Stéganographie : Bitmap et GIF

Foud Hind et Patti Philippe

3 avril 2020

Table des matières

1 Introduction

L'objectif du projet est d'implémenter un programme employant la stéganographie.

Dans le cadre du cours de Systèmes, la stéganographie nous permet d'approfondir l'utilisation de la mémoire pour stocker des informations. Nous devons apprendre en détail la structure des fichiers bitmaps et GIF pour pouvoir y cacher des messages. De plus, ce sujet nous permet d'approfondir le thème de la sécurité, un thème qui est fort lié aux systèmes d'exploitations.

Nous avons choisi d'implémenter 2 programmes en C qui cacheront un message à l'intérieur d'un bitmap et d'un GIF. Dans ce rapport, nous expliquerons pour chaque format la raison de notre choix, la technique utilisée, l'avancement du projet et un guide pour utiliser les executables.

2 Stéganographie : Définition

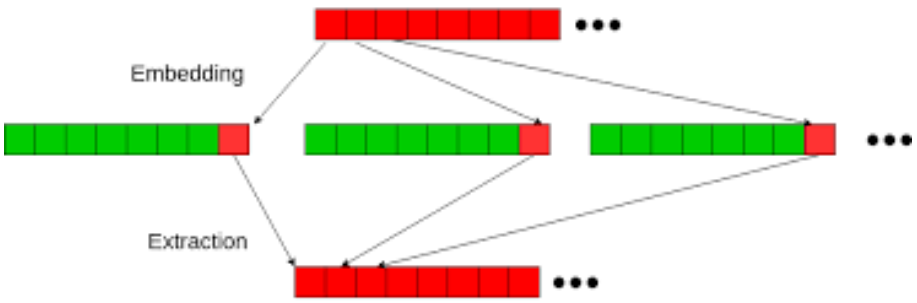
La stéganographie est l'art de la dissimulation : son objet est de faire passer inaperçu un message dans un autre message. Elle se distingue de la cryptographie, « art du secret », qui cherche à rendre un message inintelligible à autre que qui-de-droit.

3 Technique utilisée

Pour cacher un message, nous employons la technique Least Significant Bit (LSB).

Comme vous le voyez sur le schéma ci-dessous, cette technique consiste à caché un bit d'information dans le bit le moins important d'un byte. Pour décripter le message, il suffit de lire le dernier bit de ces bytes et de reformer un message compréhensible.

Pour que le message ait le plus petit impact possible sur le fichier source, il faut employer des bytes qui peuvent perdre un bit d'information. Par exemple, pour une couleur codée en rgb, le changement du dernier bit a un impact très faible, quasi invisible à l'oeil nu.



4 Bitmap

4.1 Pourquoi le bitmap ?

4.2 Structure

4.3 Application du LSB

4.4 Avancement

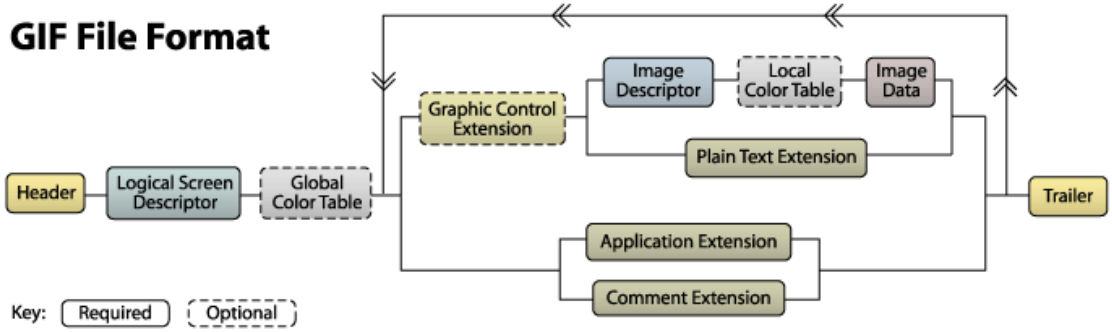
4.5 Guide pratique

5 GIF

5.1 Pourquoi le GIF ?

Le format GIF permet de stocker plusieurs images dans un fichier. Ceci permet de créer des diaporamas, voire des animations si les images sont affichées à un rythme suffisamment soutenu. Chaque image d'une animation peut avoir sa propre palette. Le GIF est un format très utilisé, particulièrement sur les réseaux sociaux. Ce projet peut donc intéressé pas mal de gens.

La structure du format GIF est connue et disponible en ligne. En voici un schéma :



5.2 Application du LSB

Nous avons choisi de cacher des informations dans les couleurs qui se trouvent dans les Local Color Table. Cette section facultative peut revenir devant chaque bloc image data. Si cette section n'existe pas, nous la rajoutons.

Il est possible de calculer la taille maximale du message cachée à l'avance, en comptant le nombre de bloc image data et en le multipliant par le nombre de byte dans la Global Color Table.

5.3 Avancement

5.3.1 Lecture

Le programme peut déja lire un gif en entier, section par section. Les tailles des sections sont indiquées à des endroits différents par sections. La lecture se fait donc différemment en fonction de la section.

De plus, on peut déja savoir le nombre maximal de Local Color Table pour le fichier modifié.

5.3.2 Lecture

Le programme réécrit le gif dans un nouveau fichier, en insérant des Local Color Table, identique à la Global Color Table. Ceci permettra d'y insérer un message. L'insertion du message devrait être évidente, vue qu'elle sera presque identique à la stéganographie dans un fichier bitmap.

Il y a toutefois encore un bug dans la réécriture du fichier gif, probablement à l'écriture des nouvelles Local Color Table.

5.4 Guide pratique

Pour exécuter le projet actuel, exécuter le makefile à la racine du fichier. Voici quelques options possibles avec le makefile :

```
#NOM : Makefile
#CLASSE : SYSG5
```

```
#OBJET : Steganographie avec Bitmap et GIF
#AUTEUR : Foud Hind et Patti Philippe
#HOWTO : make ; make run_gif; make build_gif; make clean

# make                : execute les demos
# make run            : execute les demos
# make build          : compile les demos
# make clean          : supprime les fichiers generes

# make run_bmp        : execute uniquement la demo pour le format bmp
# make build_bmp      : compile uniquement la demo pour le format bmp
# make clean_bmp      : supprime uniquement les fichiers generes lies au format bmp

# make run_gif        : execute uniquement la demo pour le format GIF
# make build_gif      : compile uniquement la demo pour le format GIF
# make clean_gif      : supprime uniquement les fichiers generes lies au format GIF
```

6 exemples latex

Vous découpez votre travail en chapitre (section), en sous-chapitre (subsection) et en sous-sous-chapitre (subsubsection) Il suffit de taper le texte au kilomètre. *Il est possible d'insister* sur un passage. Vous appelez la commande :

```
aspell -t - -encoding='iso8859-15' -c VotreFichier.tex
```

6.0.1 Une énumération

Si vous devez utiliser des puces :
- point 1
- point 2

Si vous devez énumérer :

1. point 1
2. point 2

Vous pouvez mélanger à autant de niveaux que vous le souhaitez.

6.0.2 Un tableau

Jour	Heures	Local	Cours
Mardi	3..4	503	Système
Mardi	5..8	503	Sécurité
Mercredi	1..2	201	Assembleur
Mercredi	3..4	003	Labo assembleur

6.0.3 Intégrer une source

Pour énumérer une source, vous pouvez préciser ce que vous souhaitez avec lstset.

```
MOV EAX,10 ; place 10 dans le registre EAX
ADD EAX,20 ; ajoute 20 au contenu de EAX
```

6.0.4 Intégrer un graphique

Vous pouvez intégrer un graphique mais il faut qu'il soit en format eps.
Vous pouvez dessiner avec l'outil oodraw qui permet d'exporter votre dessin dans ce format eps.
Vous pouvez transformer un jpeg en eps avec l'outil sam2p ou convert qui se trouve dans le package ImageMagick

6.0.5 Mode mathématique

C'est un des points forts de latex qui permet d'écrire des formules mais aussi des caractères spéciaux tels que : $2^{32} \approx 10^9$ car $\log_{10} 2 \approx 0.3$ et $32 * 0.3 \approx 9$.

6.0.6 Quelques trucs faciles

- \\ permet de passer à la ligne suivante.
- \\[2cm] permet de passer à la ligne suivante + une tabulation verticale de 2cm. Sont autorisés mm, cm et pts.
- \newpage permet de forcer un passage à la page suivante.

6.0.7 Compiler le texte

Un script permet d'automatiser cette compilation :

```
#! /bin/sh
FN=Mrapport # Le nom du document.
latex $FN.tex
latex $FN.tex # 2 passages pour la TOC
rm $FN.aux $FN.log $FN.out
dvips $FN.dvi -o $FN.ps
rm $FN.dvi
gv $FN.ps # pour visualiser et imprimer
```

6.1 Conclusions

Ce travail montre, qu'en quelques minutes, on peut déjà fournir un travail présenté de façon professionnelle, lisible par tous et dans un format standard. Pour ne pas avoir de soucis, les commandes à utiliser pour obtenir un document postcript sont latex et dvips, il ne faut jamais utiliser pdflatex.
Ce document peut être encore complété avec d'autres exemples qui seraient utiles. Ces nouveaux exemples pourraient être intégrés dans le premier ou un deuxième chapitre. Ceci, sans oublier qu'il s'agit d'un document utile pour commencer très rapidement à écrire en latex et non un mode d'emploi complet de latex qui serait obligatoirement très volumineux.

6.2 Références

— http ://www.grappa.univ-lille3.fr/FAQ-LaTeX/
— http ://tex.loria.fr/
— http ://tex.loria.fr/english/packages.html

6.3 Annexes

Vous trouvez dans le casier, un répertoire LATEX contenant :
— Mrapport.tex : le document latex maître
— LatexSimple.tex : ce document latex
— dessin.odg et dessin.eps : le dessin intégré dans le texte
— go : un script qui permet de compiler rapport.tex, sans argument.

7 Conclusion