Stéganographie : visite de l'univers numérique pour dissimuler des informations

F. Raynal[1], F. A. P. Petitcolas[2], C. Fontaine[3]

[1]f.raynal@miscmag.com
[2]fabienpe@microsoft.com
[3]caroline.fontaine@lifl.fr

24 avril 2002

1 Introduction

La *stéganographie*[1] cherche à cacher un *message secret*[2] dans un médium de sorte que personne ne puisse distinguer un médium vierge d'un stégo-médium. La nature de l'information dissimulée ne revêt pas d'importance (un texte en clair ou sa version chiffrée indifféremment). Ce message n'a *a priori* aucun lien avec le stégo-médium qui le transporte. Ainsi, alors que la cryptographie cherche à dissimuler le contenu d'un message, la stéganographie s'efforce de cacher l'existence même d'une communication.

Lorsqu'un attaquant tente uniquement de détecter si un message transite dans un médium sur le canal de communication, on dit de lui qu'il est *passif*. La plupart des solutions de stéganographie ne considèrent que ce type d'attaquant, bien qu'il puisse également être *actif* : l'attaquant modifie alors systématiquement le stégo-médium qui transite sur le canal.

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

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

Par exemple, la correspondance suivante, dont l'authenticité n'est pas garantie, entre Alfred de Musset et Georges Sand semble anodine :

> Quand je mets à vos pieds un éternel hommage,
> Voulez-vous qu'un instant je change de visage ?
> Vous avez capturé les sentiments d'un coeur

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

Que pour vous adorer forma le créateur.
Je vous chéris, amour, et ma plume en délire
Couche sur le papier ce que je n'ose dire.
Avec soin de mes vers lisez les premiers mots :
Vous saurez quel remède apporter à mes maux.
Alfred de Musset

Cette insigne faveur que votre coeur réclame
Nuit à ma renommée et répugne à mon âme.
George Sand

Cependant, le sens est tout autre lorsqu'on lit uniquement le premier mot de chaque alexandrin...

Le contexte dans lequel se situe un schéma de stéganographie permet de distinguer différentes catégories :

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

Il est possible de présenter les techniques de stéganographie selon différentes classifications. Par exemple, N. Johnson et S. C. Katzenbeisser dans [KP99] le font d'après les modifications induites sur le support. Ils relèvent alors six catégories :

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

Nous avons opté pour une présentation reposant sur le choix du support de dissimulation. La stéganographie est une discipline ancienne et les schémas proposés actuellement tentent de reproduire dans l'univers numérique ce qui se faisait autrefois avec de l'encre et du papier. Pour cette raison, les algorithmes actuels utilisent majoritairement du texte et des média (fichiers sons, images, vidéos) comme support.

Précisons tout de suite un point souvent mal compris : la dissimulation dans un médium est réalisée **indépendamment** du format utilisé pour représenter ce médium. Ainsi, nous ne considérons pas comme de la stéganographie l'emploi de commentaires dans des images au format JPEG, au contraire de la modification de ses pixels.

2 Historique : stéganographie analogique

2.1 Sécurité par l'obscurité

La sécurité des systèmes décrits ici repose sur la non divulgation de la méthode de dissimulation utilisée. Une fois celle-ci connue, tout le monde est capable de lire le message secret.

L'Iliade d'Homère contient l'une des premières allusions à la dissimulation d'information. Banni pour avoir tué le tyran de Corinthe, Bellérophon s'exila chez Proétos, roi de Tirynthe, qui le purifia de son meurtre. Mais l'épouse de Proétos s'éprit du héros, qui la repoussa avec dédain. Dépitée, la reine l'accusa de tentative de viol : le roi la crut. Ne voulant toutefois pas tuer son hôte de sa main en raison des lois sacrées de l'hospitalité, il préféra envoyer le héros à Iobatès, son beau-frère, roi de Lycie, après avoir remis à Bellérophon des tablettes où était gravé, en signes mystérieux, l'ordre de lui donner la mort. Iobatès lui ordonna de combattre la Chimère, persuadé qu'il succomberait dans cette lutte. Monté sur Pégase, Bellérophon tua le monstre, épousa la fille du roi de Lycie et lui succéda.

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 message dans les partitions de musique : chaque note correspond à une lettre (substitution monoalphabétique simple, voir Fig. 1). Schott a également étendu le code "Ave Maria" proposé par l'abbé Trithème dans *Polygraphiæ* (1516), l'un des premiers livres connu 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. Chaqueă 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 stégo-texte ressemble à une prière, une simple lettre de correspondance, ou encore une formule de magie.

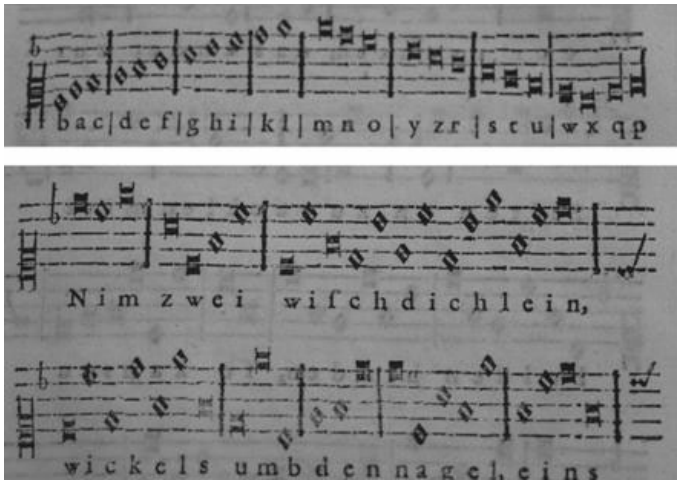

FIG. 1 – G. Schott utilise une simple correspondance entre lettres de l'alphabet et notes.

Une méthode très utilisée en stéganographie linguistique est l'acrostiche. Il s'agit d'un poème dans lequel les initiales de chaque vers, lues verticalement composent un mot clé. Par exemple, la dernière strophe du *Dormeur du val* de A. Rimbaud contient le mot "lit" :

Les parfums ne font pas frissonner sa narine ;
Il dort dans le soleil, la main sur sa poitrine,
Tranquille. Il a deux trous rouges au côté droit.

L'exemple le plus fameux est probablement l'*Amorosa visione* de Giovanni Boccaccio (1313-1375) qui semble être le plus long jamais écrit : Boccaccio écrivit d'abord trois sonnets contenant à eux trois approximativement 1500 lettres, puis écrivit un poème de telle façon que la première lettre des tercets successifs correspondît exactement aux lettres des sonnets.

Dans *The Code Breakers* [Kah96], D. Kahn raconte comment des prisonniers de guerre cachaient des message 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. Bien sûr, cette technique permet de dissimuler des messages, mais la construction du stégo-message est très laborieuse et peut parfois paraître bizarre pour un censeur expérimenté. Un autre exemple célèbre, issu du même livre, vient d'un télégramme envoyé pendant la première guerre mondiale. Le message, qui disait « père est mort », fut modifié en « père est décédé » par le censeur. La réponse fut un aveu flagrant : « père est-il mort ou décédé ? ».

2.2 Camouflage

L'utilisation du camouflage est un autre exemple intéressant de stéganographie. Même si une méthode est connue en principe, tirer partie de l'environnement et rendre difficile la recherche des données dissimulées s'avère parfois bénéfique, particulièrement lorsque les données de couverture sont très abondantes. L'adaptation moderne de ce principe est l'utilisation des propriétés de masquage de nos organes perceptifs.

Dans ses *Histoires*, Hérodote (486-425 av. J.C.) raconte comment vers 440 av. J.C., on rasa la tête d'un esclave, puis on y tatoua un message qui devint invisible après que les cheveux ont repoussé. Le but était de lancer une révolte contre les Perses. Étonnamment, la méthode était toujours utilisée par certains espions allemands au début du 20ème siècle. Hérodote raconte également comment Demeratus, un Grec à la court des Perses, avertit Sparte d'une invasion imminente de Xerxes, roi de Perse : il retira la cire d'une tablette d'écriture, écrivit le message sur le bois qui la supportait, et recouvrit à nouveau le bois, et donc le message, avec la cire. La tablette ressemblait exactement à une tablette vierge !

Énée 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 ou en dessous des lettres du message de couverture. Cette technique, toujours utilisée au 17ème siècle, fut améliorée par Wilkins qui utilisa des encres invisibles pour inscrire ces petits points au lieu de faire des trous. Cette dernière idée fut reprise par les espions allemands durant les deux guerres mondiales.

Une adaptation moderne est toujours utilisée pour sécuriser les documents électroniques et se contente d'imprimer des points minuscules (très facile grâce aux imprimantes laser modernes) pour coder un numéro d'identification d'imprimante, un nom d'utilisateur, etc.

En 1857, Brewster suggérait de cacher des messages secrets dans un espace pas plus grand qu'un point d'encre. En 1860 le problème de réduction des photographies était résolu par René Dragon (1819-1900). Pendant la guerre Franco-Prusse de 1870-1871, alors que Paris était assiégé, des messages sur microfilm étaient envoyés par pigeon entre la province et Paris. Pendant la première guerre mondiale, les messages des espions étaient réduits en micro-points grâce à plusieurs réductions photographiques. Ils étaient ensuite apposés sous forme de points ou point-virgule dans des messages de couverture aussi insoupçonnables que des magazines par exemple.

Les encres invisibles ont également été intensivement utilisées. Initialement, ces encres étaient fabriquées à partir de substances organiques, comme du lait, de l'urine, ou du sel d'ammoniac dissous dans de l'eau. Il s'agissait d'écrire en utilisant cette encre. Une exposition du papier de couverture devant une flamme révélait alors le message écrit. Par la suite, des progrès en chimie ont permis d'élaborer des encres plus efficaces. Actuellement, une telle technique est utilisée pour prévenir la contrefaçon de billets de banque. Par exemple, les photocopieurs émettent un fort rayonnement ultra-violet. 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 ultra-violets.

Les techniques de stéganographie ne servent pas uniquement à transmettre des messages ou empêcher la fabrication de contrefaçon. On raconte que dans les années 1980, Margaret Thatcher était ennuyée car des documents du gouvernement apparaissaient dans la presse. Afin d'identifier la source de la fuite, elle fit modifier les traitement de texte de ses ministres afin que l'espacement des mots soit caractéristique de chacun d'eux. Cette manipulation lui permit de déterminer l'origine de la fuite.

2.3 Leçon du passé

En 1883, Auguste Kerckhoffs a énoncé le premier principe de la cryptographie moderne. Il affirma qu'il fallait supposer connue de l'ennemi la méthode de chiffrement et que, par conséquent, la sécurité du système ne devait résider que dans le choix de la clé utilisée pour le chiffrement. Les évènements sont nombreux pour lesquels la *sécurité par l'obscurité* fut un échec. Un des derniers exemples nous est fourni par le cas des téléphones mobiles GSM.

En appliquant ce principe à la stéganographie, on obtient une tentative de définition d'un stégo-système sécurisé. Il faut qu'un adversaire, connaissant le système mais pas la clé, ne puisse obtenir aucun indice sur la tenue d'une éventuelle communication. Aucune information sur le texte dissimulé ou l'algorithme qui aurait pu être utilisé ne doit donc être obtenue à partir du médium de couverture. Cette règle constitue le principe central de tout stégo-système.

3 Stéganographie numérique

L'univers informatique recèle de nombreuses ressources pour dissimuler des informations. Nous présentons maintenant différentes solutions.

3.1 Stéganographie à l'aide d'un texte

Espacement entre les mots

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

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

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

et celle-ci

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

La superposition des deux met en relief une phrase :

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

Méthode des synonymes

Avec cette méthode, une table de synonymes est construite :

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

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

D'autres propositions ont été faites pour dissimuler le message dans les balises du langage. Par exemple, le HTML se prête parfaitement à ce type de manipulations : le texte qui apparaît sur l'écran est mis en forme à l'aide de balises, mais toutes ne sont pas intégralement interprétées. Ainsi, le code de déchiffrement des DVDs est présent dans la page http://decss.zoy.org/ de S. Hocevar, dissimulé sous forme de commentaires[3]. Il est alors possible de le visualiser en regardant les sources de la page :

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

...

void CSSdescramble(unsigned char *sec,unsigned char *key) {
  unsigned int t1,t2,t3,t4,t5,t6;
  unsigned char *end=sec+0x800;
  t1=key[0]^sec[0x54]|0x100;
  t2=key[1]^sec[0x55];
  t3=(*((unsigned int *)(key+2)))^(*((unsigned int *)(sec+0x56)));
  t4=t3&7;
  t3=t3*2+8-t4;
  sec+=0x80;
  t5=0;
  while(sec!=end) {
        t4=CSSt2[t2]^CSSt3[t1];
        t2=t1>>1;
        t1=((t1&1)<<8)^t4;
        t4=CSSt5[t4];
        t6=(((((((t3>>3)^t3)>>1)^t3)>>8)^t3)>>5)&0xff;
        t3=(t3<<8)|t6;
        t6=CSSt4[t6];
        t5+=t6+t4;
        *sec++=CSSt1[*sec]^(t5&0xff);
        t5>>=8;
```

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

```
  }
}
...

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

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

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

Utilisation de grammaires

Wayner propose dans [Way92, Way95] une solution plus robuste fondée sur l'emploi de grammaires algébriques [4] pour la génération de texte.

Les règles de production sont déterminées puis pondérées par une probabilité. On construit ainsi des arbres identiques à ceux utilisés dans l'algorithme de compression de Huffman. Les règles données respectivement dans la figure 2 pour le sujet S, 3 pour le verbe V et 4 pour l'attribut A permettent de construire des phrases à partir de l'axiome S en fonction de la chaîne binaire à dissimuler.

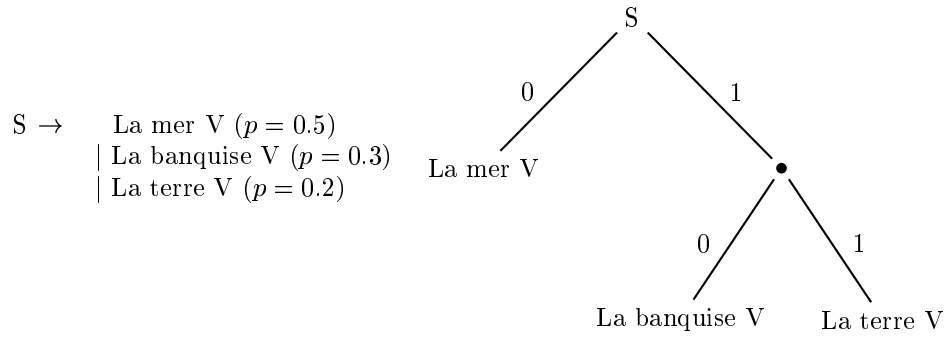

FIG. 2 – Règle de dérivation pour le sujet S et arbre de Huffman associé

Par exemple, si Alice souhaite envoyer à Bob la chaîne `11011`, elle construit une phrase avec sujet (S), verbe (V) et attribut (A) à partir des arbres prédéfinis : le `11` donne «La terre» dans l'arbre S du sujet, le `0` le verbe «devient» puis finalement «blanche.» pour l'attribut. Elle envoie donc la phrase «La terre devient blanche.» à Bob. Il reconstruit la chaîne binaire à l'aide des arbres. Si la grammaire n'est pas ambiguë, cette décomposition est alors unique.

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

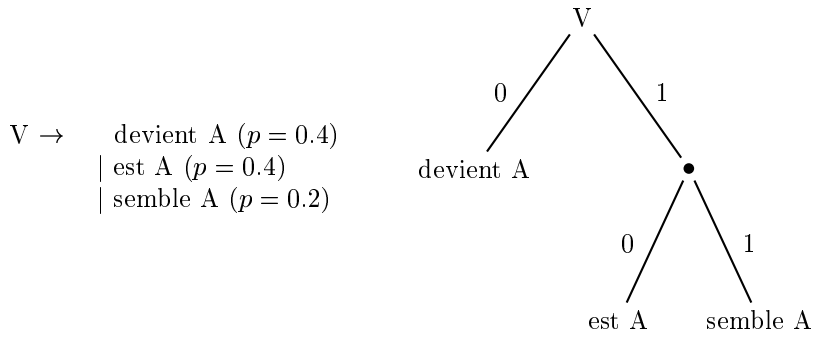

FIG. 3 – Règle de dérivation pour le verbe V et arbre de Huffman associé

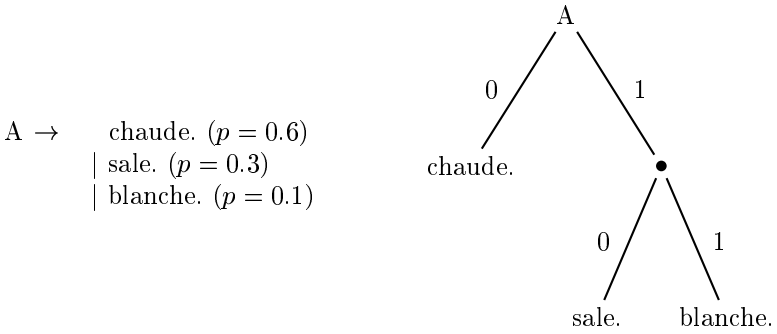

FIG. 4 – Règle de dérivation pour l'attribut A et arbre de Huffman associé

3.1.1 Stéganographie à l'aide d'un médium

Fred : je mettrais bien un truc ici sur le remplacement des derniers bits d'une image (avec exemples comme sur ta page Fabien)

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

1. la clé secrète servant à la dissimulation n'est pas connue de l'adversaire ;
2. le médium initial n'est pas disponible pour l'adversaire, ce qui est facilement réalisable à l'aide d'un appareil photo numérique ou d'un scanner.

Ainsi, pour les images, le logiciel `outguess`[5] de N. Provos [Pro01] propose deux méthodes pour dissimuler des données dans une image.

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

L'auteur distingue deux étapes dans l'insertion :

1. l'identification des bits redondants[6] car ils sont potentiellement modifiables, sans endommager l'image initiale ;
2. la sélection des bits dans lesquels l'information sera placée.

Cette séparation offre l'avantage de pouvoir travailler indépendamment d'un type de médium donné. De plus, il est ainsi facile de modifier l'une ou l'autre des deux étapes afin de mesurer la pertinence de ce changement. En revanche,

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

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

l'inconvénient vient de la perte d'information entre les deux étapes. En effet, le résultat de l'identification est ensuite transmis à la sélection : celle-ci s'effectue donc sans connaissance de la provenance des bits dans l'image et ne tient donc pas compte du fait qu'une région est peut-être moins sensible qu'une autre à des altérations. Les bits sélectionnés sont modifiés pour correspondre à ceux du message.

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

3.1.2 Stéganographie à l'aide d'autres supports

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

Fichier binaire exécutable

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

En particulier, une de ces sections comporte les instructions en langage machine. Par exemple, le tableau 1 présente un petit programme et le contenu de la section `.text` qui renferme, sous Linux, les instructions en langage machine. On constate sur cet exemple très simple que de nombreux octets ne servent à rien : les `NOP` signifient *no operation*. Cet espace est donc «perdu» puisque les instructions n'engendrent aucune opération. Il est alors possible de remplacer ces octets par ceux de notre choix, sous réserve qu'à aucun moment le registre d'instructions[7] ne viennent dans cette zone (on parle alors de *code mort*). En effet, les octets placés ont peu de chance de correspondre à une instruction valide et provoqueraient alors un échec du programme. L'emploi d'autres sections est également envisageable.

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

Système de fichiers

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

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

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

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

TAB. 1 – Programme C et sa version Assembleur dont les instructions sont écrites dans la section `.text`

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

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

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

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

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

StegFS, proposé par McDonald et Kuhn dans [MK99], s'inspire de cette seconde

solution. Ce système de fichiers ne crée pas de partition particulière, mais place ses données dans les blocs inutilisés du système de fichiers ext2[8]. Pour cela, une table d'allocation des blocs est créée exactement comme avec un système de fichiers normal, à la vue de tous ; elle contient les adresses des blocs qui dissimulent les informations. Cette table révèle que StegFS est installé sur la machine, mais d'autres traces font de même, comme la présence des outils nécessaires à son utilisation[9]. Si un attaquant est capable de constater que ce système de fichiers est présent sur une machine, il lui est néanmoins impossible de savoir s'il est utilisé pour dissimuler des informations.

4 Conclusion

Les moyens permettant de cacher de l'information sont, comme nous l'avons vu, assez divers ; il en existe encore beaucoup d'autres que nous n'avons pu présenter ici, y compris lorsqu'on quitte l'univers numérique de l'informatique. De même, les objectifs de cette dissimulation sont amenés à changer selon le contexte et les applications. On peut évidemment penser à des contextes d'espionnage, mais n'oublions pas qu'ils pourraient être utilisés sous forme de "smart images" (images contenant une information cachée, cette information donnant accès à des services, comme la visite de sites web, par exemple), ou encore pour protéger les droits d'auteurs (cf. l'article sur le tatouage).

Références

[ANS98] Anderson (R.), Needham et Shamir (A.). – The steganographic file system. *In : IWIH : International Workshop on Information Hiding.* – 1998.

[Bla93] Blaze (M.). – A cryptographic file system for UNIX. *In : ACM Conference on Computer and Communications Security*, pp. 9–16. – 1993.

[BLM99] Brassil (J. T.), Low (S.) et Maxemchuk (N. F.). – Copyright protection for electronic distribution of text documents. *In : Proceedings of the IEEE (USA)*, pp. 1181–1196. – 1999.

[CP98] Cattaneo (G.) et Persiano (G.). – *TCFS - Transparent Cryptographic File System.* – Rapport de recherche, DIA, Universita Degli Studi Di Salerno, 1998. http://tcfs.dia.unisa.it.

[CTL98] Collberg (C.), Thomborson (C.) et Low (D.). – Manufactiruing cheap resilient and stealthy opaque constructs. *In : Proceedings of the ACM Symposium on the Principles of Programming Languages*, pp. 194–196. – 1998.

[EFS98] Encrypting file system for windows 2000, 1998. http://www.microsoft.com/windows/server/Technical/security/encrypt.asp.

[Gut96] Gutmann (P.). – Secure filesystem (SFS) for DOS/Windows, 1996. http://www.cs.auckland.ac.nz/~pgut001/sfs/.

[Kah96] Kahn (D.). – *The codebreakers.* – Scribner, 1996.

[ker] The international kernel patch. http://www.kerneli.org/.

[KP99] Katzenbeisser (S.) et Petitcolas (F.) (édité par). – *Information hiding : techniques for steganography and digital watermarking.* – Artech House Books, 1999.

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

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

[MK99] McDonald (A. D.) et Kuhn (M. G.). – Stegfs : A steganographic file system for linux. *In : International Workshop on Information Hiding*, pp. 462–477. – 1999.

[Pro01] Provos (N.). – Defending against statistical steganalysis. *In : Proceedings of the 10th USENIX Security Symposium*. – Washington DC, USA, août 2001.

[Sim84] Simmons (G. J.). – The prisoners' problem and the subliminal channel. *In : Advances in Cryptology*. CRYPTO '83, pp. 51–67. – Plenum Press, 1984.

[Way92] Wayner (P.). – Mimic functions. *Cryptologia*, vol. 16, 1992, pp. 193–214.

[Way95] Wayner (P.). – Strong theoretical steganography. *Cryptologia*, vol. 19, 1995, pp. 285–299.

[ZFK+98] Zöllner (J.), Federath (H.), Klimant (H.), Pfitzmann (A.), Piotraschke (R.), Westfeld (A.), Wicke (G.) et Wolf (G.). – Modelling the security of steganographic systems. *In : IWIH : International Workshop on Information Hiding*. – 1998.