Image Steganography

```
What is Steganography?
----------------------
Steganography is a technique of hiding secret data within an ordinary fila/image/vedio/message to avoid detection.

Why?
----
The advantage of steganography over cryptography is
    i. In cryptography information is encrypted in non readable format.
    ii. But in steganography we hide the existance of data.
There are various techinques to implement this like LSB coding, MSB coding, parity coding etc..

Here parity bit coding is used.

Sender side (Encrypting):
-------------------------
i. Read image name, message that is to be encoded, password and new imagename from the user.
ii. Convert data into binary and encode it in the new image

Receiver Side (Decrypting):
---------------------------
i. Read image name and password from the user
ii. Decode the image and print the secret message
```

```
How it works
------------
Show options i.e Encode and Decode

Encode the data :
================
Every byte of data is converted to its 8-bit binary code using ASCII values. Now pixels are read from left to right in
a group of 3 containing a total of 9 values. The first 8-values are used to store binary data. The value is made odd
if 1 occurs and even if 0 occurs.
For example :
-------------
```

Suppose the message to be hidden is "Hii" . Since the message is of 3-bytes, therefore, pixels required to encode the data is 3 x 3 = 9. Consider a 4 x 3 image with a total 12-pixels, which are sufficient to encode the given data.

Before encoding the message convert it into cipher text where key is current time in hours
Add password to the message and (key*7+7)

i.e
password + encrypted_message + (key*7+7)

Image data (Every pixcel is combination of (Red, Green, Blue) values)
[(27, 64, 164), (248, 244, 194), (174, 246, 250), (149, 95, 232),
(188, 156, 169), (71, 167, 127), (132, 173, 97), (113, 69, 206),
(255, 29, 213), (53, 153, 220), (246, 225, 229), (142, 82, 175)]

ASCII value of 'H' is 72 whose binary equivalent is 01001000.
Taking first 3-pixels (27, 64, 164), (248, 244, 194), (174, 246, 250) to encode. Now change the pixel to odd for 1 and even for 0. So, the modifies pixels are (26, 63, 164), (248, 243, 194), (174, 246, 250). Simmilarly do for all characters.

Create new image with new values

Decode the data
==========
Read image name with extension and password

Decode data from the image

For decoding the data reverse the process
if number is even bin_value = 0
else bin_value = 1

Now the data is decoded into plane text but it is non readable format

Compare the passwords if both are same decrypt the data with "key = (key - 7)//7"

if not instead of warning the user decrypt the data with "key = (key - 9)//9"

Display the message

NOTE:

-> Incase any errors occurs like file doesn't exist show warning with pop up window

In [15]:

```
from tkinter import *
from tkinter import messagebox as mb
from PIL import Image
import datetime
import hashlib
#hashlib module to convert password to hash
from random import randint
def generate_data(pixels, data):
    """This function takes two arguments
    1.pixels(It contains image data in pixcels in (Red, Greeen, Blue) format.
    2.Data i.e encrypted data.
    The objective of this function is to convert given data into binary and encode it into image pixcels.
    """
    data_in_binary = []
    #An empty list to store binary data of the ecncrypted message
    # Here each charccter is converted into binary and appended to empty
    try:
        for i in data:
            binary_data = format(ord(i), '08b')
            data_in_binary.append(binary_data)

        length_of_data = len(data_in_binary)
         #Create an iterable object of pixcels to encode data
        image_data = iter(pixels)
        #This for loop iterates lenght_of_data number of times
        for a in range(length_of_data):
            pixels = [val for val in image_data.__next__()[:3] + image_data.__next__()[:3] + image_data.__next__()[:3]]
            #This for loop iterates 8 to encode each character(which is in binary) into every three pixcels
            for b in range(8):
                if (data_in_binary[a][b] == '1') and (pixels[b] % 2 == 0):
                    if pixels[b] != 0:
                        pixels[b] -= 1
                    else:
                        pixels[b] += 1
                elif (data_in_binary[a][b] == '0') and (pixels[b] % 2 != 0):
                    pixels[b] -= 1

            if (length_of_data - 1) == a:
                if pixels[-1] % 2 == 0:
                    if pixels[-1] != 0:
                        pixels[-1] -= 1
```

```
                    else:
                        pixels[-1] += 1
            else:
                if (pixels[-1] % 2 != 0):
                    pixels[-1] -= 1

            pixels = tuple(pixels)

            yield pixels[:3]
            yield pixels[3:6]
            yield pixels[6:9]
    except:
        err = """Hey user!!!
        The image you have given cannot be used for secure main_encryption
        Please try again
        But I will generate a new image which cannot be used for decryption"""
        mb.showerror("Error", err)
        en_win.destroy()

def encrypt(text, pwd):
    """This function takes two arguments
    1. Text i.e The secret message that you want to transmit by encoding it into an image.
    2. pwd i.e The password which is used as a secret key to decrypt the data.
    The objective of this function is to convert plain text and convert it into cipher text.
    """
    n = datetime.datetime.now().hour
    cap=[chr(i) for i in range(ord('A'),ord('Z')+1)]
    sma=[chr(i) for i in range(ord('a'),ord('z')+1)]
    res = "" #To store cipher text(i.e encrpted data)
    for ele in text:
        if(ele in sma):
            temp = (sma.index(ele) + n)%26
            res += sma[temp]
        elif(ele in cap):
            temp = (cap.index(ele) + n)%26
            res += cap[temp]
        else:
            res += ele
    #Converting the password to hash and append it to cipher
    hash_object = hashlib.sha256(pwd.encode())
    pas = hash_object.hexdigest()
    return pas + "----" + res + "----" + str(n*7+7)
```

```
def decrypt(text, pwd):
    '''
    This function takes two arguments
    1. Text i.e the decoded text from the image
    2. pwd i.e Password that the user has given
    The objective of this function is decrypt the message as follows
    i. If password matches gives correct message
    ii. If does not mathes gives new wrong message everytime
    iii. If the given image does not contain any text it gives some random text
    '''
    lis = text.split("----")
    hash_object = hashlib.sha256(pwd.encode())
    pas = hash_object.hexdigest()
    fake = [i for i in range(1,20) if i != 7]
    temp = fake[randint(0,len(fake)-1)]
    if len(lis) == 1:
        st = "Hey User! This image is not encoded with any data"
        n = abs((randint(100,999)-temp)//temp)
    elif len(lis) == 3 and lis[0]==pas:
        st = lis[1]
        n = (int(lis[-1])-7)//7
    else:
        st = lis[1]
        n = abs((int(lis[-1])-temp)//temp)
    cap=[chr(i) for i in range(ord('A'),ord('Z')+1)]
    sma=[chr(i) for i in range(ord('a'),ord('z')+1)]
    res = ""
    for ele in st:
        if ele in sma:
            temp = (sma.index(ele) - n) % 26
            res += sma[temp]
        elif(ele in cap):
            temp = (cap.index(ele) - n) % 26
            res += cap[temp]
        else:
            res += ele
    return res

def encryption(img, data):
    # This method will encode data to the new image that will be created
    si = img.size[0]
```

```
    (x, y) = (0, 0)

    for pixel in generate_data(img.getdata(), data):
        img.putpixel((x, y), pixel)
        if si - 1 == x:
            x = 0; y += 1
        else:
            x += 1

def main_encryption(img, message, new_image_name, pwd):
    # This function will take the arguments, create a new image, encode it and save it to the same directory
    if (len(message) == 0) or (len(img) == 0) or (len(new_image_name) == 0):
        mb.showerror("Error", 'Please make sure that you have given all values and no field is left empty')
        en_win.destroy()
        return
    try:
        image = Image.open(img, 'r')
    except IOError:
        mb.showerror("Image invalid", "Please enter image with valid path and extension")
        en_win.destroy()

    text = encrypt(message, pwd)
    new_image = image.copy()
    encryption(new_image, text)
    new_image_name += '.png'
    new_image.save(new_image_name, 'png')
    mb.showinfo("Success", "Your data is succesfully encrypted!!!\n Your new_image name is extended with .png")
    en_win.destroy()

def main_decryption(img, strvar, pwd):
    # This function will decode the image given to it and extract the hidden message from it
    try:
        image = Image.open(img, 'r')
    except IOError:
        mb.showerror("Invalid Image","Please check the path and extension")
        de_win.destroy()

    data = ''
    image_data = iter(image.getdata())

    while True:
        try:
```

```
            pixels = [value for value in image_data.__next__()[:3] + image_data.__next__()[:3] + image_data.__next__()[:
        except:
            mb.showinfo("Warn","This type of images cannot be decoded")
            de_win.destroy()
            return
        else:
            binary_string = ''

            for i in pixels[:8]:
                if i % 2 == 0:
                    binary_string += '0'
                else:
                    binary_string += '1'

            data += chr(int(binary_string, 2))
            if pixels[-1] % 2 != 0:
                strvar.set(decrypt(data, pwd))
                break

def encode():
    global en_win
    en_win = Toplevel(win)
    en_win.geometry('650x280')
    en_win.title("encode")
    en_win.resizable(0,0)
    en_win.config(bg = "#d1d44c")
    Label(en_win, text = "Keep your data safe", font =("Comic Sans MS", 15), bg='AntiqueWhite').place(x=210, y=10)
    Label(en_win, text='Enter the path to the image(with extension):', font=("Times New Roman", 13),
          bg='AntiqueWhite').place(x=10, y=50)
    Label(en_win, text='Enter the data to be encoded:', font=("Times New Roman", 13), bg='AntiqueWhite').place(
        x=10, y=90)
    Label(en_win, text='Enter password for encryption:', font=("Times New Roman", 13), bg='AntiqueWhite').place(
        x=10, y=130)
    Label(en_win, text='Enter the output file name (without extension):', font=("Times New Roman", 13),
          bg='AntiqueWhite').place(x=10, y=170)

    img_path = Entry(en_win, width=45,textvariable="abc.png", state="normal", font =( "Times New Roman", 9, "bold"))
    img_path.insert(0,r"C:\Users\My_pc\Desktop\image1.jpg")
    img_path.place(x=350, y=50)

    text_to_be_encoded = Entry(en_win, width=45,font =( "Times New Roman", 9, "bold"))
    text_to_be_encoded.insert(0,"This is my secret Message")
```

```
    text_to_be_encoded.place(x=350, y=90)

    pwd = Entry(en_win,width=45,show = "*",font =( "Times New Roman", 9, "bold"))
    pwd.insert(0,"Password")
    pwd.place(x = 350, y = 130)

    after_save_path = Entry(en_win, width=45,font =( "Times New Roman", 9, "bold"))
    after_save_path.insert(0,r"C:\Users\My_pc\Desktop\new_image")
    after_save_path.place(x=350, y=170)

    if img_path.get() != None and text_to_be_encoded.get() != None and after_save_path.get() != None:
        Button(en_win, text='Encode the Image', font=('Helvetica', 12), bg='PaleTurquoise', command=lambda:
        main_encryption(img_path.get(), text_to_be_encoded.get(), after_save_path.get(), pwd.get())).place(x=220, y=215)
    else:
        mb.showerror("Warning", "None of the fields should be empty!!!")
        en_win.destroy()
def decode():
    global de_win
    de_win = Toplevel(win)
    de_win.geometry('680x380')
    de_win.title("encode")
    de_win.resizable(0,0)
    de_win.config(bg = "#4cd273")

    Label(de_win, text = "Decode the message", font=("Comic Sans MS", 15), bg = "#4cd273").place(x = 220, y = 10)

    Label(de_win, text='Enter the path to the image (with extension):', font=("Times New Roman", 12),
          bg='#4cd273').place(x=10, y=50)
    Label(de_win, text='Enter password for decryption:', font=("Times New Roman", 12),
          bg='#4cd273').place(x=10, y=90)
    img_name = Entry(de_win, width=45,font =( "Times New Roman", 9, "bold"))
    img_name.insert(0,r"C:\Users\My_pc\Desktop\new_image.png")
    img_name.place(x = 310,y = 50)

    de_pass = Entry(de_win, width = 45, show = "*",font =( "Times New Roman", 9, "bold"))
    de_pass.insert(0,"Password")
    de_pass.place(x = 310, y = 90)

    text_strvar = StringVar()

    Button(de_win, text='Decode the data', font=('Helvetica', 12), bg='orange', command=lambda:
    main_decryption(img_name.get(), text_strvar, de_pass.get())).place(x=220, y=140)
```

```
    Label(de_win, text='The secret message is:', font=("Times New Roman", 15), bg='Bisque').place(
            x=200, y=190)
    text_entry = Entry(de_win, width=65, text=text_strvar, state='readonly',font=("Times New Roman", 15))
    text_entry.place(x=12, y=225, height=100)

win = Tk()
win.geometry('700x400')
win.title("Image Steganography")
win.resizable(0,0)
win.config(bg = '#1CF2E7')
info = r'''Hey Buddy!!!
This is Dev.
I am here to encode and decode your messages.
Please provide the required details in the corresponding fileds.
If you are a new user some default values are given to input fields.
Replace them with your details.
Note: If the files are on the same folder 'NO NEED TO GIVE PATH'
Thanks for choosing me to encode and decode your messages.
Have a Great Day!!!
'''
mb.showinfo("Read me", info)
Label(win, text = "Image Steganography", font =( 'Playfair Display', 25), bg = '#1CF2E7').place(x = 210, y = 20)
Button(win, text='Encode', width=25, font=('Times New Roman', 13), bg='SteelBlue', command=encode).place(
        x=240, y=100)
Button(win, text='Decode', width=25, font=('Times New Roman', 13), bg='SteelBlue', command=decode).place(
        x=240, y=160)
win.update()
win.mainloop()
```

In [11]:
```
import datetime
hour = datetime.datetime.now().hour
print(hour)
```

```
13
```

In [8]:

```python
from random import randint
res = []
lis = [i for i in range(1,20) if i != 7]
for a in range(1,23):
    enc = a*7+7
    for i in range(len(lis)):
        te = lis[i]
        dec = (enc - te)//te
        res.append(enc == dec)
```

In [9]:

```python
print(res)
```

```
[False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False]
```