Image, Sécurité et Deep Learning

Sécurité Multimédia – Tp1

Emery Bourget-vecchio
M2 Imagine
24/10/2021

1) Chiffrement d'images

a) Implémentation d'une méthode de chiffrement

Image de base

i) & ii)

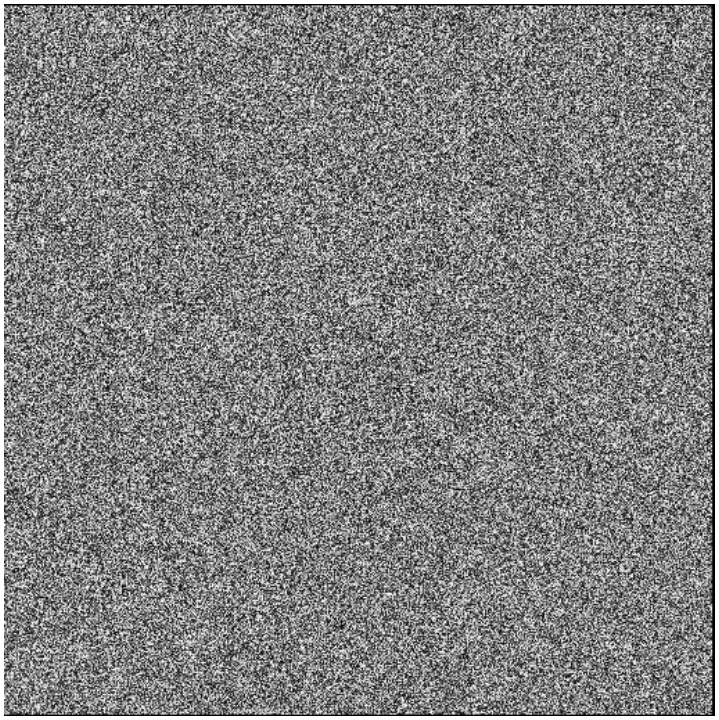


Image encodée avec XOR (SEED=53)

iii)

Image décodée à partir de l'image du ii)

iv)

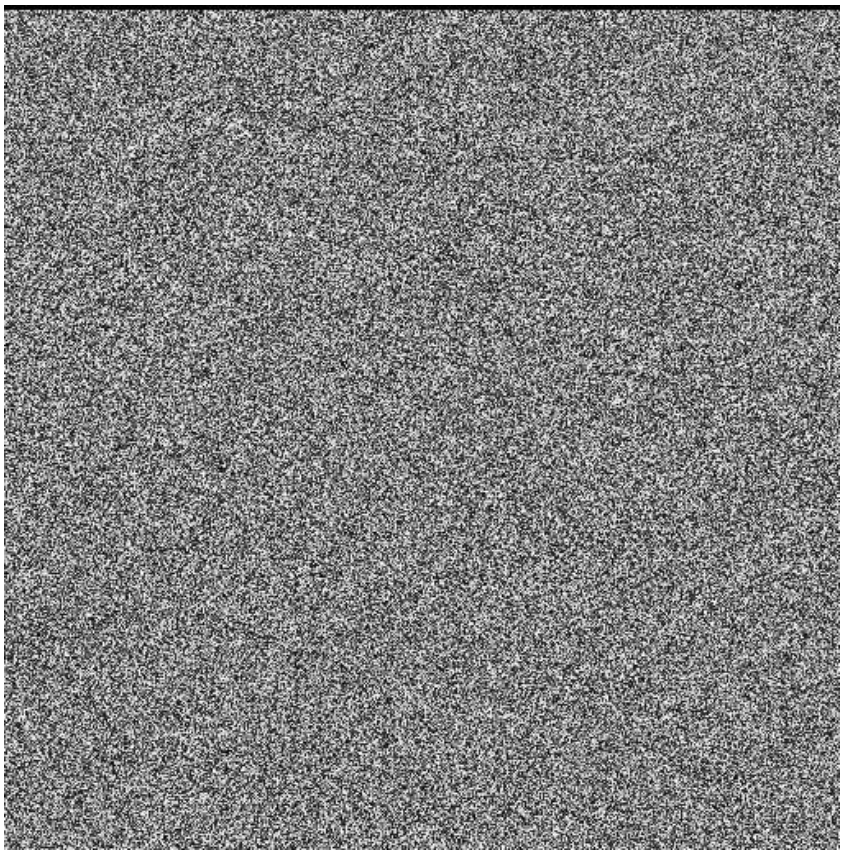


Image encodée avec XOR (SEED=20)

On remarque que les deux images encodées avec une clé différente donnent des images chiffrées qui sont également différentes.
Pour le tester, j'ai essayé de décrypter l'image avec un SEED = 20 avec la séquence binaire obtenue avec le SEED = 53. Si les deux images encodées sont identiques, alors on devrait obtenir l'image originale.

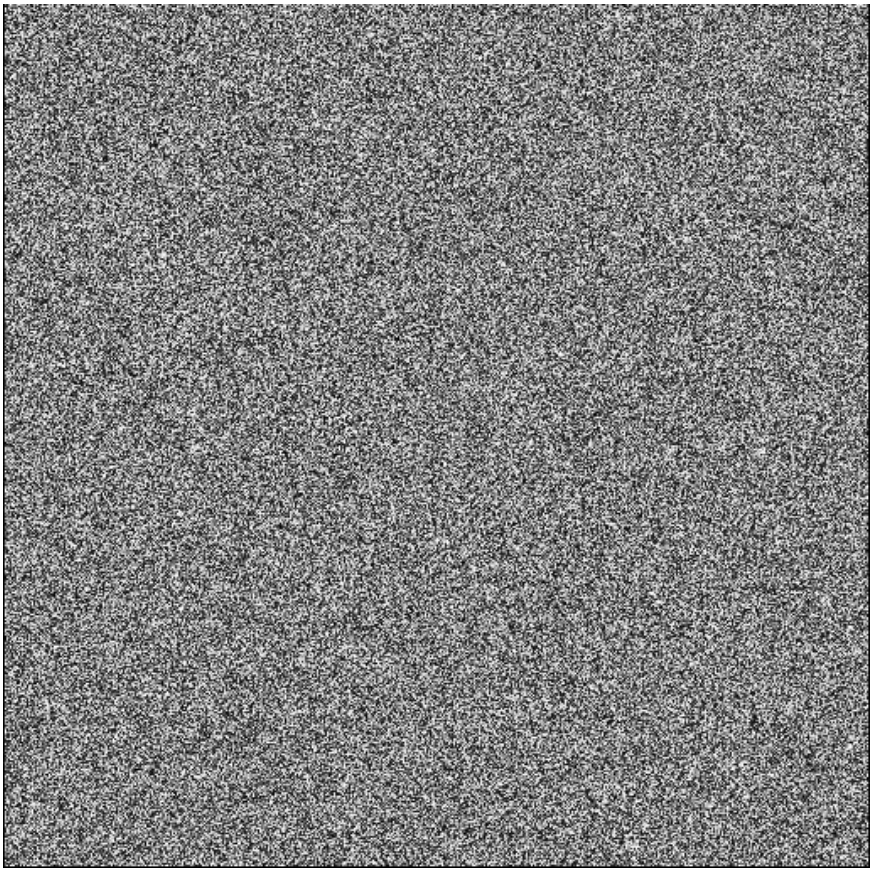


Image «déchiffrée»

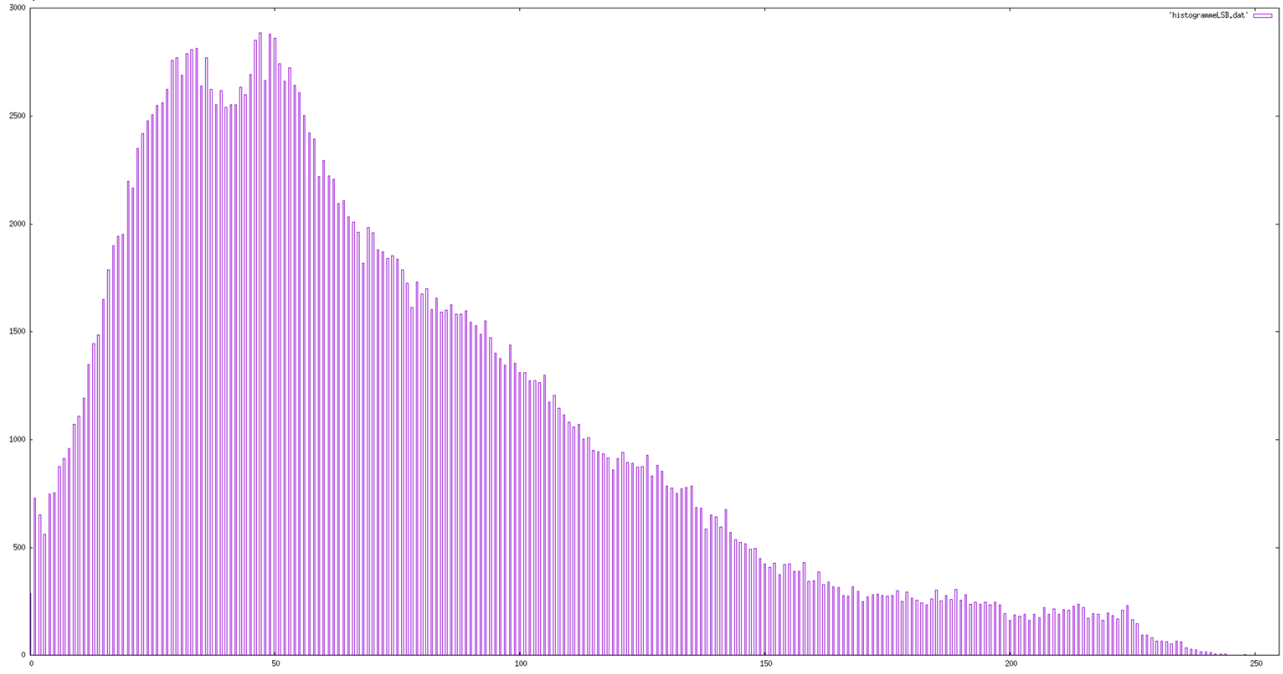
On remarque qu'on obtient une image avec du bruit, ce qui prouve que les deux images chiffrées sont différentes selon le SEED.

b) Analyse statistique

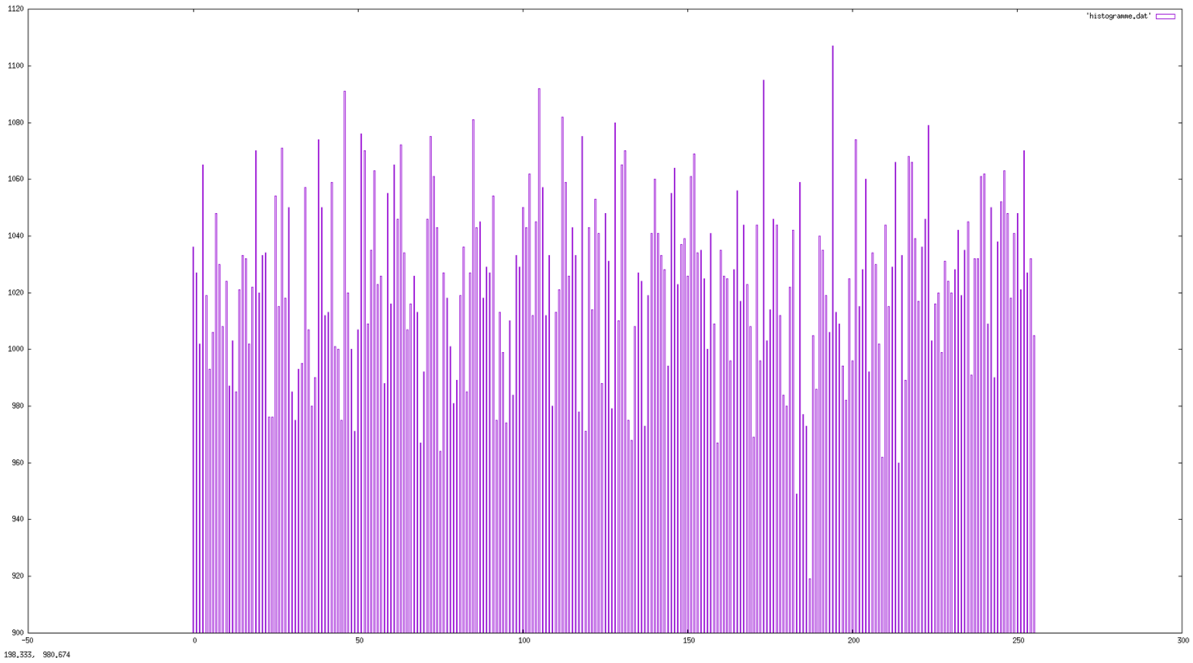
i) Entre l'image chiffrée et déchiffrée, j'obtiens un **PSNR** de 7,821889.

ii) Pour l'image de base, mon entropie est 7,416031 et pour l'image encodée, l'entropie est 7,999345.

iii)



Histogramme pour l'image claire



Histogramme pour l'image chiffré

iv) **PSNR** : On remarque qu'il est faible (un PSNR entre deux images qui se ressemble est compris entre 30dB et 50dB). Ce qui signifie qu'il n'y a visuellement pas de corrélation entre l'image claire et chiffrée.

Entropie : L'entropie de l'image chiffrée est de quasiment 8, ce qui montre que l'image est « aléatoire » ce qui est voulu pour une image chiffrée.
Histogramme: L'histogramme de l'image chiffré est complément aléatoire, aucune zone ne se démarque et il est complètement impossible de faire un lien entre lui et l'histogramme de l'image claire.

Conclusion
Avec toutes les analyses que nous avons réalisées ci-dessus, on peut donc dire que la compression est efficace et possède un haut niveau de complexité.
Cependant, on peut toujours connaître des métadonnées comme la taille de l'image et si c'est une image en N&B ou en couleur.
De plus, la clé est générée pseudo aléatoirement, ce qui est une faille en soi. Dans notre cas j'ai pris des SEED relativement peu élevés, il est donc facile avec un algorithme glouton de trouver la bonne clé. Pour avoir un meilleur chiffrement il faudrait prendre un SEED très élevé ou même mieux, réaliser un chiffrement parfait en prenant une clé parfaitement aléatoire.
Enfin la clé permet décoder et de décoder, il faudrait pour renforcer la sécurité utiliser une clé asymétrique avec une clé publique et une clé privée.

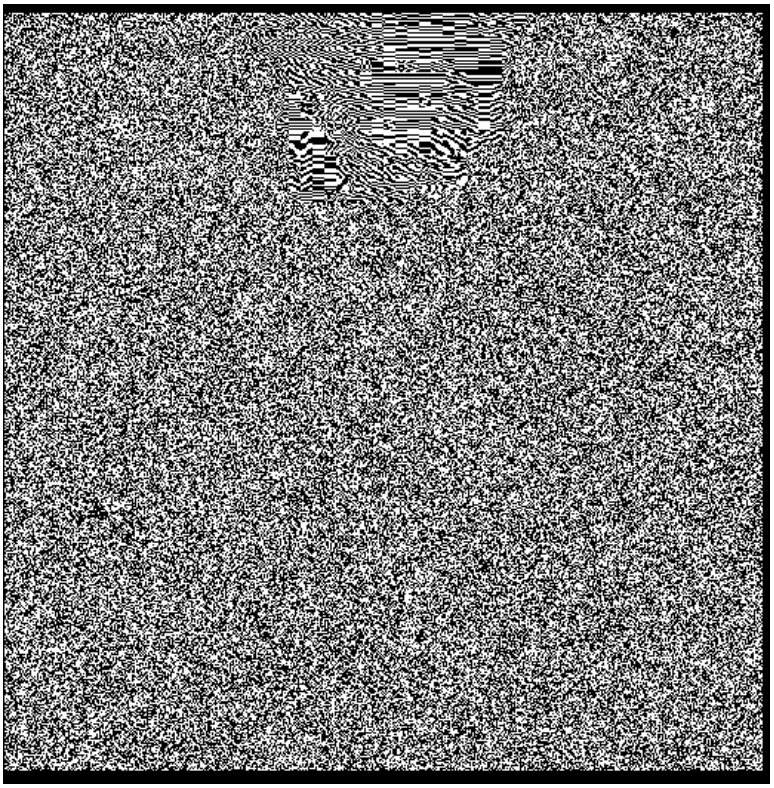
2) Insertion de données cachées

a) Plan binaire

i)

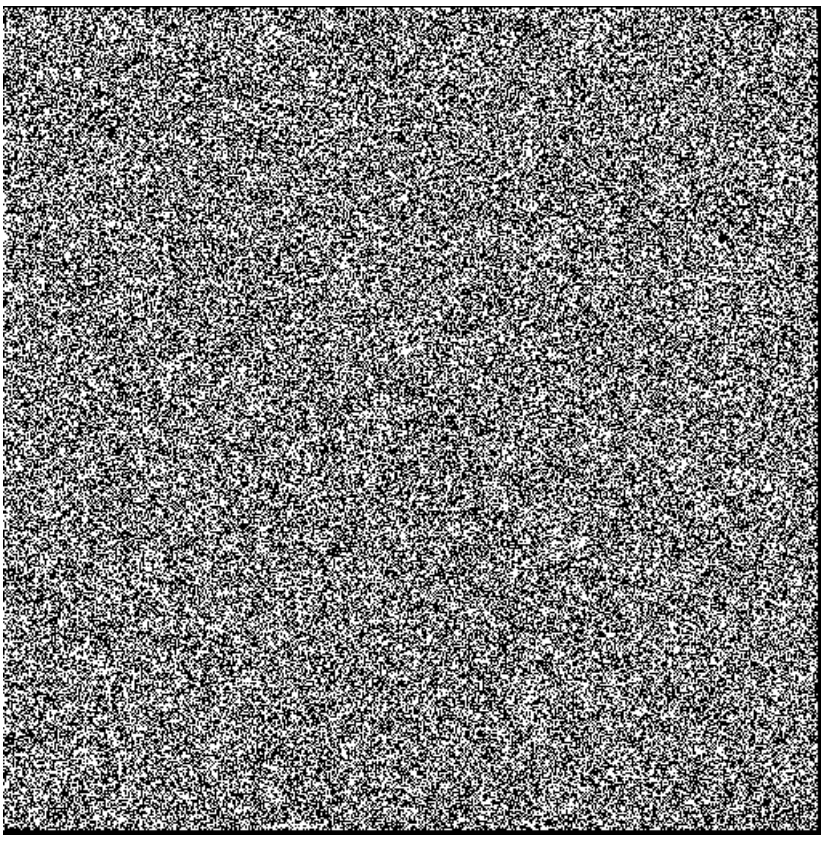
MSB de l'image claire

Pour le MSB de l'image claire, on peut distinguer presque tous les contours de notre image, les zones plus claires de l'image de base sont marquées ici par des zones blanches. Cela est normal, car si k=7, cela veut dire que la valeur du pixel est supérieure à 128. Comme l'image de base est plutôt sombre, alors on a beaucoup de valeur de notre image qui ne dépasse pas la valeur 128.

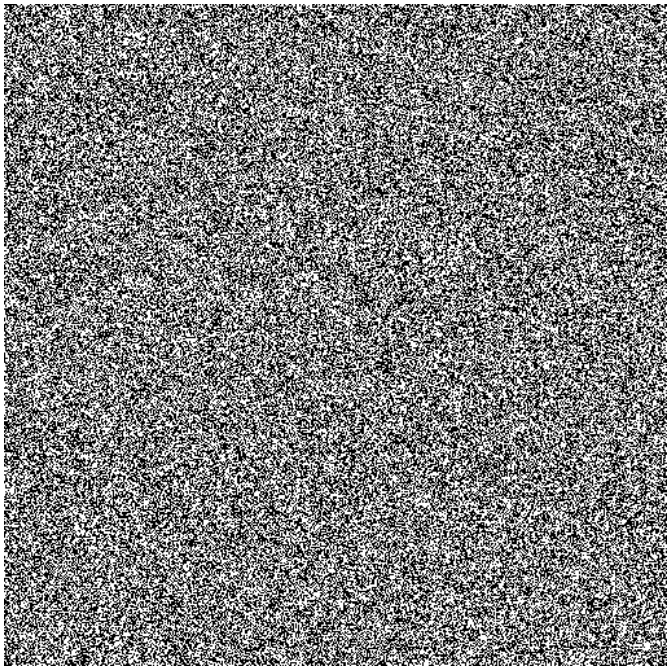


LSB de l'image claire

Pour le LSB de l'image claire, on remarque que toute l'image est bruitée sauf la partie qui représente le ciel où il y a une espèce d'anomalie. Cela peut s'expliquer par la faible variation de couleur à ce niveau.



MSB de l'image chiffrée



LSB de l'image chiffrée

Pour le LSB et le MSB de l'image chiffré, tout est bruité sans distinction de contour.

Pour conclure, plus k est élevé, plus on est amené à avoir des contours et des formes distinguables de l'image originale. Ce qui est normal, car cela correspond aux bits les plus signifiants
À l'inverse, plus le k est faible, plus on va obtenir des images bruitées, encore une fois normales, car cela correspond aux bits les moins significatifs. Comme les valeurs vont très souvent varier, il est normal d'avoir du bruit.

b) Implémentation de la méthode d'insertion

i) J'ai d'abord créé une séquence de bits de la même taille que l'image.

Extrait de la séquence de bit

ii) Le code permettant de réaliser la substitution se trouve dans la fonction hideMessage()

iii) J'ai ensuite inséré la séquence de bit dans chacun des MSB de chaque pixel de mon image. J'obtiens l'image suivante :

Image avec le MSB remplacé par la séquence aléatoire

Le **PSNR** est 8.985950

Cela signifie qu'il y a très peu de ressemblance entre l'image de base et l'image contenant le message codé. Cela est normal, car on modifie la valeur du MSB, qui est le bit le plus significatif, c'est-à-dire que c'est celui qui contient le plus d'information.
Par exemple, si on prend un pixel aléatoire de l'image de base, on a 101101(45). Si l'on souhaite inséré un 1 au MSB, on obtient alors 10101101 ce qui transforme notre chiffre à 173. On passe donc à un niveau de gris plutôt sombre à un niveau de gris clair, d'où l'image très claire qu'on obtient ci-dessus.

iv) J’ai ensuite inséré la séquence de bit dans chacun des LSB de chaque pixel de mon image. J’obtiens l’image suivante :

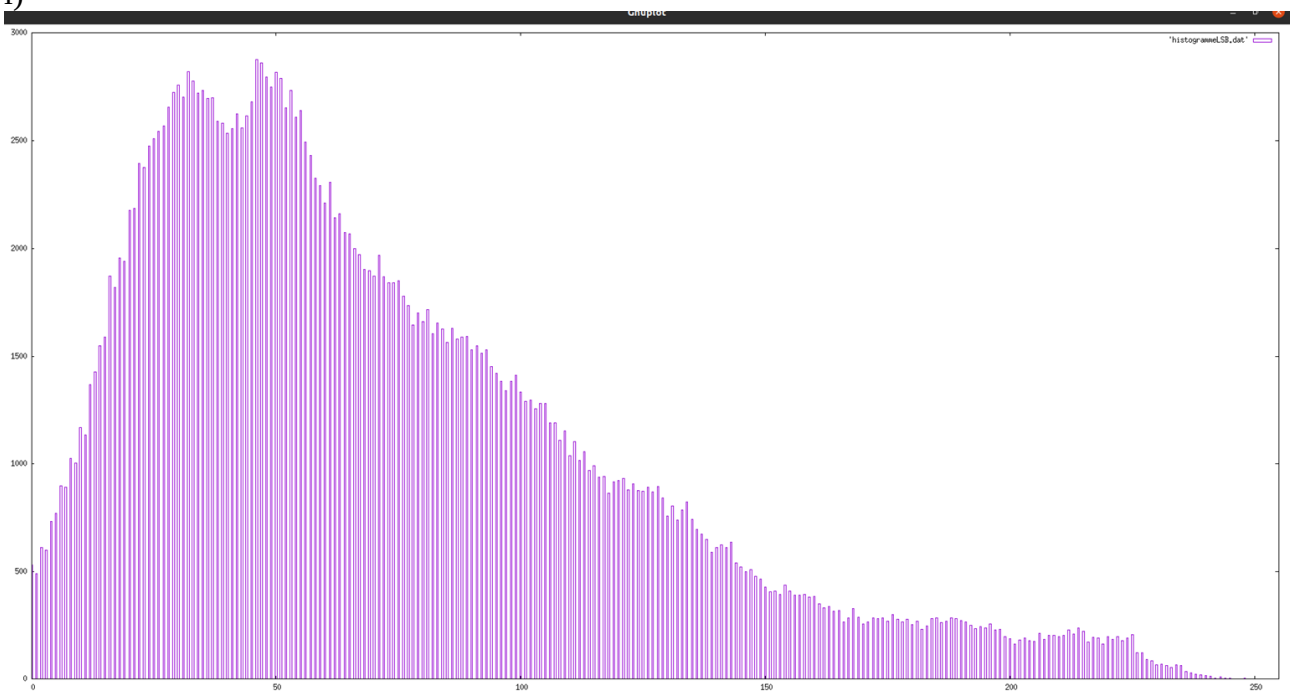
Image avec le LSB remplacé par la séquence aléatoire

Le **PSNR** est de 51.149464

Cela signifie qu’on a un haut niveau de corrélation entre l’image de base et l’image contenant le message codé. Cela est normal, car comme son nom l’indique, le LSB représente le bit le moins important(modifié un LSB ne fera varier que de 1 le niveau de gris) , donc le modifié ne changera quasiment pas le code source de l’image et sera indétectable à l’œil nu.

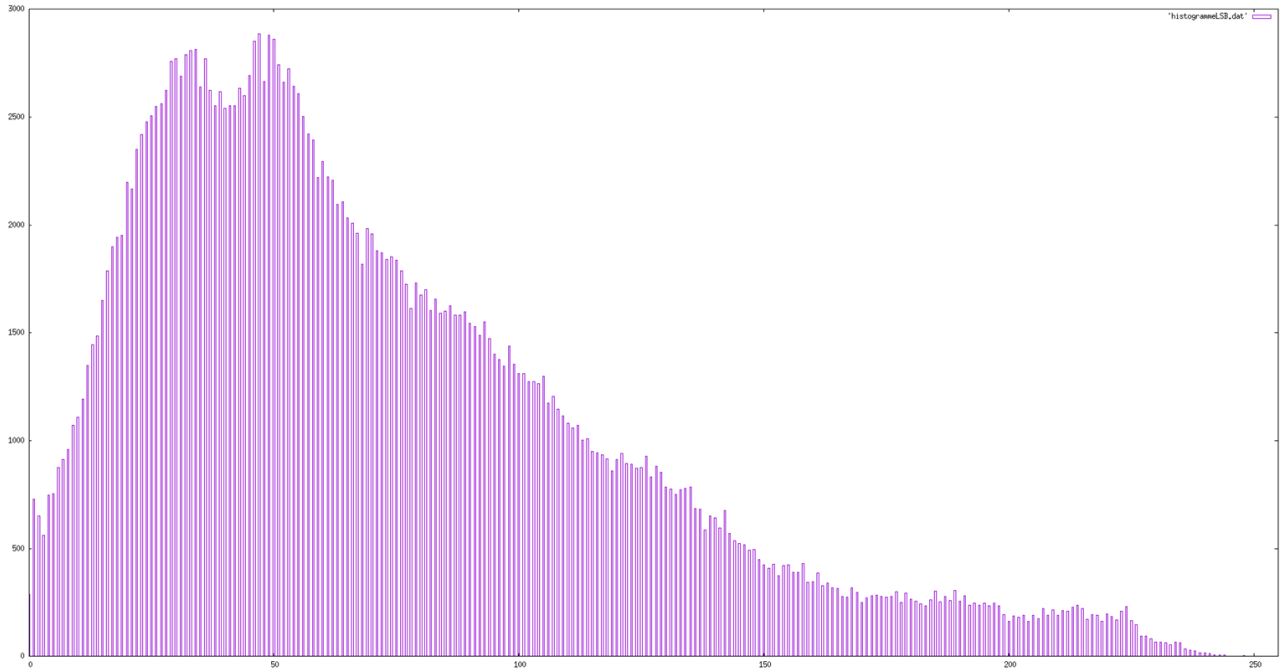
c) un peu de stéganalyse

i)



Histogramme de l'image marquée après substitution des LSB

ii)



Histogramme de l'image de base

En comparant les deux histogrammes, on remarque très peu de différence, juste un peu du bruit sur l'image modifiée. Cela montre que l'image modifiée est très peu différente de l'image de base.

iii) Si le gardien est passif (c'est-à-dire qu'il ne fait qu'observer le trafic) , il ne remarquera rien d'anormal juste en observant l'image.
Cependant, si le gardien s'intéresse à la composition de l'image, il pourra remarquer une différence avec cette formule.

- Théorie (Cachin 1998) : comparaison de la distribution du support avant et après l'insertion en utilisant la divergence de Kullback Leibler

$$D_{KL}(P_c||P_s) = \sum_{x \in C} P_c(x) \log_2 \frac{P_c(x)}{P_s(x)}$$

- Si $D_{KL}(P_c||P_s) = 0$, parfaitement sûr
- Si $D_{KL}(P_c||P_s) < \varepsilon$, dit ε-sûr

Ainsi, si le résultat n'est pas égal à 0, alors cela pourra éveiller les soupçons du gardien et il pourra essayer de devenir « actif » en compressant l'image ou en la filtrant pour détruire le message codé ou bien même devenir « malicieux ». afin d'extraire le message pour le contourner pour ses propres fins.

3) Insertion de données cachées… dans les images chiffrées

a) Implémentation naïve

i)

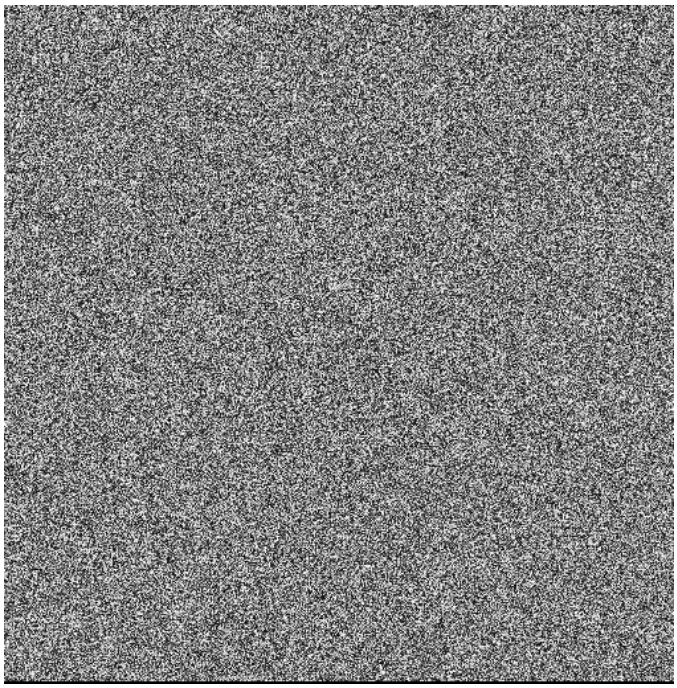


Image chiffrée en XOR

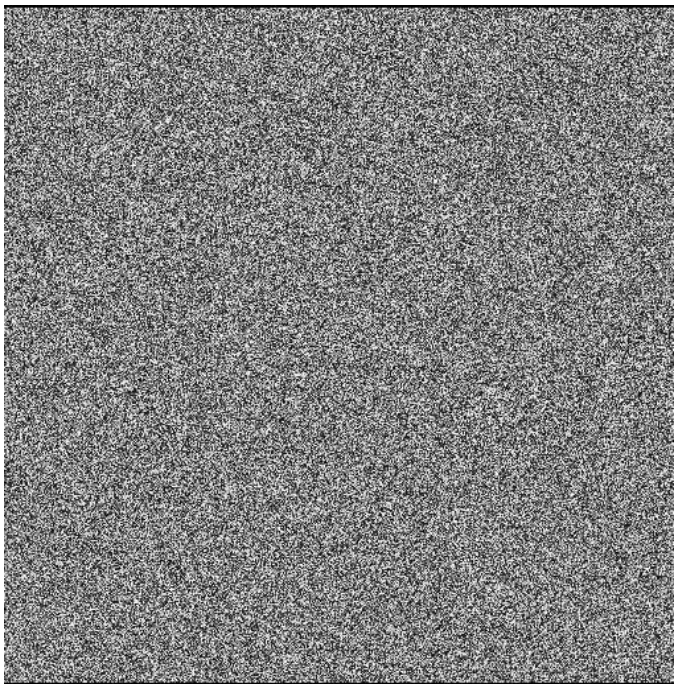


Image chiffrée en XOR avec message secret dans les MSB sauf première ligne et première colonne

ii)

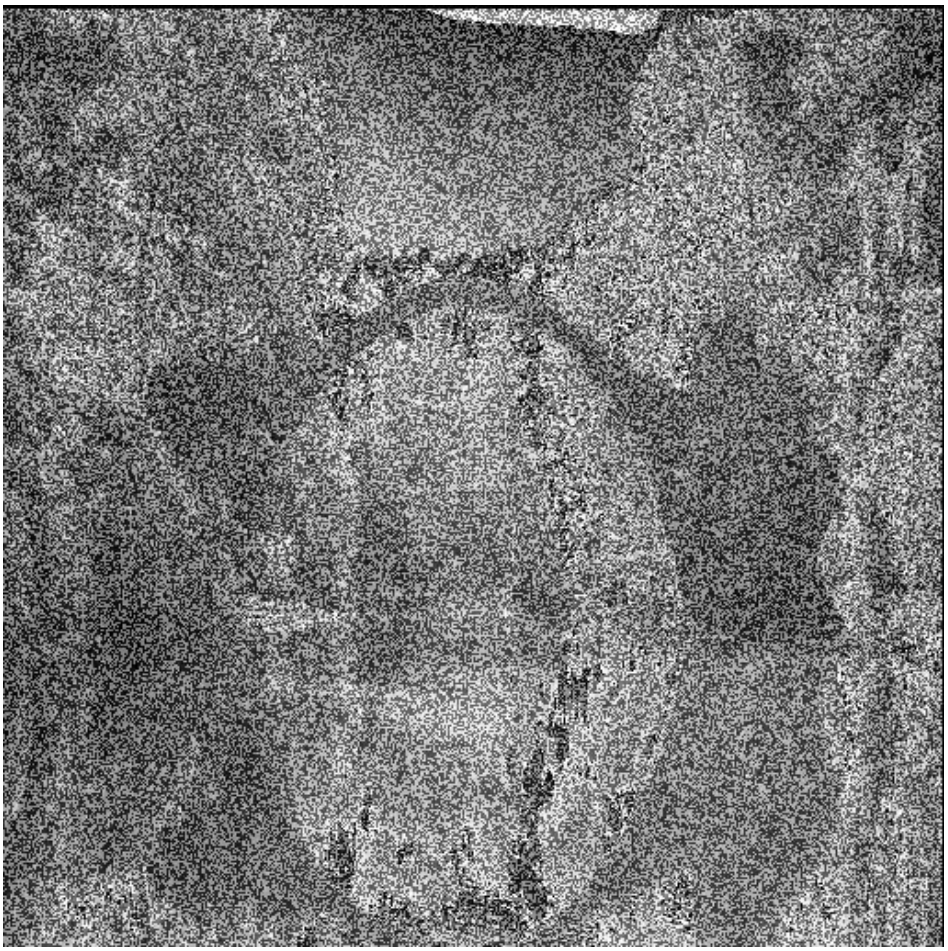


Image déchiffrée en XOR avec message secret dans les MSB

On remarque qu'on obtient le même type d'image que dans le **2)b)iii)**, ce qui montre que la plupart des MSB de l'image déchiffrée sont mal reconstruits.

b)**Prédiction des valeurs des MSB**

i) Fait dans le code avec la fonction predict()

ii) Fait dans le code inverBit()

```
Value: 200
InverseValue 72
```

```
Value: 50
InverseValue 178
```

iii)

Image reconstruite en prédisant la valeur du MSB de chaque pixel

Le **PSNR** est 19.663700

Les artefacts doivent provenir de la comparaison de distance entre la distance de (pred(i),ImgIn(i) et (pred(i) et inverse(Imgin(i)). On remarque que les erreurs apparaissent surtout sur les contours, ce qui est logique, car c'est dans ce genre de cas où la prédiction va indiquer la mauvaise valeur et comme notre algorithme se base sur le principe de la propagation, on peut voir l'erreur se propager sous forme d'une diagonale.

iv) La fonction de prétraitement permet d'éviter les erreurs liées aux contours en vérifiant pour chaque pixel si la distance (pred(i),ImgIn(i)) est moins élevée que la distance (pred(i),Inv(ImgIn(i))). Si c'est le cas, alors tout va bien et ImgPre(i) = ImgIn(i). Cependant si ce n'est pas le cas, il faut alors faire un ajustement pour corriger cette erreur, ainsi si la valeur du pixel est inférieure à 128, alors on lui retire 63 et à l'inverse s'il est supérieur à 128, alors on lui rajoute 63.

On obtient ainsi l'image suivante :

Image avec un prétraitement

Le **PSNR** est 46.398434

v) En appliquant le prétraitement de l'image originale, le chiffrement de l'image prétraitée, l'insertion de donnée cachée, puis la reconstruction par prédiction, on obtient l'image suivante :

Image avec un prétraitement reconstruite

On remarque qu'il n'y a plus d'artefacts et que l'image reconstruite est fortement similaire à l'image originale.

Le **PSNR** est 46.398434

On remarque que le **PSNR** est exactement le même qu'avec l'image prétraitée sans encodage, en calculant le **PSNR** entre l'image avec un prétraitement et l'image reconstruite on obtient un PSNR infini, ce qui signifie que les deux images sont absolument identiques et que la compression et le chiffrement n'ont pas affecté la qualité de l'image. Finalement, le seul passage qui nous fait perdre réellement de la qualité est le prétraitement de l'image, mais cela est au final nécessaire pour pouvoir restituer correctement l'image.