**STEGANOGRAPHY WITH IMAGES**

A REPORT

Submitted by

**K. TARUN PRABHATH (17MIS1078)**

**COURSE NAME:** INFORMATION AND SYSTEM SECURITY

**COURSE CODE:** SWE3002

**PROF:** PUNITHA. K

**SLOT:** A1

in partial fulfilment for the award

of

**M. Tech. Software Engineering (Integrated)**

**School of Computer Science and Engineering**



**OCTOBER 2020**

**Objective:**

The secret data can be data of any format like text or even a file. In a nutshell, the main motive of steganography is to hide the intended information within any file, usually an image, audio, or video, without actually changing the external appearance of the file, i.e. it should look the same as before.

**Introduction:**

Steganography is a technique of hiding information behind the scene. It’s is not like cryptography which focuses on encrypting data, steganography focuses more on hiding the data (data can be a file, image, message or video) within another file, image, message or video to avoid any attraction. Image

Steganography- Images are composed of digital data which describes what’s inside the picture, usually the colors of all the pixels. Since we know every image is made up of pixels and every pixel contains 3-values (red, green, blue).

Image comprises of Pixels are the building blocks of an image. Every pixel contains three values: also known as RGB values. Every RGB value ranges from 0 to 255. The object of steganography is to send a message through some innocuous carrier. Computer based steganography allows changes to be made to what are known as digital carriers such as images or sounds. The changes represent the hidden message but result if successful in no discernible change to the carrier. The information may be nothing to do with the carrier sound or image or it might be information about the carrier such as the author or a digital watermarking or fingerprint.

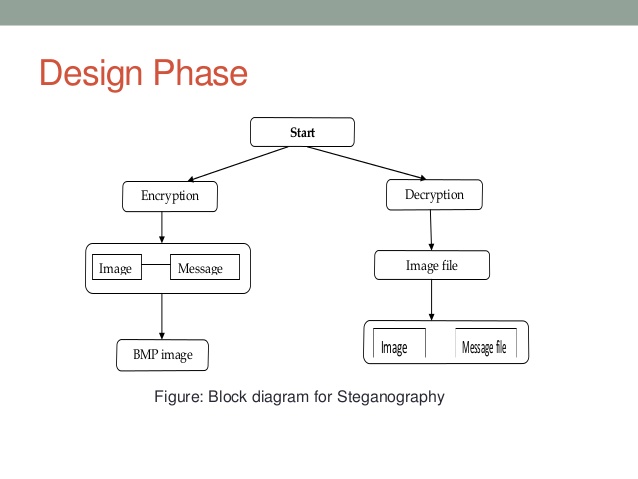
When message is hidden in the carrier a stego carrier is formed foe example a stego-image. Hopefully it will be perceived to be as close as possible to the original carrier or cover image by the human senses. Images are the most widespread carrier medium. They are used for steganography in the following way. The message may firstly be encrypted. The sender embeds the secret message to be sent into a graphic file. This results in the production of what is called stego-image. Additional secret data may be needed in the hiding process e.g. a stegokey etc. this stego-image is then transmitted to the recipient.

The recipient extractor extracts the message from the carrier image. The message can only be extracted if there is shared secret between the sender and the recipient. This could be the algorithm for extraction or a special parameter such as stegokey. A stego analyst or attacker may try to intercept the stego image.

**Feasibility Study:**

We consider the question of whether steganography can protect the information what we need to hide from the unwanted persons. Here we are hiding the text because to share that secretly to the other end user without leaking any information in between. So, it will be challenging when we do that by using an image to hide the text. The image hiding is also challenging by using another image but through encryption techniques we can achieve that one and decryption to know the actual information in the output.

**System Analysis & Design:**  
  
Steganography system requires any type of image file and the information or message that is to be hidden. It has two modules encrypt and decrypt. The algorithm used for Encryption and Decryption in this application provides using several layers lieu of using only LSB layer of image. Writing data starts from last layer (8st or LSB layer); because significant of this layer is least and every upper layer has doubled significant from its down layer. So, every step we go to upper layer image quality decreases and image retouching transpires.  
  
The **encrypt module** is used to hide information into the image; no one can see that information or file. This module requires any type of image and message and gives the only one image file in destination.  
  
The **decrypt module** is used to get the hidden information in an image file. It takes the image file as an output, and give two files at destination folder, one is the same image file and another is the message file that is hidden it that.



**Steganography with images:**

**Encoding:**

There are a lot of algorithms that can be used to encode data into the image, and in fact, you can also make one yourself. The one being used in this blog is easy to understand and implement, as well. The algorithm is as follows:

1.For each character in the data, its ASCII value is taken and converted into 8-bit binary [1].

2.Three pixels are read at a time having a total of 3\*3=9 RGB values. The first eight RGB values are used to store one character that is converted into an 8-bit binary.

3.The corresponding RGB value and binary data are compared. If the binary digit is 1 then the RGB value is converted to odd and, otherwise, even.

4.The ninth value determines if more pixels should be read or not. If there is more data to be read, i.e. encoded or decoded, then the ninth pixel changes to even. Otherwise, if we want to stop reading pixels further, then make it odd.

5.Repeat this process until all the data is encoded into the image.

**Example**

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

[(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)]

***Step 1***

The ASCII value of H is 72, whose binary equivalent is 01001000.

***Step 2***

Read the first three pixels.

(27, 64, 164), (248, 244, 194), (174, 246, 250)

***Step 3***

Now, change the pixel value to odd for 1 and even for 0 as in the binary equivalent of data.

For example, the first binary digit is 0 and the first RGB value is 27, it needs to be converted to even, which implies 26.

Similarly, 64gets converted to 63because the next binary digit is 1so the RGB value should be made odd.

So, the modified pixels are:

(26, 63, 164), (248, 243, 194), (174, 246, 250)

***Step 4***

Since we have to encode more data, the last value should be even. Similarly, i can be encoded in

this image.

While making the pixel values odd/even by doing +1 or -1, you should take care of binary

conditions. I.e., the pixel value should be more than or equal to 0 and less than or equal to 255.

The new image will look like:

[(26, 63, 164), (248, 243, 194), (174, 246, 250), (148, 95, 231),  
(188, 155, 168), (70, 167, 126), (132, 173, 97), (112, 69, 206),  
(254, 29, 213), (53, 153, 220), (246, 225, 229), (142, 82, 175)]

**Decoding:**

For decoding, we shall try to find how to reverse the previous algorithm that we used to encode data.

1.Again, three pixels are read at a time. The first 8 RGB values give us information about the secret data, and the ninth value tells us whether to move forward or not.

2.For the first eight values, if the value is odd, then the binary bit is 1, otherwise it is 0.

3.The bits are concatenated to a string, and with every three pixels, we get a byte of secret data, which means one character.

4.Now, if the ninth value is even then we keep reading pixels three at a time, or otherwise, we stop.

**For example**

Let’s start reading three pixels at a time. Consider our previously encoded image.

[(26, 63, 164), (248, 243, 194), (174, 246, 250), (148, 95, 231),  
(188, 155, 168), (70, 167, 126), (132, 173, 97), (112, 69, 206),  
(254, 29, 213), (53, 153, 220), (246, 225, 229), (142, 82, 175)]

***Step 1***

We first read the three pixels:

[(26, 63, 164), (248, 243, 194), (174, 246, 250)

***Step 2***

Reading the first value: 26, which is even, therefore the binary bit is 0. Similarly, for 63, the binary bit is 1and for 164it is 0. This process continues until the eight RGB value.

***Step 3***

We finally get the binary value: 01001000 after concatenating all individual binary values. The final binary data corresponds to decimal value 72, and in ASCII, it represents the character H.

***Step 4***

Since the ninth value is even, we repeat the above steps. We stop when the ninth value encountered is odd.

**Hiding an Image with another Image:**

we will hide one image inside another and convert it into another image and then extract back both the images from the previous image.

The idea behind image-based Steganography is very simple. Images are composed of digital data (pixels), which describes what’s inside the picture, usually the colors of all the pixels. Since we know every image is made up of pixels and every pixel contains 3-values (red, green, blue).

For example, suppose we have to hide img2 in img1, where both img1 and img2 are NumPy nd array of pixel values. The size of img2 must be less than the size of img1. We are using color images; hence both will have 3 values (red, green, blue). Each pixel value varies from 0 to 255, so each pixel value is of 1 byte or 8 bits. Let img[i][j][l] be the pixel value at location (i, j) and of channel l where i varies from 0 to width and j varies from 0 to height and l varies from 0 to 2

**Encoding:**

Let img1[i][j][l] and img2[i][j][l] be some pixel value of each image. Let v1 be 8 bits binary representation of img1[i][j][l] and v2 be 8 bits binary representation of img2[i][j][l]. Therefore, v3=v1[:4]+v2[:4], where, v3 is the first 4 bits of v1 and v2. Then we assign img1[i][j][l] to v3.

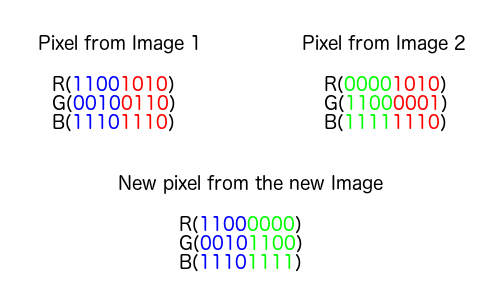


Figure2: Image hiding in another image

**Decoding:**

Let img[i][j][l] be the pixel value of the image. Let v1 be 8 bits binary representation of img[i][j][l].

Let v2=v1[:4] +4 random bits and v3=v1[4:] +4 random bits. Then we assign img1[i][j][l] to v2 and img2[i][j][l] to v3. Here img1 and img2 are the final images produced after decoding.

**Risk Analysis:**

Digital Forensic experts employ many techniques, and we will limit the discussion

to those applicable to steganography and steganalysis. In some cases, steganography software itself may be discovered on computer equipment under investigation. The Steganography Application Fingerprint Database currently contains identifying information on 625 applications

associated with steganography, watermarking, and other data-hiding applications. Similarly, the National Institute of Standards and Technology maintains a list of digital signatures in the National Software Reference Library, some of which are for steganography software. Even when

software has been removed; traces can sometimes be found in places like the Windows registry or in system backup files. When steganography software installation has been identified intent should be assumed until proven otherwise.

In addition to detecting the software used for steganography, digital forensics experts can detect files with similar visual properties but different file sizes, hash values, and statistical properties. If files have been deleted, they may be retrieved from the Recycle Bin or similar Trash container, or even reconstructed with special forensics tools for file recovery.

An additional method of detection uses a list of keywords to search for file names and content in program files and data files. The list should be specific with regard false positives and false negatives, depends on the quality of the keyword dictionary.

to steganography. For instance, the search term “steg\*” can be used to identify steganography. The effectiveness and efficiency in detection, while preventing

Though steganography tools may be used for legitimate business applications such as protecting strategic corporate information during transmission they have emerged as a significant issue to forensic investigators and others who are concerned with malicious and illegal uses. As steganography tools become more widely available and easier to use, protection against malicious use demands attention, and the balance between protection from illicit use and

interference with legitimate use emerges as a new challenge. In this section, we will focus on protection against malicious use, and not discuss specific potential business applications such as watermarking for protection of intellectual property.

**Results:**

This module is Steganography with images which will hide the text in the image and we can do this through encryption and to extract the text from the image again we need to perform the decryption technique to get the hidden text from image.

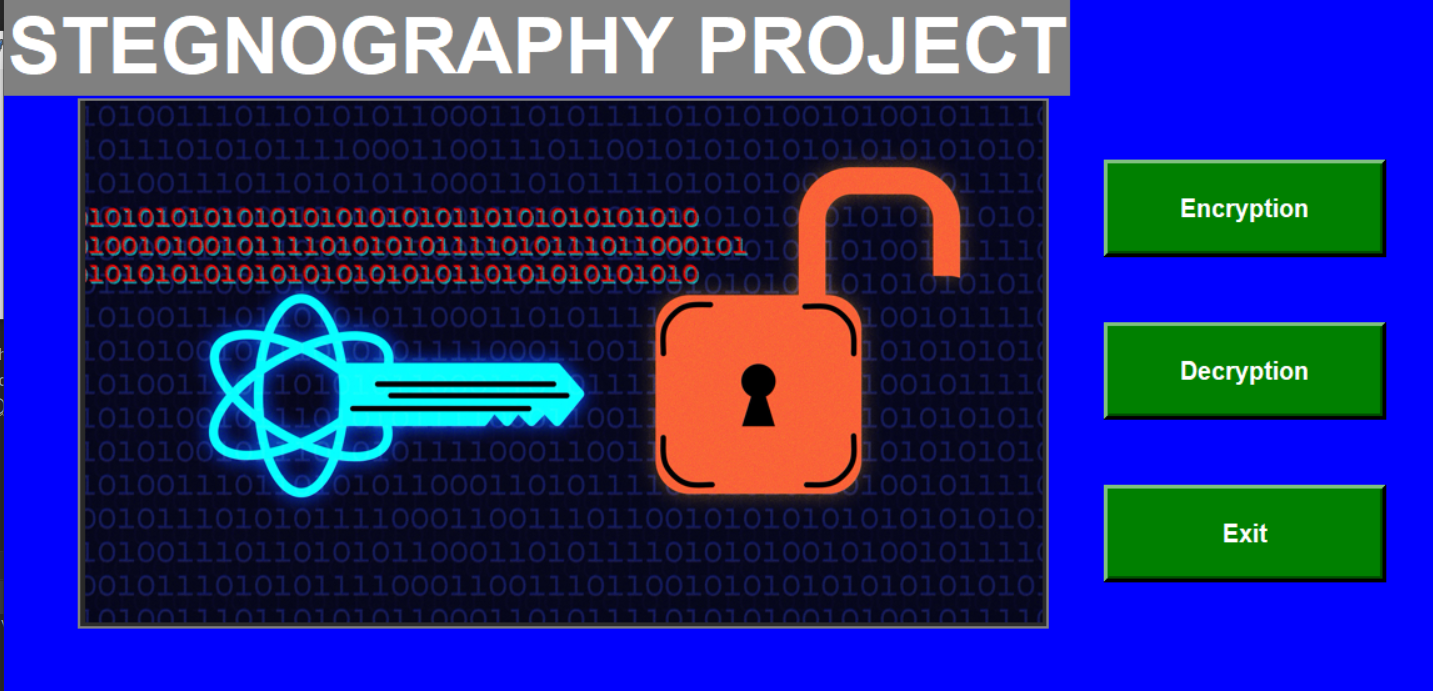


Figure3: Homepage of Steganography with images

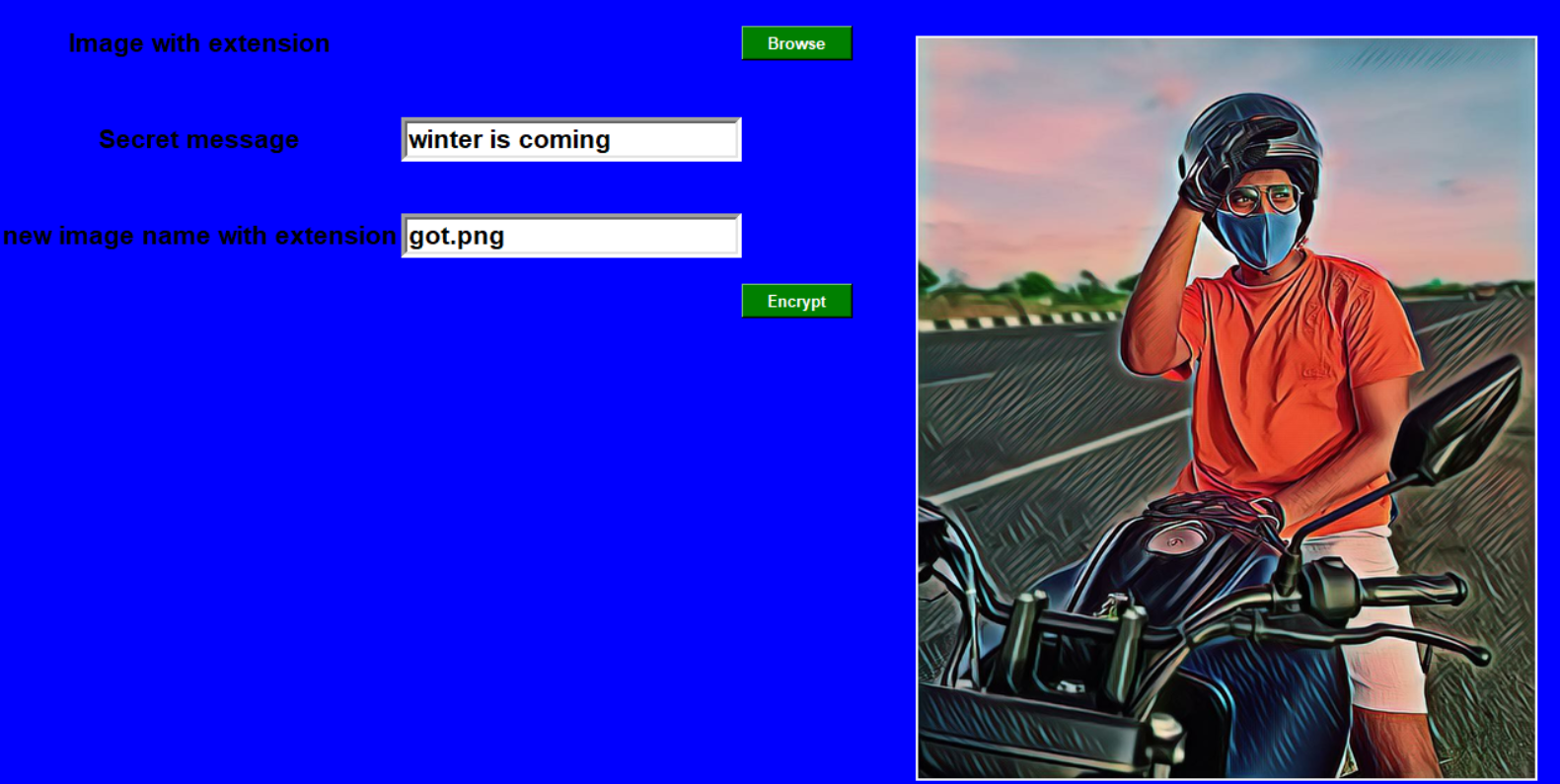


Figure4: Encryption technique



Figure5: Decryption technique

**Image hiding:** This module is Steganography with images in which will hide an image in another image and we can do this through encryption and to extract the hidden image from the encrypted image again, we then need to perform the decryption technique to get the hidden the image from image.

Figure6: Input image Figure7: Hidden image



Figure8: Encrypted image

**Conclusion:**

In this project we are hiding the text and image through steganography algorithms using encryption and retrieve them back by decrypting the encrypted files. Thus, we can securely transfer the information to a person without any loss of data and stolen of data.

**References:**

https://towardsdatascience.com/steganography-hiding-an-image-inside-another-77ca66b2acb1

https://www.researchgate.net/publication/314116270\_Image\_Steganography

http://www.ijettjournal.org/volume-4/issue-7/IJETT-V4I7P186.pdf

https://www.researchgate.net/publication/306301164\_Analysis\_of\_Different\_Text\_Steganography\_Techniques\_A\_Survey.

**Appendix:**

from tkinter import \*  
from tkinter import filedialog  
from PIL import ImageTk  
import encrypt  
import decrypt1  
def hide(root):  
 root.withdraw()  
  
def show(root):  
 root.update()  
 root.deiconify()  
  
def home():  
 root = Toplevel()  
 hide(root)  
 root.title("STEGNOGRAPHY")  
 root.geometry("1360x650+0+0")  
 root.config(bg="blue")  
 label = Label(root, text="STEGNOGRAPHY PROJECT", font=("araial", 47, "bold"), fg="white", bg="grey").grid(  
 row=0, column=0)  
  
 photo2 = ImageTk.PhotoImage(file='crypto15.png')  
 photo = Label(root, image=photo2, bg='grey')  
 photo.place(relx=0.05, rely=0.450, anchor=W)  
  
  
  
 def Exit():  
 root.destroy()  
  
  
 def ad():  
 encrypt.encode()  
  
 def bd():  
 decrypt1.main()  
  
 btnexit = Button(root, text="Exit", padx=16, pady=16, bd=5, bg='green', fg='white', font=('arial', 15, 'bold'),  
 width=15, height=1,command=Exit).place(relx=0.75, rely=0.600)  
 btnencry = Button(root, text="Encryption", padx=16, pady=16, bd=5, bg='green', fg='white',  
 font=('arial', 15, 'bold'),  
 command=ad, width=15, height=1).place(relx=0.75, rely=0.200)  
 btndecry = Button(root, text="Decryption", padx=16, pady=16, bd=5, bg='green', fg='white',  
 font=('arial', 15, 'bold'),  
 command=bd, width=15, height=1).place(relx=0.75, rely=0.400)  
  
 show(root)  
 root.mainloop()  
home()

**Encrypt:**

from tkinter import \*  
from tkinter import filedialog  
from PIL import Image  
from PIL import ImageTk, Image  
  
  
# Convert encoding data into 8-bit binary  
# form using ASCII value of characters  
image = Image.open("crypto15.PNG", 'r')  
data = "lhjknxz"  
  
  
def hide(root):  
 root.withdraw()  
  
  
def show(root):  
 root.update()  
 root.deiconify()  
  
def genData(data):  
 # list of binary codes  
 # of given data  
 newd = []  
  
 for i in data:  
 newd.append(format(ord(i), '08b'))  
 return newd  
  
  
# Pixels are modified according to the  
# 8-bit binary data and finally returned  
def modPix(pix, data):  
 datalist = genData(data)  
 lendata = len(datalist)  
 imdata = iter(pix)  
  
 for i in range(lendata):  
  
 # Extracting 3 pixels at a time  
 pix = [value for value in imdata.\_\_next\_\_()[:3] +  
 imdata.\_\_next\_\_()[:3] +  
 imdata.\_\_next\_\_()[:3]]  
  
 # Pixel value should be made  
 # odd for 1 and even for 0  
 for j in range(0, 8):  
 if (datalist[i][j] == '0') and (pix[j] % 2 != 0):  
  
 if (pix[j] % 2 != 0):  
 pix[j] -= 1  
  
 elif (datalist[i][j] == '1') and (pix[j] % 2 == 0):  
 pix[j] -= 1  
  
 # Eigh^th pixel of every set tells  
 # whether to stop ot read further.  
 # 0 means keep reading; 1 means the  
 # message is over.  
 if (i == lendata - 1):  
 if (pix[-1] % 2 == 0):  
 pix[-1] -= 1  
 else:  
 if (pix[-1] % 2 != 0):  
 pix[-1] -= 1  
  
 pix = tuple(pix)  
 yield pix[0:3]  
 yield pix[3:6]  
 yield pix[6:9]  
  
  
def encode\_enc(newimg, data):  
 w = newimg.size[0]  
 (x, y) = (0, 0)  
  
 for pixel in modPix(newimg.getdata(), data):  
  
 # Putting modified pixels in the new image  
 newimg.putpixel((x, y), pixel)  
 if (x == w - 1):  
 x = 0  
 y += 1  
 else:  
 x += 1  
  
  
# Encode data into image  
def encode():  
 root = Toplevel()  
  
 hide(root)  
 root.title("Encryption")  
 root.geometry("1350x650+0+0")  
 root.config(bg="blue")  
  
 label0 = Label(root, text="Image with extension", font=("araial", 16, "bold"), fg="black",bg="blue", height=3).grid(row=5,column=0)  
  
 def browse():  
 img = filedialog.askopenfilename(title='Select the image')  
 global image  
 image = Image.open(img, mode='r')  
 image1 = image.resize((500, 600), Image.ANTIALIAS)  
 image1 = ImageTk.PhotoImage(image1)  
 panel = Label(root, image=image1)  
 panel.image = image1  
 panel = panel.place(relx=0.550, rely=0.050)  
 return image  
  
 browse = Button(root, text="Browse", bg='green', fg='white', font=('arial', 10, 'bold'),  
 width=10, height=1, command=browse).grid(row=5, column=2)  
  
  
  
 label0 = Label(root, text="Secret message", font=("araial", 16, "bold"),bg="blue", fg="black", height=3).grid(row=7,column=0)  
  
 dataen = Entry(root, font=('arial', 16, 'bold'),bd=5, width=22, justify='left')  
 dataen.grid(row=7, column=1)  
  
  
 label0 = Label(root, text="new image name with extension", font=("araial", 16, "bold"), fg="black",bg="blue",  
 height=3).grid(row=11, column=0)  
 new\_img = Entry(root, font=('arial', 16, 'bold'), bd=5, width=22, justify='left')  
 new\_img.grid(row=11, column=1)  
  
 def encrypt():  
 global data  
 data = str(dataen.get())  
 if (len(data) == 0):  
 raise ValueError('Data is empty')  
  
 global image  
 newimg = image.copy()  
 encode\_enc(newimg, data)  
 new\_img\_name = str(new\_img.get())  
  
 newimg.save("C:/Users/admin/Desktop/iss project/" + new\_img\_name, str(new\_img\_name.split(".")[1].upper()))  
  
 ency = Button(root, text="Encrypt", bg='green', fg='white', font=('arial', 10, 'bold'),  
 width=10, height=1, command=encrypt).grid(row=13, column=2)  
  
 show(root)  
 root.mainloop()

**Decrypt:**

import tkinter as tk  
from tkinter import \*  
from tkinter import filedialog  
from PIL import Image  
from PIL import ImageTk, Image  
  
image = Image.open("new.png")  
root = Toplevel()  
data1 = ''  
def genData(data):  
  
 newd = []  
  
 for i in data:  
 newd.append(format(ord(i), '08b'))  
 return newd  
  
def modPix(pix, data):  
 datalist = genData(data)  
 lendata = len(datalist)  
 imdata = iter(pix)  
  
 for i in range(lendata):  
  
 # Extracting 3 pixels at a time  
 pix = [value for value in imdata.\_\_next\_\_()[:3] +  
 imdata.\_\_next\_\_()[:3] +  
 imdata.\_\_next\_\_()[:3]]  
  
 # Pixel value should be made  
 # odd for 1 and even for 0  
 for j in range(0, 8):  
 if (datalist[i][j] == '0') and (pix[j] % 2 != 0):  
  
 if (pix[j] % 2 != 0):  
 pix[j] -= 1  
  
 elif (datalist[i][j] == '1') and (pix[j] % 2 == 0):  
 pix[j] -= 1  
  
 # Eigh^th pixel of every set tells  
 # whether to stop ot read further.  
 # 0 means keep reading; 1 means the  
 # message is over.  
 if (i == lendata - 1):  
 if (pix[-1] % 2 == 0):  
 pix[-1] -= 1  
 else:  
 if (pix[-1] % 2 != 0):  
 pix[-1] -= 1  
  
 pix = tuple(pix)  
 yield pix[0:3]  
 yield pix[3:6]  
 yield pix[6:9]  
  
  
def encode\_enc(newimg, data):  
 w = newimg.size[0]  
 (x, y) = (0, 0)  
  
 for pixel in modPix(newimg.getdata(), data):  
  
 # Putting modified pixels in the new image  
 newimg.putpixel((x, y), pixel)  
 if (x == w - 1):  
 x = 0  
 y += 1  
 else:  
 x += 1  
  
  
# Decode the data in the image  
def decode(image):  
  
 data = ''  
 imgdata = iter(image.getdata())  
  
 while (True):  
 pixels = [value for value in imgdata.\_\_next\_\_()[:3] +  
 imgdata.\_\_next\_\_()[:3] +  
 imgdata.\_\_next\_\_()[:3]]  
 # string of binary data  
 binstr = ''  
  
 for i in pixels[:8]:  
 if (i % 2 == 0):  
 binstr += '0'  
 else:  
 binstr += '1'  
  
 data += chr(int(binstr, 2))  
 if (pixels[-1] % 2 != 0):  
 return data  
  
 # Main Function  
  
  
def main():  
 root = Toplevel()  
 root.title("Decryption")  
 root.geometry("1350x650+0+0")  
 root.config(bg="blue")  
  
 def browse():  
 img = filedialog.askopenfilename(title='Select the image')  
 global image  
 image = Image.open(img, mode='r')  
 image1 = image.resize((500,500), Image.ANTIALIAS)  
 image1 = ImageTk.PhotoImage(image1)  
 panel = Label(root, image=image1)  
 panel.image = image1  
 panel = panel.place(relx=0.050, rely=0.050)  
 browse = Button(root, text="Browse", padx=16,pady=16,fg='white', bg='green', font=('arial', 10, 'bold'),  
 width=10, height=1, command=browse).place(relx=0.75, rely=0.200)  
 def temp():  
 global data1  
 data1 = decode(image)  
 print(data1)  
 label0 = Label(root, text="Secret Message - "+data1, font=("araial", 16, "bold"), fg="white", bg="blue",  
 height=3).place(relx=0.50, rely=0.600)  
  
 decrypt = Button(root, text="Decrypt",padx=16,pady=16, fg='white', bg='green', font=('arial', 10, 'bold'),  
 width=10, height=1, command=temp).place(relx=0.75, rely=0.400)  
  
  
 root.mainloop()

**Image Hiding Code:**

from tkinter import filedialog  
  
import cv2  
import numpy as np  
import random  
  
  
# Encryption function  
def encrypt():  
 # img1 and img2 are the  
 # two input images  
 #img1 = cv2.imread('pic1.jpg')  
 #img2 = cv2.imread('pic2.jpg')  
 im1 = filedialog.askopenfilename()  
 l1 = im1.split('/');  
 img1 = cv2.imread(l1[len(l1) - 1]);  
 im2 = filedialog.askopenfilename()  
 l2 = im2.split('/');  
 img2 = cv2.imread(l2[len(l2) - 1]);  
 print(img1);  
 print(img2);  
  
 for i in range(img2.shape[0]):  
 for j in range(img2.shape[1]):  
 for l in range(3):  
 # v1 and v2 are 8-bit pixel values  
 # of img1 and img2 respectively  
 v1 = format(img1[i][j][l], '08b')  
 v2 = format(img2[i][j][l], '08b')  
  
 # Taking 4 MSBs of each image  
 v3 = v1[:4] + v2[:4]  
  
 img1[i][j][l] = int(v3, 2)  
  
 cv2.imwrite('new79.png', img1)  
  
  
# Decryption function  
def decrypt():  
 # Encrypted image  
 img = cv2.imread('new79.png')  
 width = img.shape[0]  
 height = img.shape[1]  
  
 # img1 and img2 are two blank images  
 img1 = np.zeros((width, height, 3), np.uint8)  
 img2 = np.zeros((width, height, 3), np.uint8)  
  
 for i in range(width):  
 for j in range(height):  
 for l in range(3):  
 v1 = format(img[i][j][l], '08b')  
 v2 = v1[:4] + chr(random.randint(0, 1) + 48) \* 4  
 v3 = v1[4:] + chr(random.randint(0, 1) + 48) \* 4  
  
 # Appending data to img1 and img2  
 img1[i][j][l] = int(v2, 2)  
 img2[i][j][l] = int(v3, 2)  
  
 # These are two images produced from  
 # the encrypted image  
 #cv2.imwrite('pic2\_re.png', img1)  
 cv2.imwrite('pic3\_re.png', img2)  
  
  
# Driver's code  
encrypt()  
decrypt()