DÉFI 1 - Défi LSB - Stéganographie

Objectif : Ce défi consiste à trouver du image dissimulée dans une image.

L'objectif de la stéganographie et de la cryptographie est de transmettre un message à un tiers à l'insu de tous :
➢ La stéganographie se fonde sur la dissimulation : il s'agit de faire passer inaperçu un message dans un autre message. Ici, nous dissimulons une image dans une image.
➢ La cryptographie, « art du secret », oeuvre à rendre un message inintelligible pour tous, excepté pour le tiers destinataire.
Les deux techniques peuvent se cumuler. Pour résumé : la cryptographie repose sur le fait que le message ne soit pas compris, la stéganographie repose sur le fait que le message ne soit pas détecté.

► **Ceci est un défi : saurez-vous retrouver une image encodée par stéganographie dans une autre image ?**

Pour valider votre réponse, il faudra aussi envoyer un script PYTHON annoté et propre à l'adresse mail au bas de la page.

Amusez-vous bien !

Support PYTHON- gestion des images matricielles

Nous allons travailler sur des images matricielles au format TIF qui code chaque pixel sur 4 octets : $[\![0,255]\!]^4$ (rouge, vert bleu, transparence). Les fonctions à utiliser pour la lecture et la modification des ces images, lors de ce défi, sont dans le module `imageio` :

```
import imageio as im
```

Notation	Description
`im.imread( )`	transforme une image en un tableau de 4 octets
`im.imsave( )`	transforme un tableau de nombres en image

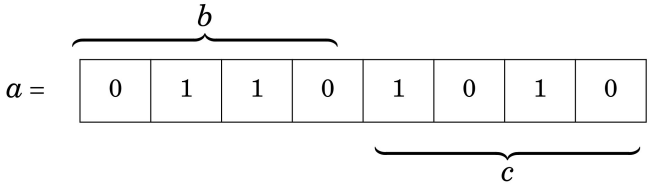
Attention ! Les fonctions `imread` et `imsave` du module `matplotlib.pyplot` utilisées au TP précédent codent un pixel en $[0,1]^4$: un matrice de type `ndarray` de nombre entre 0 et 1 (rouge, vert, bleu, transparence).
Ici, on préfère travailler sur des entiers !

Attention ! Afin de lire et d'écrire des fichiers sans se préoccuper de leur adresse physique (sur votre ordinateur), je vous recommande de mettre tous vos fichiers dans le même répertoire, puis de redémarrer le Shell à cette même adresse : `Exécuter/Démarrer le script` ou `Ctrl+Mal+E` (à faire uniquement une fois au début).

Stéganographie - méthode LSB

La méthode LSB (Least Significant Beat) est aussi dite méthode des bits de poids faible.
L'idée est de prendre une image et de la modifier de manière aussi discrète que possible afin d'y dissimuler l'information à transmettre.

Dans une image matricielle, la couleur d'un pixel est déterminée par trois nombres. Pour ce défi, chaque teinte sera codée par un octet ('bytes' en anglais = 8 bits), c'est à dire un nombre de $[\![0,255]\!]$.
Exemple pour $a = 106$, on a : $a = 2^6 + 2^5 + 2^3 + 2^1$



Les bits de poids forts sont à gauche, ils sont prépondérant dans la proportion de la teinte. Les bits de poids faible sont à droite, ils assurent une finesse dans le dégradé des teintes. Modifier les bits de poids faibles ne change donc pas de manière importante la teinte. On introduit, ici, une découpe par le milieu avec $b = 6$ et $c = 10$ dont la concaténation en binaire redonne a.

Concrètement, si l'on modifie les 4 derniers bits, la nuance initiale $n \in [\![0,255]\!]$ est donc modifiée en $n' \in [\![n-16, n+16]\!]$. La variation de teinte est légère, quasi imperceptible pour l'oeil. En pratique, si l'image leurre est riche en motif, en variations brutales de couleurs, sans zone monochrome, la modification passe inaperçue.

Ainsi, l'image leurre est définie en modifiant chaque teinte (1 octet = 8 bits) de chaque pixel par :
- les quatre bits de poids fort sont ceux de l'image leurre : b
- les quatre bits de poids faible sont les quatre bits de poids fort de l'image dissimulée : c

Exercice 1 Soit $a \in [\![0,255]\!]$ l'entier associé à un octet et les entiers b et c respectivement associés aux bits de poids fort et au bits de poids faible de a.
1. Justifier que `b=a//16`.
2. Exprimer c en fonction de a.

Exercice 2
1. Écrire une fonction `cacher(imag1,imag2,imag3)` qui vérifie que `imag1` et `imag2` sont deux images de même taille et qui dissimule, par la méthode LSB, l'image `imag1` dans l'image `imag2` pour retourner le résultat dans l'image `imag3`.
2. Écrire une fonction `trouver(imag1,imag2)` qui retourne l'image `imag2` dissimulée dans l'image leurre `imag1`.
3. Relever le défi en retrouvant l'image dissimulée dans l'image `defi.tif`.

Lycée Saint-Exupery – maffre.saintexupery@gmail.com – MPSI