

LYCÉE BLAISE PASCAL CLERMONT-FERRAND

Stéganographie et Cryptographie visuelles

Projet d'Informatique et Sciences
du Numérique

Groupe : Clémence Vessaire et Naouel Kouiss TS5

Dossier personnel de Naouel Kouiss

2016-2017

Sommaire :

Présentation du projet.....	2
Cahier des charges.....	3
Répartition des tâches.....	3
Réalisation	
Stéganographie.....	4
Cryptographie.....	6
Interface graphique.....	7
Conclusion.....	13
Annexes	
Décodage de la stéganographie.....	14
Décodage de la cryptographie.....	15
Interface TKInter.....	16

Présentation du projet :

Lors d'une heure de libre avec Clémence, nous nous sommes amusées avec un livret de codage et de décryptage de texte. Nous nous sommes alors mises d'accord sur le fait que cela serait très intéressant d'en faire notre projet d'ISN. Nous en avons donc parlé avec notre professeur M. Faury, qui nous a conseillé de travailler sur le codage des images avec la stéganographie et la cryptographie visuelles.

La stéganographie est l'art de la dissimulation. Le but étant de faire passer inaperçu un message dans un autre. Cette méthode de dissimulation est connue depuis l'antiquité et l'on doit le premier usage connu de cette technique, aux alentours de 600 av. J.-C., à Nabuchodonosor, roi de Babylone, qui employait une méthode originale : il écrivait sur le crâne rasé de ses esclaves, attendait que leurs cheveux aient repoussé, et les envoyait à ses généraux. Il suffisait ensuite de raser à nouveau le messenger pour lire le texte. Il s'agit bien ici de stéganographie à proprement parler car l'information est cachée.

Dans le cadre de notre projet, le but est de dissimuler une image dans une autre en modifiant les bits de poids faible.

Quant à la cryptographie, étymologiquement la science du secret, il s'agit d'une science ancienne qui englobe la cryptographie et la cryptanalyse. Les premières méthodes de chiffrement remontent à l'Antiquité et se sont améliorées avec la fabrication de différentes machines de chiffrement, pour obtenir un rôle majeur lors de la Première et la Seconde Guerre mondiale. La cryptographie est l'art du secret, celle que nous utilisons dans notre projet est la cryptographie à clé secrète, cela signifie qu'au préalable, l'expéditeur et le receveur du message ont dû se mettre d'accord sur la clé choisie. Ce genre de codage est utilisé depuis bien longtemps, on en retrouve des traces jusqu'à 2000 ans avant JC. La cryptanalyse est, quant à elle, l'analyse de la cryptographie, donc son décodage. Pour notre projet, nous binarisons les images que nous codons et décodons avec une clé préalablement créée, constituée de pixel noirs et blancs.

Cahier des charges :

Le projet se rapporte aux domaines de compétences suivants :

- **Dimension algorithmique** : Création d'algorithmes qui permettent de coder une image selon deux méthodes : stéganographie et cryptographie à clé secrète.

- **Éléments de programmation** : Pour la réalisation, nous utilisons le langage de programmation Python avec la distribution Pyzo qui inclut des modules complémentaires, un éditeur de programme et une console qui affiche les résultats du programme exécuté dans l'éditeur. Nous avons utilisé ces modules et ces bibliothèques : imageio (pour travailler avec les images), matplotlib (pour afficher les images), pillow (pour travailler avec les images et en créer à partir de tableaux de valeurs numpy), numpy (pour créer des matrices), et Tkinter (pour l'interface utilisateur).

Production finale attendue : Un algorithme qui permet de coder n'importe quelle image selon une clé donnée (ou générée aléatoirement) ou bien des algorithmes de codage et de décodage permettant de cacher une image dans une autre.

Caractéristiques de la production finale : La stéganographie fonctionne correctement, ainsi qu'une petite interface qui permet de choisir la méthode et de rentrer les images que l'on souhaite.

Contraintes à respecter : Date butoir, travail en équipe, dossier écrit (5-10 pages)

Matériel et logiciel à mettre en œuvre : Python et éditeur d'image

Répartition des tâches :

- Clémence :
 - Codage de la stéganographie
 - Codage de la cryptographie
 - Algorithme de la binarisation d'image
- Naouel :
 - Décodage de la stéganographie
 - Décodage de la cryptographie
 - Interface utilisateur

Réalisation :

Stéganographie

J'ai donc commencé par le décodage de la stéganographie. Pour débiter, j'ai d'abord fait en sorte de décodé un octet. Cela consiste à mettre à 0 les quatre bits de poids fort de l'octet que l'on vient de récupérer. Ensuite, on décale de quatre rangs vers la gauche les quatre bits de poids faible, qui deviennent donc fort, et on met à 0 les quatre nouveaux bits de poids faible. A cause de cela, l'octet perdra donc en qualité par rapport à l'original.

```
def decode_octet(octet_code):  
    return (0b00001111 & octet_code) << 4
```

J'ai ensuite continué par décodé un pixel. Pour cela, j'ai donc commencé par mettre sous forme de liste les octets du pixel, puis j'ai remis dans un **for** mon code pour le décodage d'un octet. Les valeurs obtenues sont ensuite allées se mettre dans un tableau, que j'ai préalablement défini par ma liste `pixel_decode = []`

```
def decode_pixel(pixel_code):  
    pixel_code = list(pixel_code) #mettre les octets du pixel dans une liste  
    pixel_decode = [] #initialisation d'un tableau pour les pixels  
    decodes  
    for octet in pixel_code: #pour chaque octet du pixel  
        pixel_decode.append((0b00001111 & octet) << 4) #mettre les bits de poids fort à 0  
        et décaler de 4 rangs vers la gauche  
    return pixel_decode
```

Pour finir sur la stéganographie, j'ai terminé par coder la fonction de décodage de l'image cachée. Pour cela, j'ai d'abord importer les modules **matplotlib** et **imageio**.

```
import imageio  
import matplotlib.pyplot as plt
```

J'ai continué par entrer deux variables, hauteur et largeur, pour obtenir les valeurs des images de ces dernières. J'ai ensuite mis une fonction **for** qui lit chaque pixel de chaque ligne. Puis une autre fonction **for** pour chaque pixel de chaque colonne. Ensuite, j'ai créé une liste de chacun des pixels, cela me donne donc les trois composantes RGB. Puis j'ai remis dans un **for** mon code pour le décodage d'un pixel. J'ai mis les valeurs trouvées dans un tableau après avoir transformé les listes en tuple. Pour terminer, j'ai affiché l'image décodée grâce au module **matplotlib**.

```

import imageio
import matplotlib.pyplot as plt

image_codee =
imageio.imread("https://upload.wikimedia.org/wikipedia/commons/a/a8/Steganography
_original.png")

def decode_image(image_codee):
    largeur = image_codee.shape[0] #pour obtenir la valeur de la largeur
    hauteur = image_codee.shape[1] #pour obtenir la valeur de la hauteur
    for pixel_l in range(largeur): # pour chacun des pixels de la ligne
        for pixel_h in range(hauteur): # pour chacun des pixels de la colonne
            pixel_code = list(image_codee[pixel_l][pixel_h]) # donne les trois
composantes RGB
            pixel_decode = []
            for octet in pixel_code:
                pixel_decode.append((0b00001111 & octet) << 4) # mettre les bits
de poids fort à 0 et puis décaler de 4 rangs vers la gauche
            image_codee[pixel_l][pixel_h] = tuple(pixel_decode)
    plt.imshow(image_codee) # mettre image_codee sous plt
    plt.show()
decode_image(image_codee) # afficher l'image décodée

```

Cryptographie

La stéganographie nous a pris plus de temps que nous le pensions. Nous n'avons donc pas pu travailler efficacement pour la cryptographie. Pour cette partie, nous n'avons donc pas vraiment travaillé en groupe. De mon côté, j'ai tout de même essayé de comprendre le principe du décodage de la cryptographie.

Pour commencer à appréhender la cryptographie, j'ai travaillé sur une liste de bits que j'ai créée.

```
image_codée = [0, 1, 0, 1, 1, 0, 1, 0]
```

J'ai ensuite généré une clé de codage aléatoire de la même taille que mon image. Afin de construire cette dernière, j'ai utilisé le module random et la fonction random.randint(x,y) qui permet de donner un chiffre aléatoire entre x et y. Afin de bien respecter la règle du bipixel clé, j'ai ensuite ajouté, à la suite de ma clé, l'inverse de mon chiffre aléatoire.

```
import random          #importation du module random

i = 0                  #on initialise une valeur i à 0
image_codée = [0, 1, 0, 1, 1, 0, 1, 0]          #image codée
cle = []              #initialisation de la cle

#generation aleatoire de la cle avec le bipixel cle
while i < len(image_codée)/2:                    #tant qu'on est pas a la fin de la liste divisee par 2
    alea = random.randint(0,1)                  #on choisit un chiffre aleatoirement entre 0 et 1
    cle.append(alea)                            #on l'ajoute dans cle
    if alea == 0:                               # si alea = 0, on ajoute son inverse 1
        cle.append(1)
    elif alea == 1:                             #si alea = 1, on ajoute son inverse 0
        cle.append(0)
    i +=1                                       #on incremente la valeur de i de +1
print(cle)
```

Enfin, afin d'obtenir la valeur du pixel original, il faut réaliser un OU exclusif entre l'image codée et la clé de codage. Pour se faire, j'ai utilisé la fonction Xor, qui est désignée sur Python par x^y . J'ai utilisé cette fonction sur le premier pixel de chaque binôme. En fonction du résultat du OU exclusif entre les 2 valeurs, j'ajoute un 1 ou un 0 à l'image décodée : si $x^y = 0$, alors on insère un 1, sinon si $x^y = 1$, alors on insère un 0.

```
image_decodée = []          #initialisation image_decodée
i = 0                       #on remet i à 0

while i < len(image_codée):  #tant qu'on est pas a la fin de la liste
    if image_codée[i] ^ cle[i] == 0:          #si OU exclusive entre l'image et la cle est egal a 0 alors 1
        image_decodée.append(1)
    elif image_codée[i] ^ cle[i] == 1:        #si OU exclusive entre l'image et la cle est egal a 1 alors 0
        image_decodée.append(0)
    i += 2                                    #on incremente la valeur de i de +2
print(image_decodée)                    #on affiche l'image decodée
```

Evidemment, ce code ne permet pas encore de décoder n'importe quelle image qui aurait été codée préalablement, mais je pense avoir compris au moins le principe du décodage de la cryptographie. Je regrette que nous n'ayons pas eu le temps de finaliser la cryptographie et j'essaierai à l'avenir d'être mieux organisée dans un travail de groupe afin de pouvoir

terminer le projet.

Interface Tkinter

Tkinter est un module intégré à la bibliothèque standard de Python, permettant de créer des interfaces graphiques. J'ai donc commencé par importer la bibliothèque de Tkinter.

```
from tkinter import *
```

Avant de commencer à commenter mon code, notez que sur Tkinter, pour afficher un bouton, un titre, un cadre ou une entrée, il faut faire `.pack()`, ainsi que `fenetre.mainloop()` qui annonce le démarrage de la boucle Tkinter.

J'ai donc commencé par créer la première fenêtre à laquelle j'ai assigné un titre : « Choisis ton codage » et une couleur. J'ai ensuite mis `value = StringVar()`, ceci permet de lier de façon dynamique le contenu de l'entrée au reste du programme. J'ai ensuite créé un cadre pour la fenêtre et deux boutons : le **bouton1** pour le choix de la *Stéganographie* et le **bouton2** pour le choix de la *Cryptographie*.

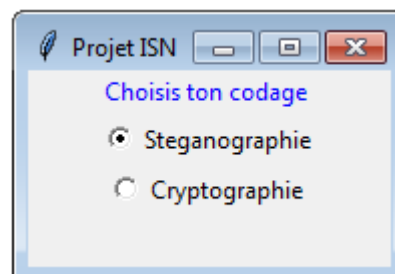
```
fenetre = Tk()
fenetre.title("Projet ISN")
titre = Label(fenetre, text = "Choisis ton codage ", fg = 'blue') #titre
titre.pack()

cadre = Frame(fenetre) #Cadre pour la fenêtre
cadre.pack() #Affichage du cadre

bouton1 = Radiobutton(fenetre, text="Steganographie", variable = value, value = 0,
command = choix1) #bouton pour le choix Steganographie
bouton2 = Radiobutton(fenetre, text="Cryptologie", variable = value, value = 1,
command = choix2) #bouton pour le choix Cryptologie

bouton1.pack()#Affichage du bouton1
bouton2.pack()#Affichage du bouton2
bouton1.select()

fenetre.mainloop() #Démarrage de la boucle Tkinter
```



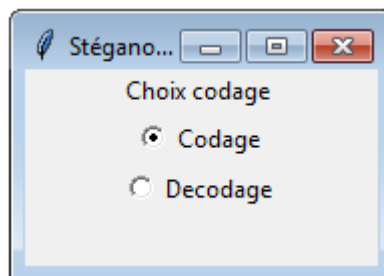
Stéganographie

J'ai assigné au **bouton1** la commande **choix1** `command = choix1` . Cette fonction va permettre à l'utilisateur de choisir s'il veut coder ou décoder de la Stéganographie. J'ai donc défini la fonction **choix1**. A l'intérieur de cette dernière, j'ai créé une nouvelle fenêtre, un nouveau cadre, ainsi qu'un titre « Choix codage/décodage ». J'ai aussi créé deux boutons :
_le **bouton1** pour le choix *Codage*

_le **bouton2** pour le choix *Décodage*

```
def choix1():
    fenetre1 = Tk()
    fenetre1.title("Stéganographie")
    titre = Label(fenetre1, text = "Choix codage ")
    titre.pack()
    cadre = Frame(fenetre1)
    cadre.pack()
    bouton1 = Radiobutton(fenetre1, text="Codage", variable = value, value = 0, command =
recuperer11) #bouton pour le choix Codage
    bouton2 = Radiobutton(fenetre1, text="Decodage", variable = value, value = 1, command
= recuperer12) #bouton pour le choix Decodage
    bouton1.pack()
    bouton2.pack()
    bouton1.select()
    fenetre1.mainloop()
```

J'ai assigné au **bouton1** la commande **recupere11** `command = recuperer11` . Cette fonction est pour le codage de la stéganographie. J'ai ensuite assigné au **bouton2** la fonction **recupere12**. Cette fonction représente le décodage de la stéganographie.



Codage Stéganographie

J'ai donc défini la fonction **recupere11**, dans laquelle j'ai créé une nouvelle fenêtre et son titre « Codage Stéganographie », ainsi qu'un nouveau cadre. J'ai ensuite mis deux titres :

_titre1 « Insérer l'url de l'image conteneur » , sous lequel j'ai mis **entree1** composé d'un cadre où l'utilisateur pourra rentrer l'URL. `entree1 = Entry(cadre, width=30)`

_titre2 « Insérer l'url de l'image à cacher », sous lequel j'ai mis **entree2** composé d'un cadre où l'utilisateur pourra rentrer l'URL. `entree2 = Entry(cadre, width=30)`

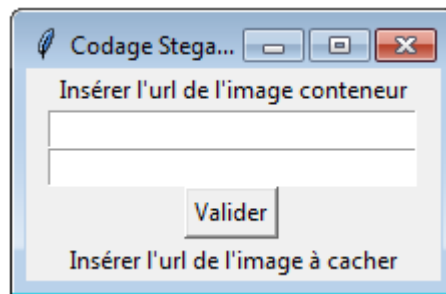
J'ai ensuite ajouté un bouton **valider** qui permet d'effectuer le codage sur l'image à coder en **entree1** et l'image conteneure en **entree2**.

```
valider = Button(cadre, text="Valider", command= lambda: codage_stega(entree1, entree2))
```

```

def recupere11(): #Codage Steganographie          pour recuperer les images telecharger
    fenetre2 = Tk()
    fenetre2.title("Codage Steganographie")
    titre1 = Label(fenetre2, text = "Insérer l'url de l'image conteneur")
    titre1.pack()
    cadre = Frame(fenetre2)
    cadre.pack()
    entree1 = Entry(cadre, width=30)
    entree1.pack()
    titre2 = Label(fenetre2, text = "Insérer l'url de l'image à cacher")
    titre2.pack()
    entree2 = Entry(cadre, width=30)
    entree2.pack()
    valider = Button(cadre, text="Valider", command= lambda: codage_stega(entree1, entree2))
    valider.pack()
    fenetre2.mainloop()

```



Décodage stéganographie

J'ai donc défini la fonction **recupere12**, dans laquelle j'ai créé une nouvelle fenêtre et son titre « Décodage Stéganographie », ainsi qu'un nouveau cadre. J'ai ensuite mis un titre, **titre1** « Insérer l'url de l'image à décoder », sous lequel j'ai mis **entree1** composé d'un cadre où l'utilisateur pourra rentrer l'URL. `entree1 = Entry(cadre, width=30)`

J'ai ensuite ajouté un bouton **valider** qui permet d'effectuer le décodage sur l'image à décoder en **entree1**.

```

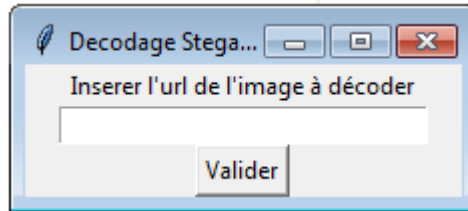
valider = Button(cadre, text="Valider", command= lambda: decodage_stega(entree1))

```

```

def recupere12(): #Decodage Steganographie          pour recuperer les images telecharger
    fenetre2 = Tk()
    fenetre2.title("Decodage Steganographie ")
    titre1 = Label(fenetre2, text = "Insérer l'url de l'image à décoder")
    titre1.pack()
    cadre = Frame(fenetre2)
    cadre.pack()
    entree1 = Entry(cadre, width=30)
    entree1.pack()
    valider = Button(cadre, text="Valider", command= lambda: decodage_stega(entree1))
    valider.pack()
    fenetre2.mainloop()

```



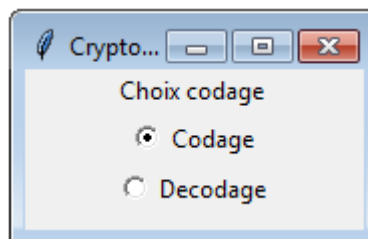
Cryptographie

J'ai assigné au **bouton1** la commande `choix2` `command = choix1`. Cette fonction va permettre à l'utilisateur de choisir s'il veut coder ou décoder de la cryptographie. J'ai donc défini la fonction **choix2**. A l'intérieur de cette dernière, j'ai créé une nouvelle fenêtre, un nouveau cadre, ainsi qu'un titre « Choix codage/décodage ». J'ai aussi créé deux boutons :

_le **bouton1** pour le choix *Codage*

_le **bouton2** pour le choix *Décodage*.

```
def choix2():
    fenetre1 = Tk()
    fenetre1.title("Cryptographie")
    titre = Label(fenetre1, text = "Choix codage ")
    titre.pack()
    cadre = Frame(fenetre1)
    cadre.pack()
    bouton1 = Radiobutton(fenetre1, text="Codage", variable = value, value = 0, command =
recupere21) #bouton pour le choix Codage
    bouton2 = Radiobutton(fenetre1, text="Decodage", variable = value, value = 1, command
= recupere22) #bouton pour le choix Decodage
    bouton1.pack()
    bouton2.pack()
    bouton1.select()
    fenetre1.mainloop()
```



J'ai assigné au **bouton1** la commande `recupere21`. Cette fonction est pour le codage de la cryptographie. J'ai ensuite assigné au **bouton2** la fonction `recupere22`. Cette fonction représente le décodage de la cryptographie.

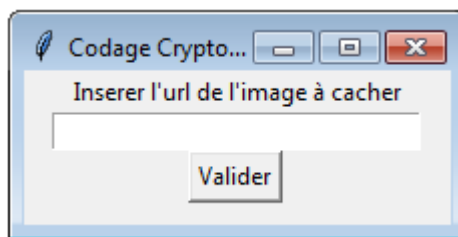
Codage cryptographie

J'ai donc défini la fonction **recupere21**, dans laquelle j'ai créé une nouvelle fenêtre et son titre « Codage Cryptographie », ainsi qu'un nouveau cadre. J'ai ensuite mis un titre, **titre** « Insérer l'URL de l'image à cacher », sous lequel j'ai mis **entree1** composé d'un cadre où l'utilisateur pourra rentrer l'URL. `entree1 = Entry(cadre, width=30)`

J'ai ensuite ajouté un bouton **valider** qui permet d'effectuer le codage sur l'image à cacher en **entree1**.

```
valider = Button(cadre, text="Valider", command= lambda: codage_crypto(entree1))
```

```
def recupere21():    #Codage crypto
    fenetre2 = Tk()
    fenetre2.title("Codage Cryptographie")
    titre = Label(fenetre2, text = "Inserer l'url de l'image à cacher")
    titre.pack()
    cadre = Frame(fenetre2)
    cadre.pack()
    entree1 = Entry(cadre, width=30)
    entree1.pack()
    valider = Button(cadre, text="Valider", command= lambda: codage_crypto(entree1))
    valider.pack()
    fenetre2.mainloop()
```



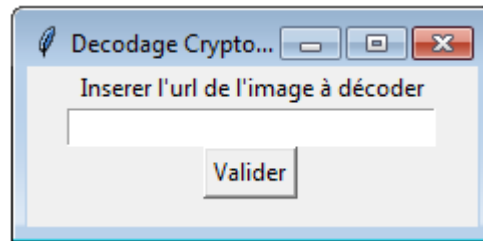
Décodage cryptographique

J'ai donc défini la fonction **recupere22**, dans laquelle j'ai créé une nouvelle fenêtre et son titre « Décodage Cryptographie », ainsi qu'un nouveau cadre. J'ai ensuite mis un titre, **titre** « Insérer l'url de l'image à décoder », sous lequel j'ai mis **entree1** composé d'un cadre où l'utilisateur pourra rentrer l'URL. `entree1 = Entry(cadre, width=30)`

J'ai ensuite ajouté un bouton **valider** qui permet d'effectuer le décodage sur l'image en **entree1**.

```
valider = Button(cadre, text="Valider", command= lambda: decodage_crypto(entree1))
```

```
def recupere22():    #Decodage crypto
    fenetre2 = Tk()
    fenetre2.title("Decodage Cryptographie")
    titre = Label(fenetre2, text = "Inserer l'url de l'image à décoder")
    titre.pack()
    cadre = Frame(fenetre2)
    cadre.pack()
    entree1 = Entry(cadre, width=30)
    entree1.pack()
    valider = Button(cadre, text="Valider", command= lambda:
decodage_crypto(entree1))
    valider.pack()
    fenetre2.mainloop()
```



Conclusion :

J'ai décidé de m'inscrire à cette option, Informatique et Sciences du Numérique dans le but de développer mes connaissances dans le domaine de l'informatique. Cet enseignement et ce projet m'ont permis de découvrir un monde numérique que je ne connaissais pas du tout. Cela nous a aussi appris à travailler en groupe et ce ne pourra être qu'un plus pour nos études et nos emplois futurs. Le projet nous a également permis de développer des capacités d'entreprenariat car nous devons d'être autonomes et productifs pour l'avancement du projet.

Le projet de stéganographie et cryptographie visuelles est un projet très intéressant. Cela m'a appris de nombreuses choses à propos de ces méthodes de codages, ainsi que sur Python. J'ai entre autre appris à manier Tkinter pour l'interface. Mais c'est un projet que l'on peut évidemment améliorer. A cause de la date butoir, je n'ai malheureusement pas eu le temps de terminer la cryptographie. Malgré cela, ma partie sur la stéganographie marche parfaitement bien et l'interface fonctionne également.

Annexes :

Décodage de la stéganographie

```
def decode_octet(octet_code):
    return (0b00001111 & octet_code) << 4
# print(bin(decode_octet(0b01101101)))

def decode_pixel(pixel_code):
    pixel_code = list(pixel_code)
    pixel_decode = []
    for octet in pixel_code:
        pixel_decode.append((0b00001111 & octet) << 4)
    return pixel_decode
# print(decode_pixel((140,213,109)))

image_codee =
imageio.imread("https://upload.wikimedia.org/wikipedia/commons/a/a8/Steganography_original.png")

import matplotlib.pyplot as plt

def decode_image(image_codee):

    largeur = image_codee.shape[0] #pour obtenir la valeur de la largeur
    hauteur = image_codee.shape[1] #pour obtenir la valeur de la hauteur
    #print(largeur, hauteur)

    for pixel_l in range(largeur): # pour chacun des pixels de la ligne
        for pixel_h in range(hauteur): # pour chacun des pixels de la colonne
            pixel_code = list(image_codee[pixel_l][pixel_h]) # donne les trois composantes
            pixel_decode = []
            for octet in pixel_code:
                pixel_decode.append((0b00000011 & octet) << 2) # mettre les bit de poids
            # fort à 0 et puis décaler de 4 rangs vers la gauche
            image_codee[pixel_l][pixel_h] = tuple(pixel_decode)
    plt.imshow(image_codee) # mettre l'image codee sous plt
    plt.show()

decode_image(image_codee) # afficher l'image décodée
```

Décodage de la cryptographie

```
import random          #importation du module random

i = 0                  #on initialise une valeur i à 0
image_codee = [0, 1, 0, 1, 1, 0, 1, 0]      #image codee
cle = []               #initialisation de la cle

#generation aleatoire de la cle avec le bipixel cle
while i < len(image_codee)/2:                #tant qu'on est pas a la fin de la liste divisee
    par 2
    alea = random.randint(0,1)               #on choisit un chiffre aleatoirement entre 0 et 1
    cle.append(alea)                          #on l'ajoute dans cle
    if alea == 0:                             # si alea = 0, on ajoute son inverse 1
        cle.append(1)
    elif alea == 1:                            #si alea = 1, on ajoute son inverse 0
        cle.append(0)
    i +=1                                     #on incremente la valeur de i de +1
print(cle)

image_decodee = []      #initialisation image_decodee
i = 0                   #on remet i à 0

while i < len(image_codee):    #tant qu'on est pas a la fin de la liste
    if image_codee[i] ^ cle[i] == 0:    #si OU exclusive entre l'image et la cle est
    egal a 0 alors 1
        image_decodee.append(1)
    elif image_codee[i] ^ cle[i] == 1:    #si OU exclusive entre l'image et la cle est
    egal a 1 alors 0
        image_decodee.append(0)
    i += 2                               #on incremente la valeur de i de +2
print(image_decodee)                #on affiche l'image decode
```

Interface TKInter

```

from tkinter import *
from tkinter.messagebox import *
from tkinter.filedialog import *
import imageio
import matplotlib.pyplot as plt

fenetre = Tk()
fenetre.title("Projet ISN")
titre = Label(fenetre, text = "Choisis ton codage ", fg = 'blue') #titre
titre.pack()

value = StringVar()

def codage_stega(entree1, entree2):
    image_conteneur = imageio.imread(entree1.get())
    image_secrete = imageio.imread(entree2.get())

    largeur = list(range(list(image_conteneur.shape)[0])) #pour obtenir la valeur de la
largeur
    hauteur = list(range(list(image_conteneur.shape)[1])) #pour obtenir la valeur de la
hauteur
    for pixel_l in largeur: #sur chaque ligne
        for pixel_h in hauteur: #sur chaque pixel de cette ligne
            pixel_1 = image_conteneur[pixel_l][pixel_h] #pour obtenir les valeurs
            pixel_2 = image_secrete[pixel_l][pixel_h]
            #fonction codage_pixel :
            pixel_c = list(pixel_1) #pour transformer les tuples en listes modifiables
            pixel_s = list(pixel_2)
            pixel_conteneur= [] #pixel conteneur une fois codé avec les bits de poids
faible à 0
            pixel_secret=[] #pixel secret une fois codé avec les bits de poids forts
décalés à gauche
            for octet in pixel_c: #pour coder chaque octet du pixel conteneur
                pixel_conteneur.append(int('11110000', 2)& octet) #mettre a zero le bit de
poids faible
            for octet in pixel_s: #pour coder chaque octet du pixel secret
                pixel_secret.append(octet>>4) #décalage du premier octet de quatre
colonnes vers la droite
            #etape 1 du codage finie
            pixel_code = list(range(len(pixel_conteneur)))
            for octet in pixel_code: #etape 2 du codage :
                pixel_code[octet] = pixel_conteneur[octet] | pixel_secret[octet] #OU
logique des deux octets
            image_conteneur[pixel_l][pixel_h] = tuple(pixel_code)
            plt.imshow(image_conteneur)
            plt.show()
            return imageio.imsave("image_conteneur.png", image_conteneur)

def decodage_stega(result):
    image_codee = imageio.imread(result.get())
    largeur = image_codee.shape[0] #pour obtenir la valeur de la largeur
    hauteur = image_codee.shape[1] #pour obtenir la valeur de la hauteur

    for pixel_l in range(largeur): # pour chacun des pixels de la ligne
        for pixel_h in range(hauteur): # pour chacun des pixels de la colonne
            pixel_code = list(image_codee[pixel_l][pixel_h]) # donne les trois composantes
RGB
            pixel_decode = []
            for octet in pixel_code:
                pixel_decode.append((0b00000011 & octet) << 2) # mettre les bit de poids
fort à 0 et puis décaler de 4 rangs vers la gauche
            image_codee[pixel_l][pixel_h] = tuple(pixel_decode)
            plt.imshow(image_codee) # mettre l'image codee sous plt
            plt.show()

def choix1():
    fenetre1 = Tk()
    fenetre1.title("Stéganographie")
    titre = Label(fenetre1, text = "Choix codage ")
    titre.pack()
    cadre = Frame(fenetre1)
    cadre.pack()
    bouton1 = Radiobutton(fenetre1, text="Codage", variable = value, value = 0, command =

```

```

recupere11) #bouton pour le choix Codage
    bouton2 = Radiobutton(fenetrel, text="Decodage", variable = value, value = 1, command
= recupere12) #bouton pour le choix Decodage
    bouton1.pack()
    bouton2.pack()
    bouton1.select()
    fenetrel.mainloop()

def choix2():
    fenetrel = Tk()
    fenetrel.title("Cryptographie")
    titre = Label(fenetrel, text = "Choix codage ")
    titre.pack()
    cadre = Frame(fenetrel)
    cadre.pack()
    bouton1 = Radiobutton(fenetrel, text="Codage", variable = value, value = 0, command =
recupere21) #bouton pour le choix Codage
    bouton2 = Radiobutton(fenetrel, text="Decodage", variable = value, value = 1, command
= recupere22) #bouton pour le choix Decodage
    bouton1.pack()
    bouton2.pack()
    bouton1.select()
    fenetrel.mainloop()

def recupere11(): #Codage Steganographie          pour recuperer les images telecharger
    fenetre2 = Tk()
    fenetre2.title("Codage Steganographie")
    titre1 = Label(fenetre2, text = "Insérer l'url de l'image conteneur")
    titre1.pack()
    cadre = Frame(fenetre2)
    cadre.pack()
    entreel = Entry(cadre, width=30)
    entreel.pack()
    titre2 = Label(fenetre2, text = "Insérer l'url de l'image à cacher")
    titre2.pack()
    entree2 = Entry(cadre, width=30)
    entree2.pack()
    valider = Button(cadre, text="Valider", command= lambda: codage_stega(entreel,
entree2))
    valider.pack()
    fenetre2.mainloop()

def recupere12(): #Decodage Steganographie          pour recuperer les images telecharger
    fenetre2 = Tk()
    fenetre2.title("Decodage Steganographie ")
    titre1 = Label(fenetre2, text = "Inserer l'url de l'image à décoder")
    titre1.pack()
    cadre = Frame(fenetre2)
    cadre.pack()
    entreel = Entry(cadre, width=30)
    entreel.pack()
    valider = Button(cadre, text="Valider", command= lambda: decodage_stega(entreel))
    valider.pack()
    fenetre2.mainloop()

def recupere21(): #Codage crypto
    fenetre2 = Tk()
    fenetre2.title("Codage Cryptographie")
    titre = Label(fenetre2, text = "Inserer l'url de l'image à cacher")
    titre.pack()
    cadre = Frame(fenetre2)
    cadre.pack()
    entreel = Entry(cadre, width=30)
    entreel.pack()
    valider = Button(cadre, text="Valider", command= lambda: codage_crypto(entreel))
    valider.pack()
    fenetre2.mainloop()

def recupere22(): #Decodage crypto
    fenetre2 = Tk()
    fenetre2.title("Decodage Cryptographie")
    titre = Label(fenetre2, text = "Inserer l'url de l'image à décoder")
    titre.pack()
    cadre = Frame(fenetre2)
    cadre.pack()
    entreel = Entry(cadre, width=30)
    entreel.pack()
    valider = Button(cadre, text="Valider", command= lambda: decodage_crypto(entreel))

```

```
        valider.pack()
        fenetre2.mainloop()

cadre = Frame(fenetre)
cadre.pack()

bouton1 = Radiobutton(fenetre, text="Steganographie", variable = value, value = 0, command
= choix1) #bouton pour le choix Steganographie
bouton2 = Radiobutton(fenetre, text="Cryptographie", variable = value, value = 1, command
= choix2) #bouton pour le choix Cryptographie

bouton1.pack()
bouton2.pack()
bouton1.select()

fenetre.mainloop()
```