir quels secteurs libres à utiliser, S-Tools ajoutera une sécurité supplémentaire en permettant de remplir tous les autres secteurs inutilisés sur le disque avec des données aléatoires.

La fonction "Analyse Disk" affiche une carte d'usage de la disquette et nous informe comment l'information peut être cachée. Les secteurs marqués en rouge sont les seuls que S-Tools ne peut utiliser car des fichiers sont déjà stockés. La barre de statut en bas de l'écran nous informe quelle quantité d'information on peut cacher sur la disquette.

FIG. 7.11 – Fonction Analyse Disk



(a) Disquette



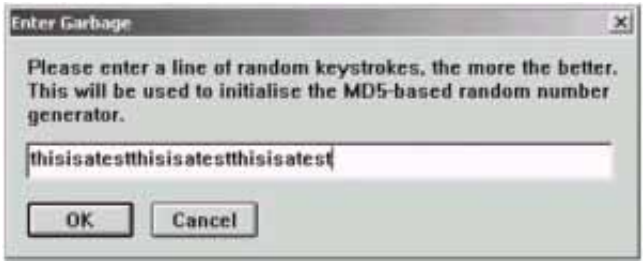
(b) Secteurs utilisables

Si on rempli l'espace libre sur une disquette après y avoir caché des fichiers, on perd les fichiers. Après avoir caché des fichiers, S-Tools oubliera leur présence jusqu'à ce que l'on décide de les révéler.

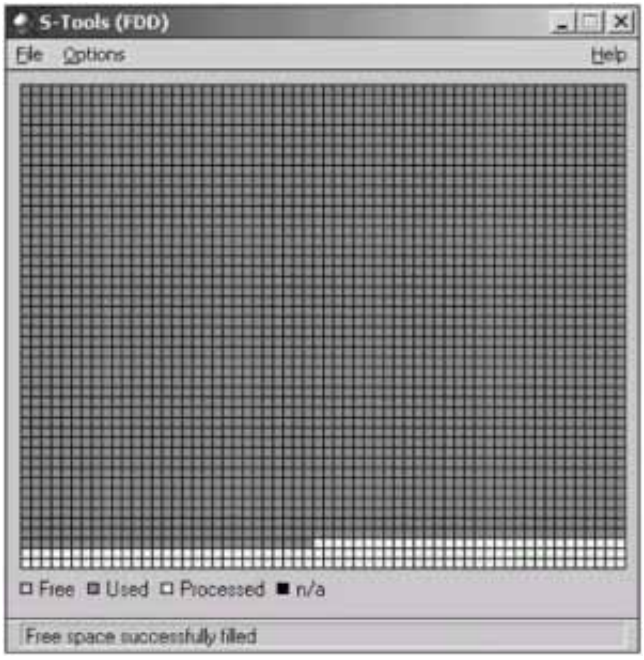
La fonction "Hide File" va permettre de cacher un fichier sur l'espace libre d'un disque. On utiliser la fonction "Analyse Disk" pour trouver l'espace libre.

Après avoir choisi le fichier à cacher on peut choisir ou non de crypter le fichier avant de l'insérer. L'utilisation d'un cryptage est recommandée dans le cas où le fichier est déjà crypté car le mot de passe qui sera entré sera aussi celui qui générera le générateur de nombres aléatoires qui est utilisé pour choisir les secteurs qui contiendront le fichier caché.

FIG. 7.12 – Analyse Disk



(a) Sélection du fichier



(b) Vu des secteurs

Le programme S-Tools fait une taille de 589 ko. Il est suffisamment petit pour être

logé sur une disquette. Comme beaucoup d'autres programmes stéganographiques, S-Tools laisse peu de traces de son passage. Pourtant, il existe un défaut : S-Tools réduit le nombre de couleur dans le fichier de couverture à un minimum de 32. De cette manière, les images en niveau de gris sont dégradées.

Dissimulation dans un fichier texte : Snow

Présentation

Snow[1] est un logiciel libre et gratuit développé par Matthew Kwan. Il fonctionne sous Windows et une version est disponible pour Unix. Selon Bender et al., le texte numérique est le support le plus difficile à utiliser pour cacher des données. Ceci est due au fait que les caractères supplémentaires et la ponctuation peut être facilement repéré par le lecteur. Le programme Snow utilise la méthode des espaces pour cacher des données dans un fichier texte ; il ajuste le nombre d'espaces de chaque ligne pour encoder son message. Snow se fie au fait que la plupart des visualiseurs et éditeurs de texte n'affiche pas ces caractères par défaut ce qui masque le fait qu'un message soit dissimulé.

Cas d'utilisation

Selon le manuel d'utilisation, les données sont dissimulées en concaténant une séquence d'espaces (7 maximum) disséminée dans les tabulations. Typiquement, elle permet de cacher 3 bits de données toutes les 8 colonnes du medium de couverture. Le logiciel donne le nombres d'octets utilisables en fonction du medium. Une fonctionnalité de compression est aussi disponible, ainsi que le protocole de chiffrement ICE. On cachera nos informations à l'aide de la commande suivante :

```
snow -C -f secret.txt -p passwd medium.txt stego.txt
```

NOM	medium.txt	secret.txt
TAILLE	2012 bits	107 bits

Le fichier original est compressé à 41,87% et l'espace utilisé par le message caché est de 25,14%. On extrait ensuite le fichier par la commande :

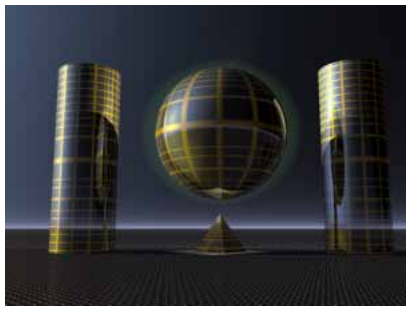
```
snow -C -p passwd stego.txt
```

[1] http ://www.darkside.com.au/snow

Exemple de stéganalyse : Stegdetect

FIG. 7.13 – Médium et image à dissimuler

(a) secret.jpg

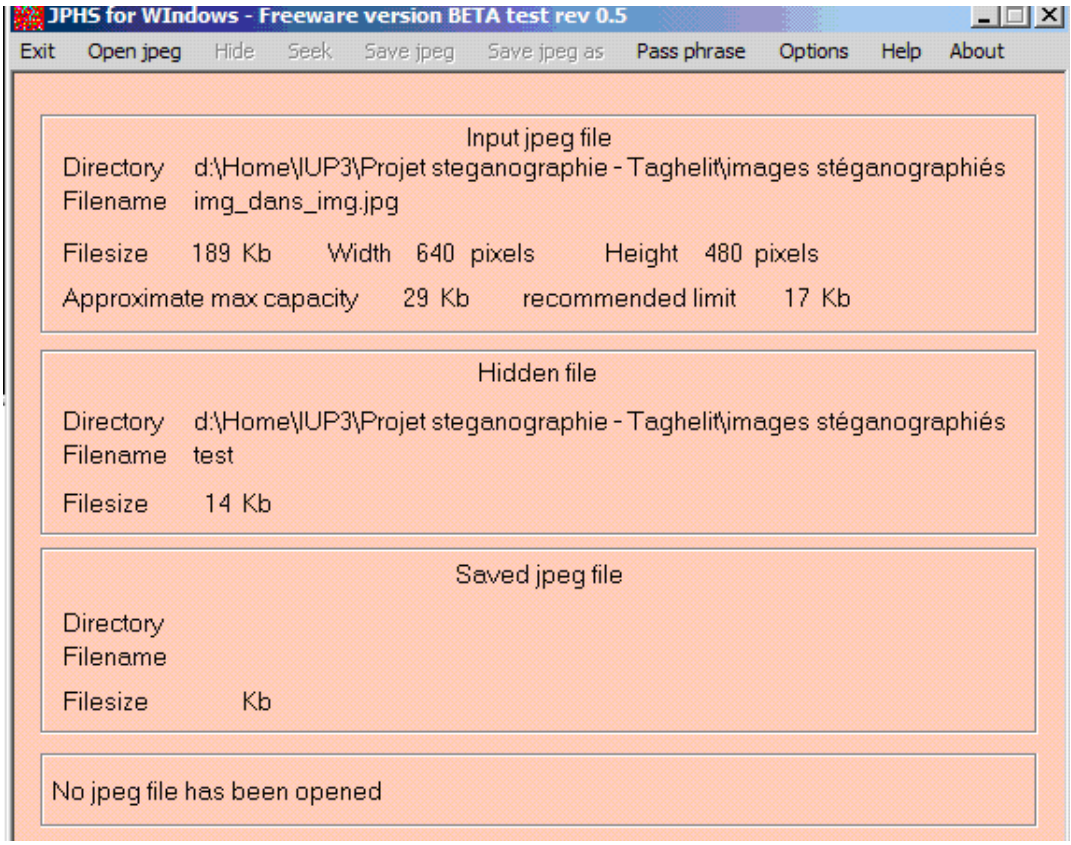


(b) yellowsphere.jpg

JPHide&Seek permet de dissimuler des données dans un fichier jpeg. Pour cela nous avons besoin d'une image porteuse : `yellowsphere.jpg`. Dans le menu, "Open Jpeg" nous permettra de définir ce fichier en tant que stégo-medium. On notera qu'un fichier jpeg peut cacher seulement 10% de sa propre taille.

On clique ensuite sur "Hide" pour sélectionner l'image qui sera dissimulé : `secret.jpg`. Une boîte de dialogue nous demande ensuite une phrase mot de passe ; on rentre "test". Et on enregistre le résultat sous le nom "`img_dans_img.jpg`".

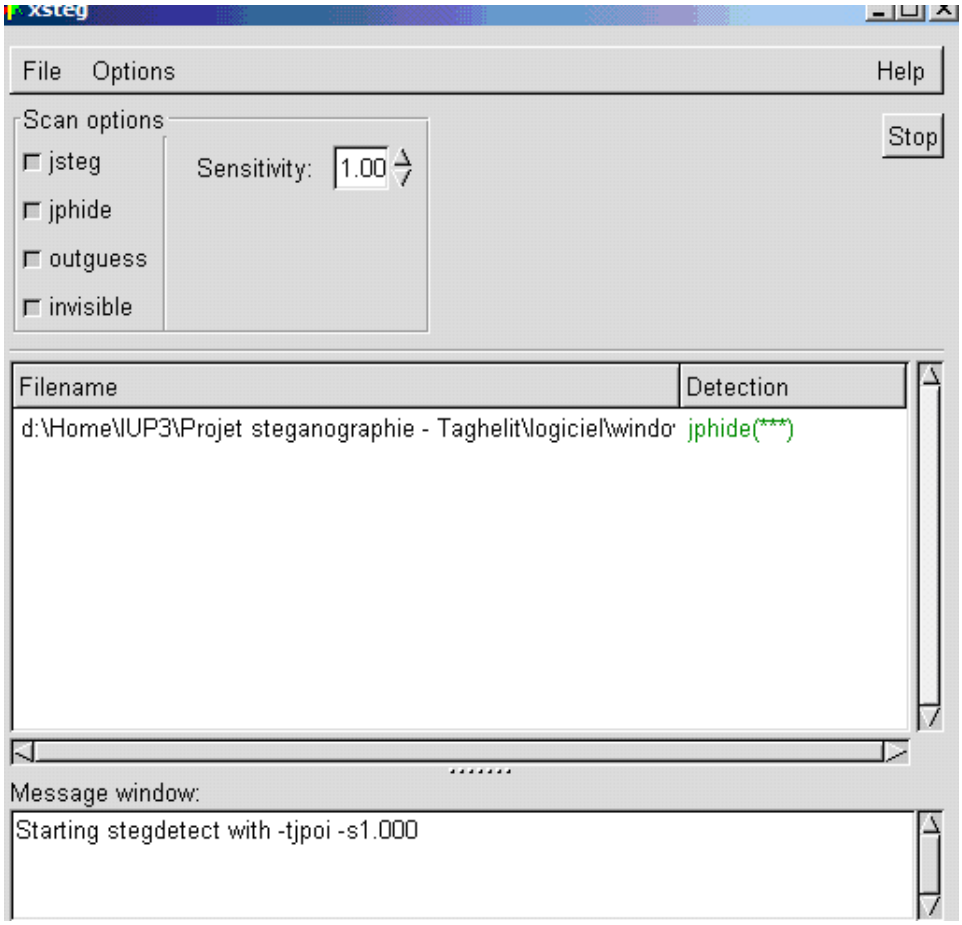
FIG. 7.14 – JPHide and Seek



Le stégo-medium est de 189 ko contre 184 ko à l'origine. Le fichier caché est de 14 ko.

Nous allons maintenant effectuer une attaque sur notre système à l'aide du logiciel stegdetect développé par le chercheur Niels Provos (disponible pour Windows et Linux). En lançant xsteg, l'interface graphique du logiciel, nous ouvrons notre image lançons une attaque.

FIG. 7.15 – Interface graphique de Stegdetect



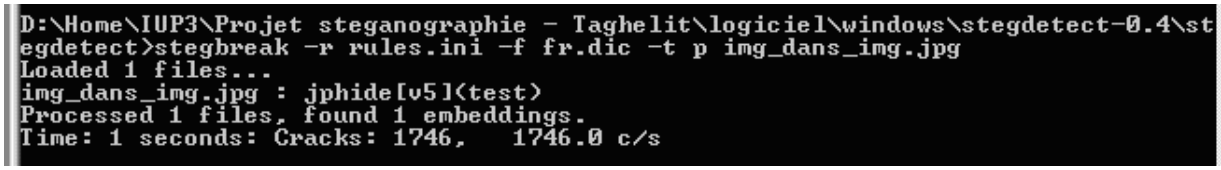
Stegdetect 0.4 peut détecter des fichiers cachés par quatre logiciel : jphide&seek, jsteg, outguess et invisible secrets. Quatre logiciels considérés comme fiable dans le monde de la stéganographie et du jpeg.

La détection est immédiate. L'image a été stéganographié à l'aide de JPhide&seek.

Le système de stéganographie utilisé est donc cassé, puisque nous avons détecté la présence dune information cachée dans `img_dans_img.jpg`.

On peut aller encore plus loin en essayant de retrouver le mot de passe utilisé à laide de Stegbreak développé par le même auteur. Une attaque par dictionnaire est effectuée sur notre stégo-image et nous obtenons le résultat suivant :

FIG. 7.16 – Attaque par dictionnaire à l'aide de Stegbreak



`-r rules.ini` permet d'utiliser des règles de transformation qui seront appliquées à la liste de mot.

`-f fr.dic` spécifie le fichier qui contient les mots pour une attaque par dictionnaire

`-t p` car nous savons que notre image contient un message caché par le logiciel JPHide&Seek.

Le mot de passe trouvé est : `test`.

Ce tutorial montre que si certaines précautions ne sont pas prises, il est facile de détecter une image.

Remarque :

– Stegdetect ne peut détecter les messages dissimulés avec JSteg plus petits que 50 octets.
– Le taux de fausses détections pour Stegdetect est de 20% pour les images cachés avec JPHide et de 60% pour les images cachées avec Outguess.