Stéganographie par LSB

Stéganographie

Le principe de la stéganographie est de cacher un message pour ne pas qu'il soit lu par quelqu'un qui n'en est pas le destinataire. Il est à distinguer de la cryptographie dont le but est de rendre inintelligible le message et non de le masquer.

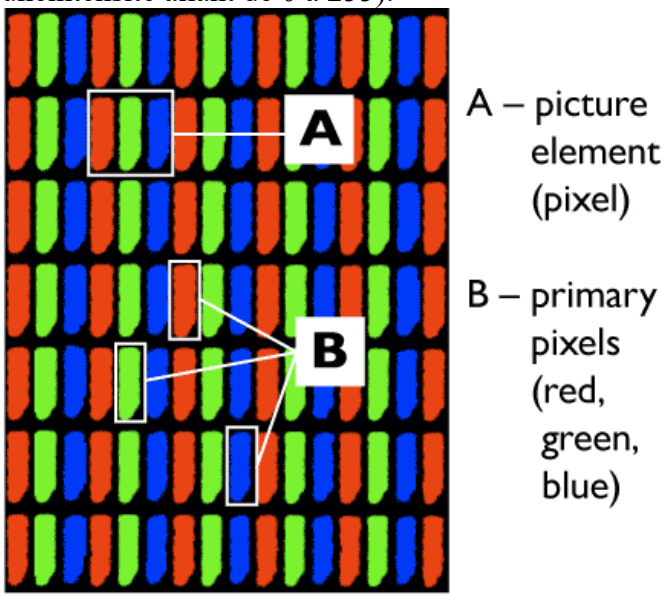
Plusieurs techniques de stéganographie sont possibles. Par exemple, une des premières utilisée durant le temps des romains était d'écrire le message à cacher sur le crane rasé d'un esclave, puis d'attendre que les cheveux aient repoussé pour envoyer l'esclave livrer le message.

La stéganographie est utile car lorsque quelqu'un intercepte un message, il ne se rends pas compte que le réel message est en fait caché dans celui qu'il lit.

La technique dite des LSB permet de cacher un fichier dans une image.

Images numériques

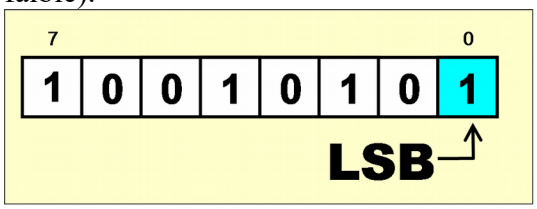
Les images numériques sont des ensembles de pixels. Chaque pixels sont des points de l'image d'une couleur. Et chaque pixels sont finalement 3 canaux de couleurs **R**ouge, **V**ert et **B**leu qui peuvent chacuns s'illuminer plus ou moins puissament. En faisant varier ainsi l'intensité de chaques canaux d'un pixel, on peut obtenir 256^3 couleurs possibles (chaque cannaux peuvent avoir uneintensité allant de 0 à 255).



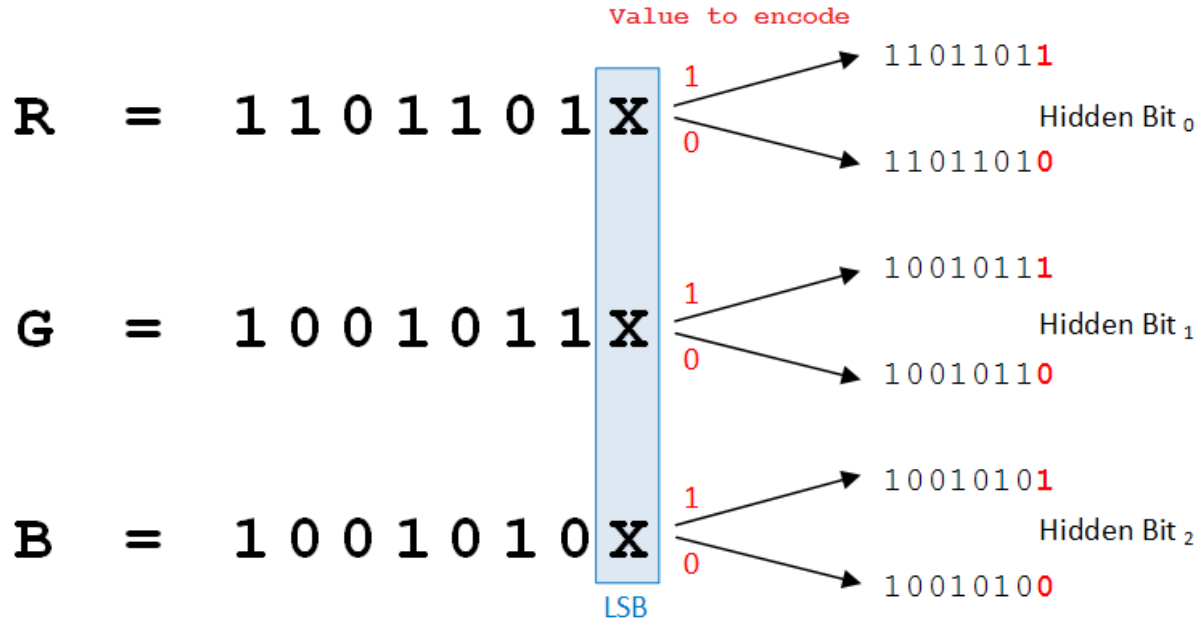
Les canaux des pixels d'une image valent chacuns 8 bits (pour avoir un nombre entre 0 et 255). Un pixel qui possède les trois canaux RVB fais donc 24 bits.

Stéganographie par LSB

Cette technique utilise les **L**east **S**ignificant **B**its des images, les bits de poids faible. En effet, on peut modifier les valeurs de l'intensité de chaque canaux de +1 ou -1 sans que cela soit visible à l'oeil nu. Et dans un bloc de 8 bits, la valeur de l'unité est le dernier bit du bloc (le bit de poids faible).



Ces LSB sont alors des bits utilisables pour contenir une autre information que l'intensité du canal. Et c'est cette place disponible qu'utilise la technique utilisée pour cacher les bits qui représentent le fichier à cacher dans une image.



On peut alors avoir 3 bits d'information par pixel, et donc la capacité d'information que l'on peut cacher dans une image est de 3*nbr_pixels_image.

Code python

Pour mettre en place cet algorithme, il a fallu prendre en compte le fait que les fichiers à cacher sont de taille variable. Il faut alors que lorsque le programme récupère le fichier caché, il sache combien de LSB prendre pour former son fichier et donc quand s'arrêter. De plus, connaître le nom du fichier peut être utile.

Ainsi, trois étapes sont mises en places lorsque l'algorithme cache le fichier dans l'image :
- 22 premiers bits alloués à la taille du fichier en bits
- 80 bits suivant alloués au nom du fichier
- les bits suivant sont disponibles pour les bits du fichier

Fonctions

```
def put_LSB(data_cover, data_in, bit_start=0):
    """Ajoute dans data_cover la data_in

    Le data_cover doit être donner en 'RGB' et converti en bits.
    Le data_in doit être uniquement un tableau de bits à rentrer dans le data_cover.
    Paramètre optionnel:
    -bit_start=0 -- Permet de définir un bit sur lequel commencer à placer la data_in dans data_cover

    """
    for i in range(bit_start, len(data_in) + bit_start):
        pixel = int(i/3)
        canal = i%3
        data_cover[pixel][canal] = data_cover[pixel][canal][:-1] + data_in[i - bit_start]
    return data_cover
```

Cette fonction permet de cacher des bits (data_in) dans les LSB (de la data_cover) et ce avec en paramètre le numéro du LSB sur lequel commencer à travailler.

```
def get_LSB(data_cover, nbr_bits=-1, bit_start=0):
    """ Récupère la data caché dans les LSB du data_cover.

    La data_cover doit être un tableau de pixels 'RGB' déjà converti en bits
    Paramètres optionnels :
    -nbr_bits=-1 -- Permet d'indiquer combien de bits doit récupérer le programme. Si il n'a pas été changé,
    alors le programme lit tout le tableau data_cover.
    -bit_start=0 -- Permet de donner un bit sur lequel commencer à récupérer la data.

    """
    data = list()

    if nbr_bits == -1:
        nbr_bits = len(data_cover) * 3#Si le nbr_bits n'as pas été changé alors on prends par défaut tout le

    for i in range(bit_start, nbr_bits + bit_start):
        pixel = int(i/3)
        canal = i%3
        data.append(data_cover[pixel][canal][7])#Récupère le LSB
    return data
```

Cette fonction est la fonction inverse de put_LSB, elle permet donc de récupérer les LSB d'une data_cover et de donner un LSB de départ et le nombre de LSB à relever.

Remarques

La taille du fichier à cacher doit être au moins 8 fois plus légère que l'image pour avoir assez de place. Notons que on perds tout de même 102 bits d'information à cause de codage du nom et de la taille du fichier en plus des données du fichier brutes.

Sources

- http://www.bibmath.net/crypto/index.php?action=affiche&quoi=stegano/infostegano (explication de la technique LSB)