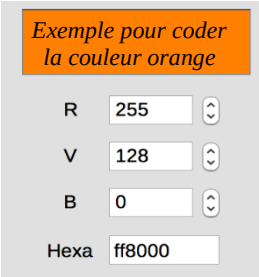
Chiffrement d'un texte dans une image


image .jpg


image .png

Le principe est de modifier de façon non perceptible le codage des couleurs d'un pixel pour y loger, pixel après pixel, une partie de l'information correspondant au codage d'une lettre d'un message.

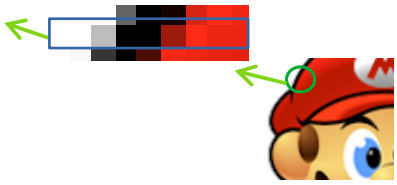
Niveaux R-V-B (-A)



Chaque pixel de couleur est constitué de trois niveaux de luminosité pour chaque couleur primaire Rouge, Vert et Bleu. *Pour les images de type png, il existe un quatrième indicateur : la transparence du pixel par rapport au fond (couche Alpha)*

Le principe de la synthèse additive permet donc de coder 256*256*256 couleurs différentes par pixel.

8 Pixels								
Rouge	255	255	125	20	20	127	241	200
Vert	255	255	125	20	20	10	13	26
Bleu	255	255	125	20	20	10	13	26



Pour loger l'information correspondant à nos caractères dans l'image, on va choisir de ne modifier par exemple que le niveau de bleu. On peut représenter **ce niveau de bleu sous forme binaire** :

Bleu	255	255	125	20	20	10	13	26
	1111 1111	1111 1111	0111 1101	0001 0100	0001 0100	0000 1010	0000 1101	0001 1010

Prenons par exemple la lettre 'z'.
Son code ASCII dans la table des caractères est 122. La valeur 122 se code **0 1 1 1 1 0 1 0** en binaire.
On va donc modifier les bits « de poids faible » de ces huit pixels ci-dessus pour y loger **un bit du code binaire de notre lettre 'z' dans chacun des pixels :**

	1111 1110	1111 1111	0111 1101	0001 0101	0001 0101	0000 1010	0000 1101	0001 1010
Nouve auniv. Bleu	254	255	125	21	21	10	13	26

Ces faibles modifications ne seront pas perceptibles dans l'image. Dans une image de 128 × 128 pixels, on dispose donc de 16 384 pixels. Chaque caractère de notre texte est codé sur 8 bits. Avec 1 bit d'un caractère par pixel, on peut donc dissimuler dans l'image 16384 ÷ 8 soit 2048 caractères !!!

Algorithme :

Coder une fonction qui reçoit le caractère à coder et retourne son code ASCII sous forme d'une liste de bits.

Ouvrir et récupérer la définition de l'image.

Pour chaque caractère du message :
    # Convertir le caractère en liste de bits
    # Pour chaque bit du caractère :
        # Lire les valeurs RVB d'un pixel
        # Remplacer le bit de poids faible de la composante Bleu avec le bit par le bit du caractère
        # Replacer le pixel dans l'image
        # Passer au pixel suivant
Enregistrer l'image modifiée.

Pour décoder l'image et retrouver notre message, il s'agit d'effectuer les opérations dans le sens inverse.

Coder une fonction qui reçoit une liste de bits (code ASCII) et retourne le caractère correspondant

Tant que le message n'est pas terminé et qu'il reste des pixels :
    # Lire la composante R,V,B d'un pixel
    # Mémoriser le bit de poids faible dans une liste
    # Si la liste contient 8 bits :
    `   # Convertir les huit bits en caractère
        # Mémoriser le caractère
    # Passer au pixel suivant
Afficher le message

L'image qui sert de support est située dans le même dossier que le script Python