HAI918I - Image, Sécurité et Deep Learning
TP3

Ingo Diab

2023

Table des matières

1 Plans binaires

A partir de masque binaire et des opérateur logique, nous pouvons récupérer seulement un bit par valeur.

```
unsigned char _randBitMask = 1 << _plan;

for(int i = 0; i<imIn.getWidth(); ++i)
    for(int j = 0; j<imIn.getHeight(); ++j)
    {
        imOut[i][j] &= _randBitMask;
        if(imOut[i][j] != 0) imOut[i][j] = 255;
    }
```

FIGURE 1 – Code pour récupérer un plan

A partir de l'image de référence, nous pouvons ainsi extraire chacun des 8 plans binaire.

FIGURE 2 – Image référence

On peut remarquer que le plan MSB ressemble bien plus à l'image référence que le plan LSB. En effet, les bits de poids fort contiennent plus d'informations que les bits de poids faible. Enlever les bits de poids forts revient donc à perdre énormément d'information de l'image d'origine.

Nous avons aussi effectuer ceci sur une image chiffrée.

Nous remarquons ici que, peut importe le plan choisi, l'image ressemble à du bruit, ce qui est normal puisque l'image de référence ressemble aussi à du bruit.

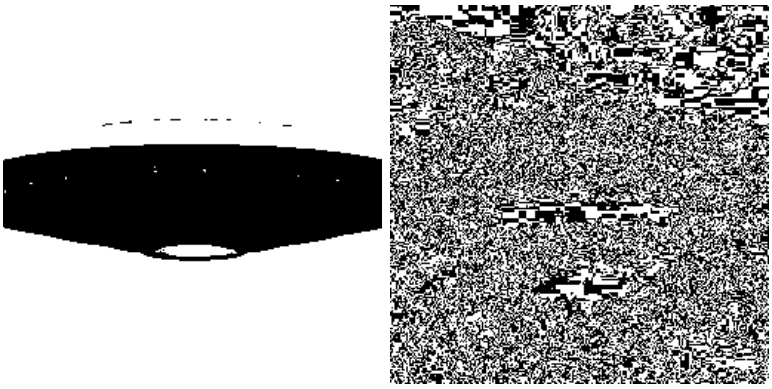

FIGURE 3 – A gauche le plan MSB, à droite le plan LSB

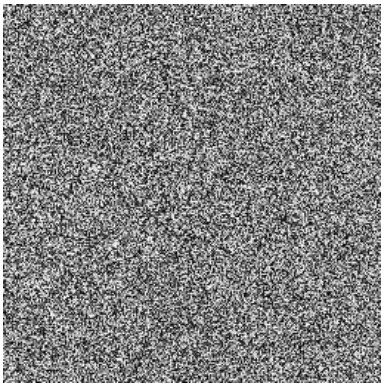

FIGURE 4 – Image référence chiffrée avec OFB

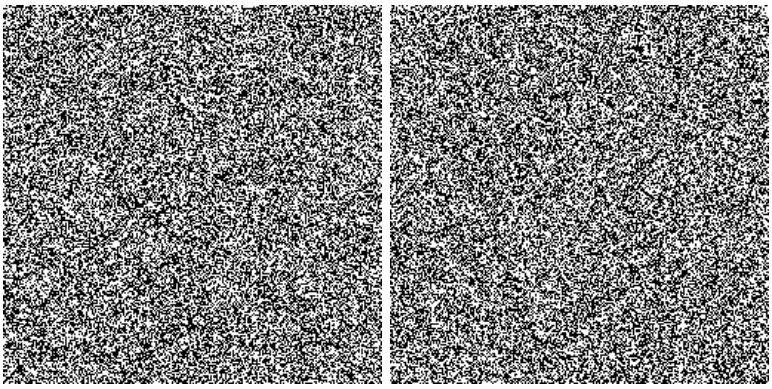

FIGURE 5 – A gauche le plan MSB, à droite le plan LSB

2 Implémentation d'une méthode d'insertion

Dans cette partie, nous avons tout d'abord générer un message composé d'autant de bits aléatoires qu'il y a de pixel dans l'image.

```
vector<unsigned char> _message = vector<unsigned char>(imIn.getTotalSize());
srand(time(NULL));
for(int i=0; i<imIn.getTotalSize(); ++i)
    _message[i] = rand()%2;
```

FIGURE 6 – Génération de bits aléatoires

Nous avons ensuite substitué un bit, à une position choisie, par le bit du message à insérer.

```
unsigned char _randBitMask = (1 << _plan)^255;

for(int i = 0; i<imIn.getWidth(); ++i)
    for(int j = 0; j<imIn.getHeight(); ++j)
    {
        unsigned char _msgValue = _message[i*imIn.getHeight()+j];
        imOut[i][j] = (imOut[i][j] & _randBitMask) | (_msgValue <<_plan);
    }
```

FIGURE 7 – Substitution d'un bit des pixels par le bit du message

Nous avons testé la substitution sur le plan MSB ainsi que sur le plan LSB.

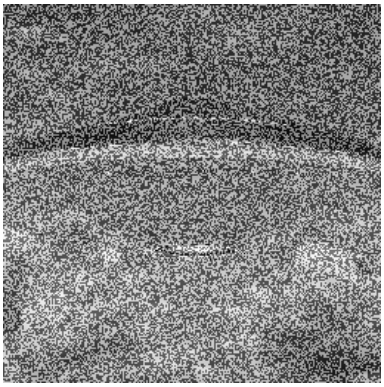

FIGURE 8 – Image avec le plan MSB substitué

```
ingo@Desktop-MS-7B17:~/Bureau/HAI918I_Image_securite_DL/TP3$ ./PSNR image.pgm image_inser_7.pgm
9.00818
```

FIGURE 9 – PSNR entre l'image avec le plan MSB substitué et l'image originale

Ici nous avons un PSNR de 9 dB. La qualité de l'image est mauvaise car nous avons remplacé les bits ayant le plus d'information par notre message, l'image a donc perdu beaucoup d'informations.

FIGURE 10 – Image avec le plan LSB substitué

```
lngo@Desktop-MS-7B17:~/Bureau/HAI918I_Image_securite_DL/TP3$ ./PSNR image.pgm image_inser_0.pgm
51.1576
```

FIGURE 11 – PSNR entre l'image avec le plan LSB substitué et l'image originale

Ici nous avons un PSNR de 51.16 dB. La qualité de l'image est très bonne et bien meilleur que lorsque la substitution se fait sur le plan MSB. Cela s'explique car ici ce sont les bits de poids faibles qui sont substitués. Ces bits ayant moins d'informations, les différences avec l'image d'origine sont beaucoup moins importantes et ne sont pas visible à l'oeil nu.

3 Un peu de stéganalyse

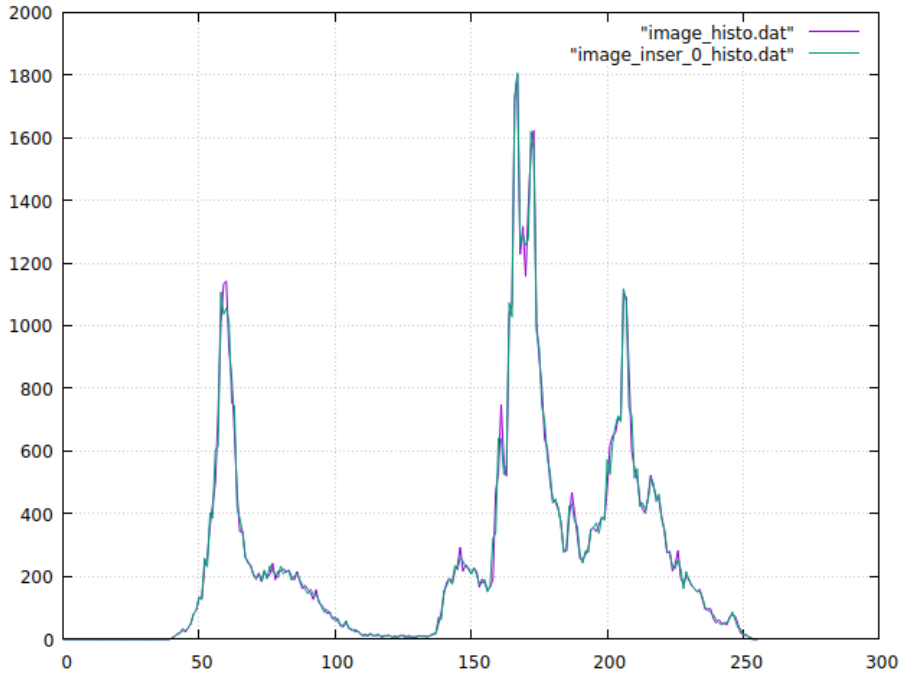

FIGURE 12 – Histogramme de l'image en violet et de l'image avec le plan LSB substitué en vert

Les deux histogrammes sont assez semblable car les 7 bits de poids fort sont identique entre les deux images. Les différences se font sur le LSB et nous avons 1 chance sur 2 de le modifier (donc de modifier la valeur de +1 ou -1). Par exemple un pixel à 64 aura 1 chance sur 2 de passer à 65 et inversement, les différences sont donc minimes.

Visuellement, un attaquant ne peut pas faire la différence entre les deux images. Néanmoins, les histogrammes sont différents, l'attaquant peut donc comparer les deux histogramme afin deux voir s'il y a un message caché.