# Rapport Stéganographie et Tatouage d’images

Pour les questions sur les algorithmes, voir le code C++ (Visual Studio)

Exercice 2 :

1. Un unsigned char correspond à 8 bits de données, si on place 2 bits par pixel, et que la taille de l’image est de N\*N, on peut donc placer N\*N/4 caractères.
2. Comme il faut 4 pixels pour écrire 1 caractère, si on veut placer nc caractères dans un carré de côté a pixels, on obtient la relation ci-dessous grâce à la première question : nc = a²/4

Donc, a = 2\*sqrt(nc)

1. 0 <= k <= N – 2\*a

0 <= k <= N – 4\*sqrt(nc)

Algorithme détaillé :

Je vais parler ici d’un exercice du TP.

Le but de cet algorithme est de caché une image en niveau de gris dans une image de couleur. Pour ceci, on découpe chaque pixel de l’image en niveau de gris qui se trouve sur 8 bits, en 3 parties, 2 de 3 bits et 1 de 2 bits.

Pour que cette image paraisse invisible à l’œil nul, on va modifier les bits de poids faibles des couleurs rouges, vertes et bleues respectivement par rapport au découpage du pixel de l’image en niveau de gris.

Tout d’abord, on vérifie si l’image en niveau de gris est de même taille ou plus petite que l’image de couleur.

Si c’est le cas, on continue notre algorithme.

On va ensuite parcourir l’image en longueur et en hauteur.

Pendant ce parcours, à chaque pixel, on va effectuer une opération.

Cette opération consiste à prendre le pixel de l’image en niveau de gris, à le découper en 3 parties comme dit précédemment, et on va ensuite remplacer les 3 bits de poids faibles de la couleur rouge par la première partie, les 3 bits de poids faibles de la couleur verte par la deuxième partie et enfin les 2 bits de poids faibles de la couleur bleue.

Une fois cela fini, on passe au pixel suivant de l’image de niveau de gris et de l’image de couleur.

L’opération inverse est aisée si on connait la taille de l’image de niveau de gris à récupérer.

On parcourt l’image en longueur et en hauteur, et à chaque pixel on récupère les 3 bits de poids faibles de la couleur rouge, les 3 bits de poids faibles de la couleur verte et les 3 bits de poids faibles de la couleur bleue.

On les concatène pour ensuite obtenir le pixel de l’image de niveau de gris.

Au final, on obtient l’image originale sans aucune perte.

J’ai choisi cet algorithme parce que c’est celui qui m’a le plus plu, et qui est probablement le plus intéressant à utiliser en cas de stéganographie amateure. On peut facile avoir des images en niveau de gris avec n’importe quel appareil photo, il suffit de donner une bonne taille à cette image pour qu’elle puisse rentrer dans une autre image et on peut envoyer n’importe quel message aux personnes souhaitées et uniquement ces personnes.

Le fait est que l’image de couleur n’a absolument aucune différence à l’œil nul entre l’originale et la photo modifiée.

État de l’art de la Stéganographie :

La Stéganographie est utilisée pour tout ce qui est watermark dans les images et les vidéos.

Plus particulièrement, le watermarking visible est extrêmement utilisé chez les artistes pour garder une jolie image tout en gardant la source de l’image. (Voir Annexe 1)

Cependant, le watermarking de type invisible est plus rarement utilisé. Souvent, le watermarking visible est utilisé pour « gâcher » une image pour qu’elle ne soit pas utiliser impunément. (Voir Annexe 2)

Pour ce qui est des vidéos, on voit souvent une watermark visible de la chaîne télé/site diffusant la vidéo. (Voir Annexe 3)

État de l’art de la Stéganalyse :

La Stéganalyse est l’analyse des images ou des vidéos cachant des messages ou même des fichiers entiers.

Dans un premier temps, le but est d’analyser si une image cache un message. (Exemple Annexe 4)

On peut ensuite essayer de trouver un moyen de décoder le message.

Cette science est très proche de la cryptanalyse dans la cryptographie.

Cette science est beaucoup plus utilisée dans la sécurité, souvent par l’armée pour les sujets sensibles.

Annexe 1 :



Annexe 2 :



Annexe 3 :



Annexe 4 :

Ici, on peut voir une image qui n’a pas été modifié, bien sûr, à l’œil nu c’est invisible.



Mais si on change les nombres impairs de l’image en 255 et les nombres pairs de l’image en 0, on obtient ceci :



On remarque une non-uniformité des couleurs.

Ici, c’est la même image modifié, toujours rien de visible à l’œil nu.



Pourtant, avec le même procédé qu’avant, on remarque que les couleurs sont beaucoup plus uniformes. On en déduit qu’il y a un message caché dans cette image.



Reste à savoir ensuite comment décoder le message…