stegano

November 4, 2019

0.1 Stéganographie

- Une des méthodes pour cacher une donnée dans une image est d'encoder cette donnée dans le bit de poids faible des octets de couleurs de l'image lorsque celle ci est codée en RGB (Palette d'une image bmp)
- Attention cependant, une image BMP ne commence pas directement par les octets qui codent chaque pixel de l'image, mais par un header tel qu'indiqué ici

1. Écrire une fonction qui prend en entrée un octet (représenté par un entier entre 0 et 255) et qui retourne la valeur du bit de poids faible.

Remarque:

- Il est possible de s'aider des bitwises operators.

[1]:
```
my_bytes = bytes([7])   # create an array of one byte with value 7
my_byte = my_bytes[0]
print("my_byte: ", my_byte)
```

```
my_byte:  7
```

2. Écrire une fonction qui prend en entrée une liste de 8 bits et retourne l'octet construit à partir de cette liste.

Remarques:

- Il est possible de s'aider des opérateurs de bit shift
- Il est plus simple de représenter le bit de poids faible à l'indice zéro de la liste de 8 bits

[2]:
```
# Octet 0b01000001 == 0x41 == 65 == b'A'
bit_list = [0b1, 0b0, 0b0, 0b0, 0b0, 0b0, 0b1, 0b0]
```

3. Retrouver les 3 octets cachés `b'hi!'` dans la suite de 3*8=24 octets suivants:

[3]:
```
data = b'\x12\x94\xc8\xb5\xf2]\xb5@\xc9@.i\xa6\xed\xb50\x8f\\\x80nNK,x'
```

4. Retrouver et vérifier, en Python, que le premier octet du premier pixel de l'image `secret_cat.bmp` se trouve à l'offset 138 du fichier.

Remarques:

- Header BMP

- L'adresse de départ est écrite en little endian

[4]:
```
four_bytes_value = b'\x39\x05\x00\x00'
value = int.from_bytes(four_bytes_value, byteorder='little')
print("0x39050000 (little endian) == ", value)
```

```
0x39050000 (little endian) ==  1337
```

5. Retrouver le texte caché (qui se termine par un octet nul) dans l'image `secret_cat.bmp`

6. Facultatif: écrire un programme Python qui cache une donnée dans une image BMP à l'aide de ce principe