Rapport SYSG5 : Steganographie

Noé DELCROIX - G55990

November 2022

Contents

1 Introduction

Dans ce rapport, je vais vous présenter le résultat de mon travail de recherche au sujet de la stéganographie et plus précisément les techniques less significant bits (LSB) et zero-width characters.
La stéganographie est un ensemble de techniques permettant de cacher des données au sein d'un autre fichier. La stéganographie n'a rien de nouveau, peut s'appliquer sur plusieurs supports que ce soit des documents physiques (papier, pierre, photos, etc.) ou informatisé (images numériques, fichiers textes, etc.).
dans ce rapport, je vais présenter deux techniques. L'une permettant de stocker des données dans les pixels que compose une image PNG (ou tout autre format d'image n'utilisant pas de compression et représentant ces pixels en true color/RVB), l'autre permet de stocker n'importe quelle donnée dans un fichier texte ou tout autre format ne vérifiant pas l'intégrité de ses données.

2 LSB

La technique de stéganographie LSB, pour least significant bit, est une méthode pour cacher des données que ce soit du texte, une image ou tout autre fichier dans une image en utilisant le bit de poids faible des octets représentant chaque pixel.
Cette technique ne comporte pas beaucoup de limitations ci ce n'est d'avoir une donnée à cacher d'une taille maximale en bits de 3/8 du nombre de pixels composant l'image au format PNG-8 (j'explique à la section x.x quelles en sont les raisons).

2.1 Format PNG

Pour illustrer cette technique de stéganographie, je vais utiliser le format d'image PNG avec une taille de canal de couleur sur 8 bits.

2.1.1 Qu'est-ce que le format PNG ?

Le format PNG a été créé dans le but de compenser les limitations du format GIF (format propriétaire) tout en le rendant beaucoup plus simple à implémenter. Il représente ses pixels sous forme matriciel avec 3 canaux de couleurs RVB.

2.1.2 Chunks

Le format PNG est composé d'un magic number et de multiples chunks. Un chunk est une structure permettant de décrire les différentes données d'un fichier PNG. Chaque chunk est composé de 4 parties :

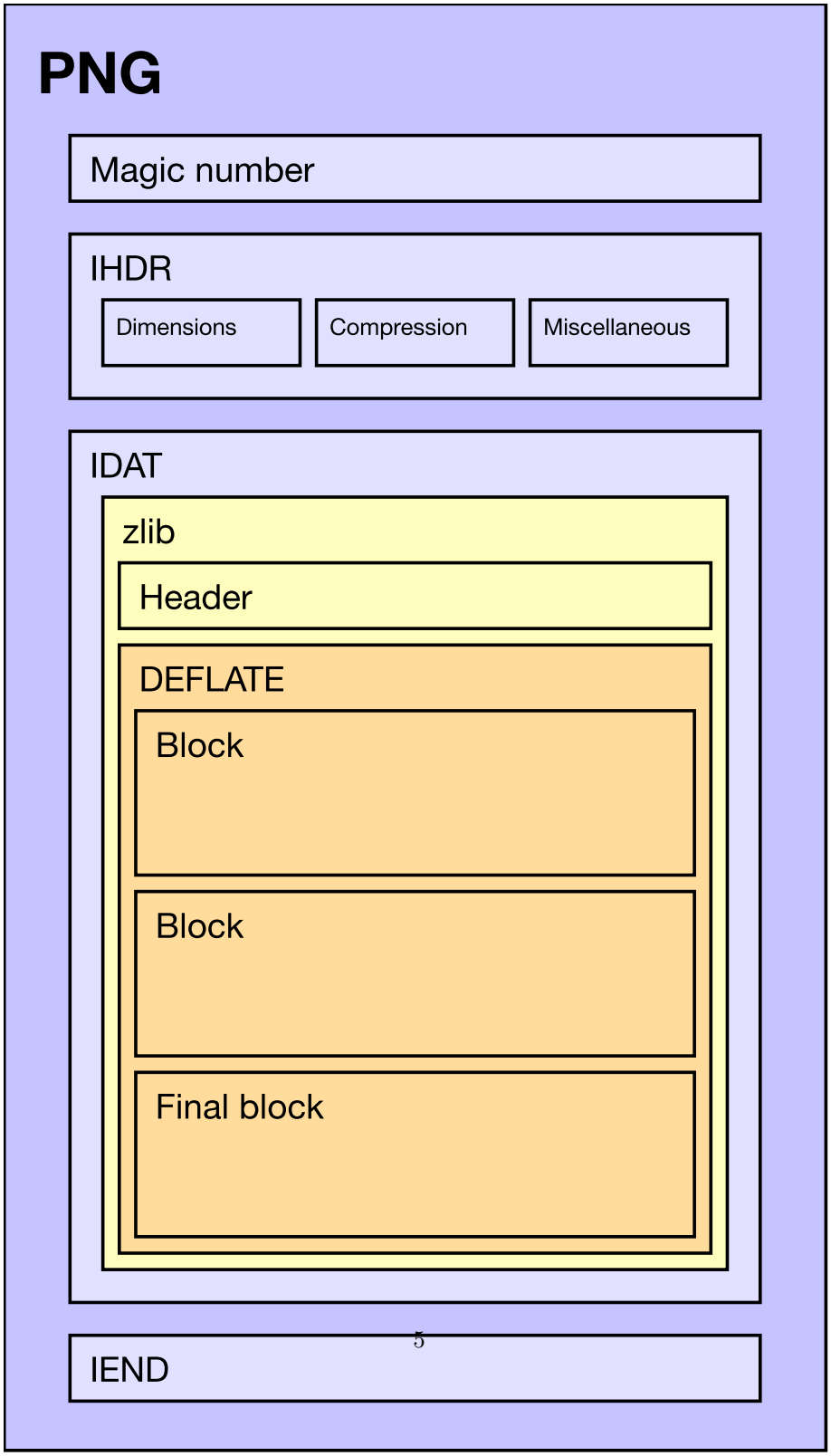
2.1.2.1 Longueur Sur 4 octets et peut être 0, spécifie la taille que prend la section données du chunk.

2.1.2.2 Type Sur 4 octets, spécifie le type du chunk (exemple : IDAT, IEND, etc.).

2.1.2.3 Données De la taille spécifiée dans la section longueur, contient les données de ce chunk.

2.1.2.4 CRC Checksum de l'intégrité de la section données.

2.2 Structure du format PNG



2.2.1 PNG file signature/magic number

0	89 50 4e 47 0d 0a 1a 0a	• Special: File signature • Length: 8 bytes	• "□PNG␍␊␚␊"

Suite de 8 octets permettant de reconnaitre le format de fichier PNG. Cette suite est celle-ci : 137 80 78 71 13 10 26 10

2.2.2 IHDR image header chunk

8	00 00 00 0d 49 48 44 52 00 00 01 2c 00 00 00 f0 08 02 00 00 00 35 94 ce c2	• Data length: 13 bytes • Type: IHDR • Name: Image header • Critical (0) • Public (0) • Reserved (0) • Unsafe to copy (0) • CRC-32: 3594CEC2	• Width: 300 pixels • Height: 240 pixels • Bit depth: 8 bits per channel • Color type: RGB (2) • Compression method: DEFLATE (0) • Filter method: Adaptive (0) • Interlace method: None (0)

Chunk indispensable et doit être en premier qui contient les méta données du fichier PNG tels que la largeur, la longueur, la taille d'un canal de couleur (8 bits en PNG-8), méthode de compression, etc.

2.2.3 IDAT chunk (Image data)

111	00 00 4f 85 49 44 41 54 78 da ed bd 59 94 5c c7 79 26 f8 fd 7f c4 bd b9 d5 0e 14 76 80 04 09 52 5c c0 45 a4 48 4a 94 65 89 6e 5b 94 45 db 92 77 f7 39 ea f6 d6 9e ee e9 33 0f d3 f3 d0 d3 0f 33 f3 3c 67 c6 ef ad ... 23 27 61 8e 1c 3b 8c 9c 84 39 72 ec 30 72 12 e6 c8 b1 c3 f8 ff 01 1b 21 8a 38 ee 4d 21 9d	• Data length: 20 357 bytes • Type: IDAT • Name: Image data • Critical (0) • Public (0) • Reserved (0) • Unsafe to copy (0) • CRC-32: EE4D219D	

Chunk contenant les pixels de l'image. Il peut y avoir plusieurs chunk IDAT et peut être compressé si un algorithme de compression est mentionné dans le chunk IHDR. C'est principalement ce chunk que nous allons utiliser.

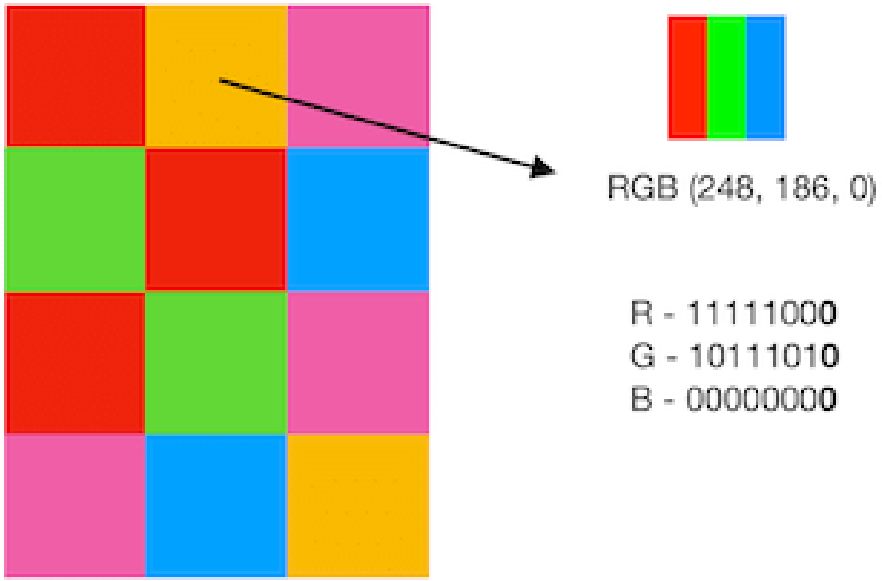
2.2.4 IEND chunk

20 578	00 00 00 00 49 45 4e 44 ae 42 60 82	• Data length: 0 bytes • Type: IEND • Name: Image trailer • Critical (0) • Public (0) • Reserved (0) • Unsafe to copy (0) • CRC-32: AE426082	

Chunk marquant la fin du fichier PNG. Il doit être placé à la fin et contient une section data vide.
D'autres chunks peuvent exister pour contenir d'autres informations facultatives

tels que la transparence, les méta données (date de création, modification, etc.), la pallette de couleurs, etc.

2.3 Structure du segment IDAT



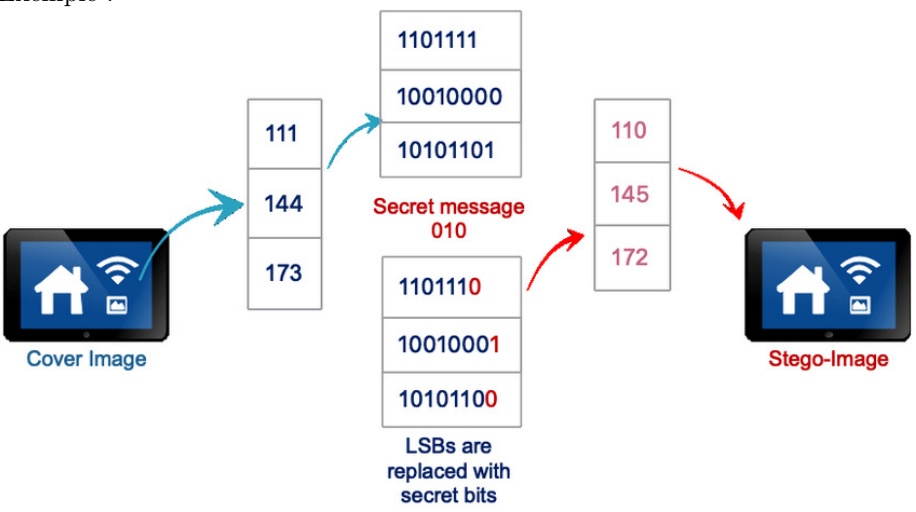
Ici, je vais considérer que nous utillisons une image PNG sans compression où les pixels sont représentés par le type true color. Deux autres type existent (grayscale et indexed color) mais je ne m'attarderai pas dessus.
Dans la section données du segment IDAT, les pixels sont allignés l'un à la suite de l'autre. Chacun des pixels est composé de 3 canaux (rouge, vert et bleu) pour représenter la couleur de ce pixel. Chaque ligne de pixels est d'une longeur définie par le segment IHDR.
Un pixel d'une image PNG-8 a donc une taille de 3 octets et permet d'avoir 256^3 couleurs différentes.

2.4 LSB sur PNG

L'idée de la stéganographie least significant bit est d'utiliser le bit de poids faible des canaux d'un pixel pour insérer des données.
Voici un exemple pour mieux comprendre :

Ici, nous avons 2 couleurs. Une couleur de référence (à gauche) et une couleur où j'ai inversé le bit de poids faible des 3 canaux de couleur.
La différence est (quasi) invisible à l'oeil nu et le fait qu'un pixel soit de taille très petite rend l'impossibilité de détecter la différence de couleur, en tant qu'humain.

L'idée est donc de stocker des données binaires dans ces bits de poids faible. Exemple :



3 LSB en pratique

4 Cacher du texte dans un fichier texte

Je vais maintenant parler d'une autre technique de stéganographie. Cette technique de stéganographie permet de cacher des données au sein d'un fichier texte, au moyen de caractères dits zero-width.
Cette technique n'a pas de limitations au niveau de la taille à insérer mais augmente la taille du fichier initial (8 fois la taille à insérer).

4.1 Caractères zero-width

Les caractères zero-width sont des caractères unicodes qui ne prennent pas de place dans le texte. Ils sont donc invisibles au lecteur du fichier.

4.2 Fonctionnement

Cette technique convertit la valeur binaire de la donnée à insérer en caractères zero-width. On utilise deux caractères zero-width au choix (pour la démo et l'exemple j'utiliserai les caractères zero-width space et zero-width non joiner). Un des deux caractères zero width va représenter la valeur 1 binaire et l'autre caractère va représenter la valeur 0 binaire.

Unicode Number	UTF-16 decimal code	Description
U+180E	6158	mongolian vowel separator
U+200B	8203	zero width space
U+200C	8204	zero width non-joiner
U+200D	8205	zero width joiner
U+200E	8206	left-to-right mark
U+200F	8207	right-to-left mark
U+FEFF	65279	zero width no-break

Voici un exemple avec les caractères zero width space (ZWSP) et zero width non-joiner (ZWNJ) :
donnée : 'h' valeur binaire : 01101000

Si l'on considère le caractère ZWSP comme valant 0 et ZWNJ comme valant 1 :

0	ZWSP
1	ZWNJ
1	ZWNJ
0	ZWSP
1	ZWNJ
0	ZWSP
0	ZWSP
0	ZWSP

On ajoute donc les caractères ZWSP ZWNJ ZWNJ ZWSP ZWNJ ZWSP ZWSP ZWSP à la fin du fichier, ceux-ci représentant les bits de la données insérée et n'étant pas vu par l'humain.

5 Cacher du texte dans un fichier texte en pratique

6 Conclusion

7 Références