Image Steganography

Group - 16

21BCY10210	-	SWATI
21BCY10019	-	SIDDHARTH DAYAL
21BCY10109	-	RUCHI BHATTACHARJEE
21BCY10123	-	HARSHITAA ASHISH

Team members

1.	SWATI	21BCY10210
2.	SIDDHARTH DAYAL	21BCY10019
3.	HARSHITAA ASHISH	21BCY10123
4.	RUCHI BHATTACHARJEE	21BCY10109

Guided by: Dr. Soma Saha

What is cryptography?

Cryptography is a technique for ensuring the privacy and security of files and communication by converting messages into an unreadable form for exchange between parties over an insecure channel.

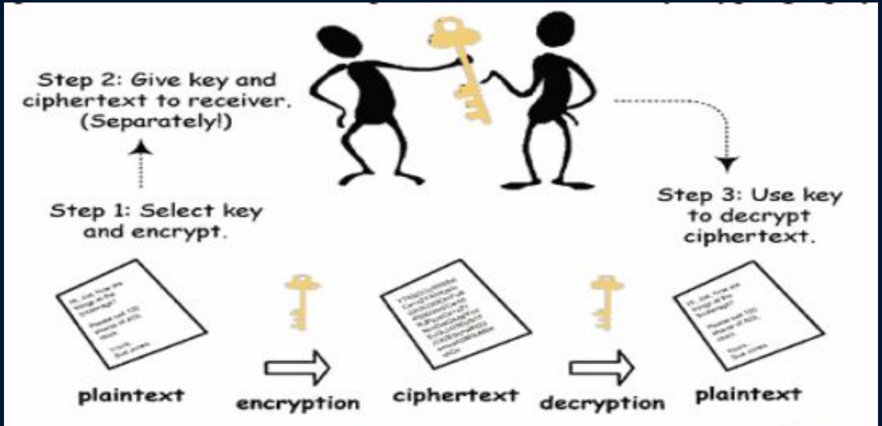
There are 2 types of cryptography :

- Symmetric (Secret Key Cryptography)
- Asymmetric (Public Key Cryptography)

Symmetric Key Cryptography

Secret key encryption, also known as symmetric-key encryption, uses a single key for both encryption and decryption. The sender encrypts the plain text message with the key and the receiver uses the same key to decrypt the encrypted message back to plain text.

Only authorized parties with knowledge of the key can perform encryption/decryption.



Step 2: Give key and ciphertext to receiver. (Separately!)
Step 1: Select key and encrypt.
Step 3: Use key to decrypt ciphertext.
plaintext
encryption
ciphertext
decryption
plaintext

Figure 2 Symmetric Key Cryptography Processes [3]

Asymmetric Key Cryptography

Asymmetric cryptography, also known as public key cryptography, uses two mathematically related keys, one for encryption and one for decryption. The encryption key is public and the decryption key is kept secret and is known as the private key. These keys cannot be derived from each other and both are required for the process to work.

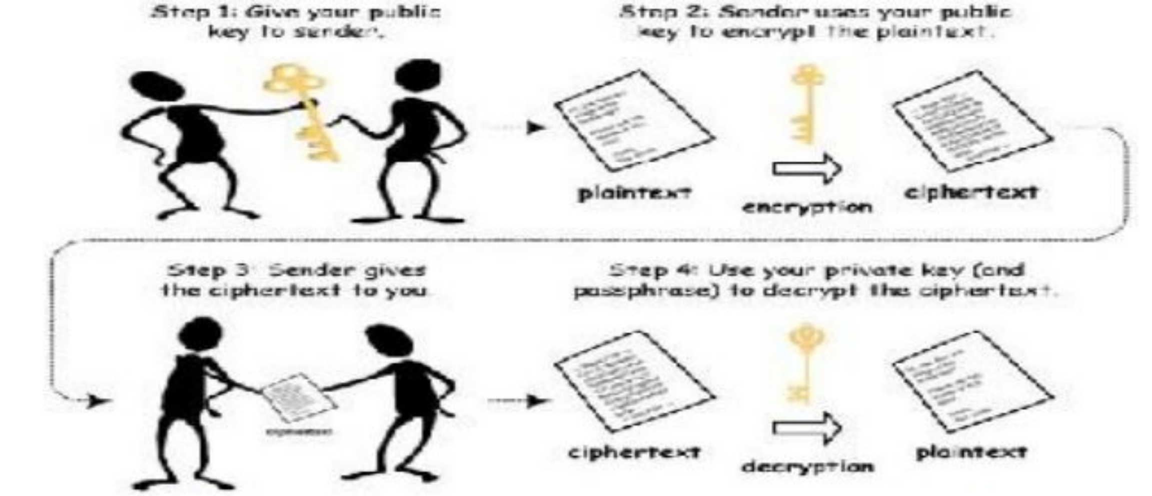
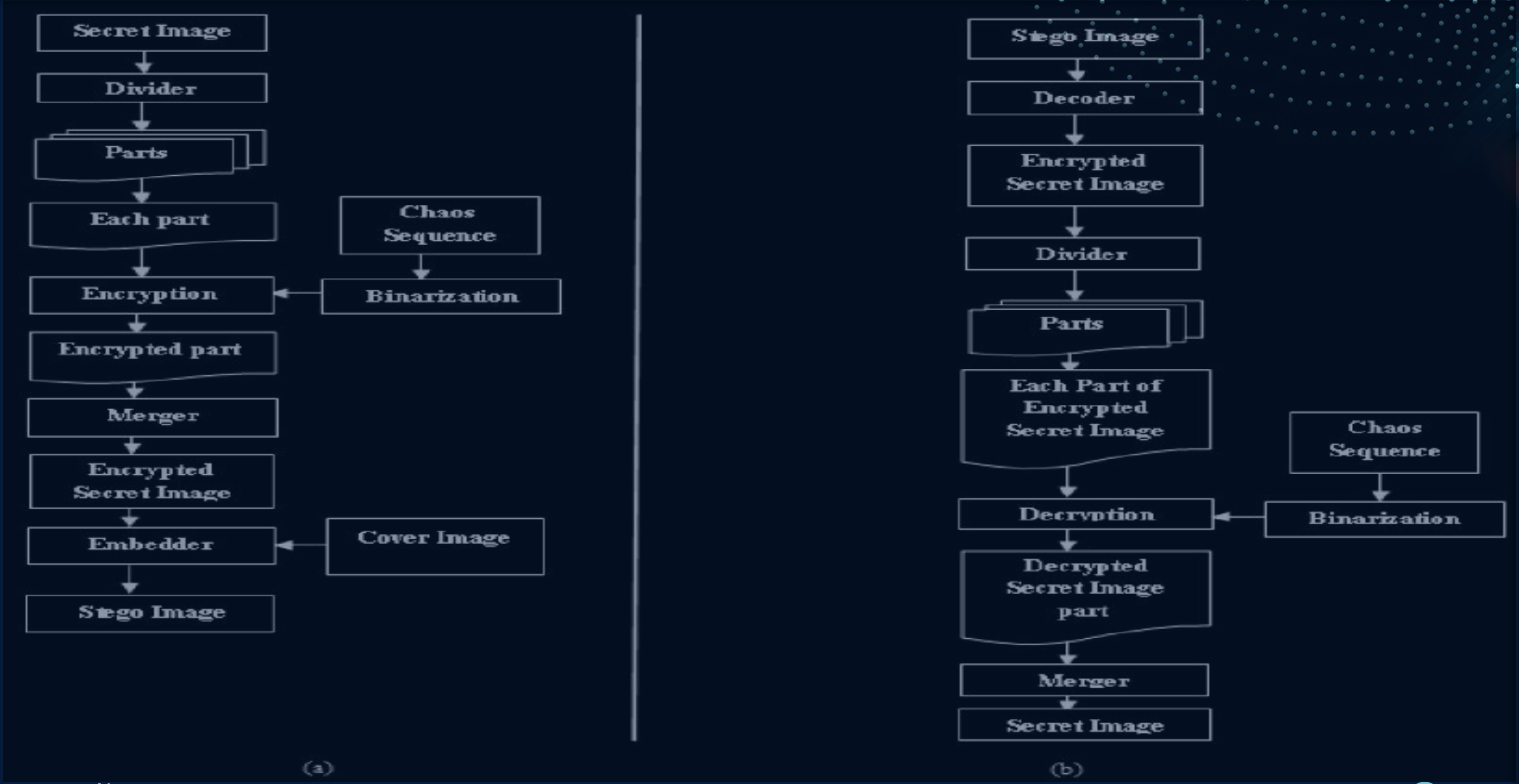

Step 1: Give your public key to sender.
Step 2: Sender uses your public key to encrypt the plaintext.
plaintext
encryption
ciphertext
Step 3: Sender gives the ciphertext to you.
Step 4: Use your private key (and passphrase) to decrypt the ciphertext.
ciphertext
decryption
plaintext

Figure 3 Symmetric Key Cryptography Processes [4]

FLOW DIAGRAM



● Encoding

● Decoding

Objective

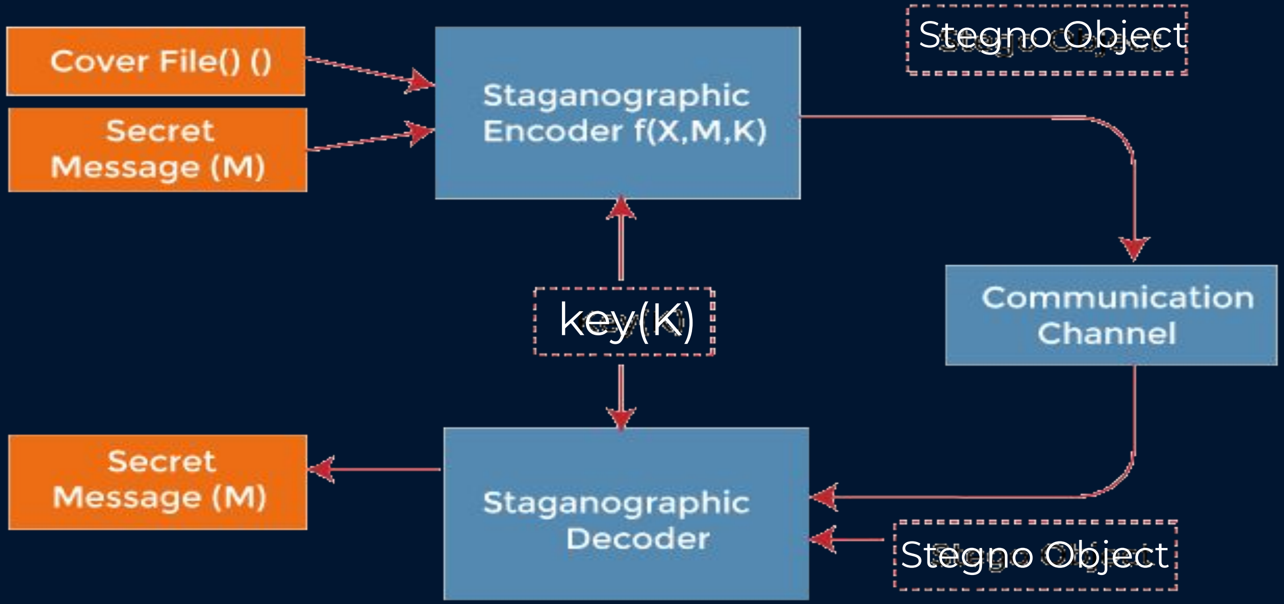
The objective of the code is to provide a GUI interface for the user to perform steganography on image files. Steganography is the practice of hiding a message or other information within another file in a way that is not immediately apparent.

The code provides two main functions: encoding text into an image and decoding text from an image.

When the user selects the "Encode" button, they are prompted to select an image file, enter the text they want to encode, and then select a location to save the encoded image.

When the user selects the "Decode" button, they are prompted to select an image file, and the program will attempt to decode any text that may have been previously encoded in the image.

SYSTEM ARCHITECTURE DIAGRAM



Organisation of Modules

Tkinter Module

Standard interface in python for creating a GUI (Graphical User Interface).

Tkinter import*

To import everything from the module

tkinter.filedialog

To work with files

From tkinter import messagebox

Import message box separately for showing messages on the screen

Organisation of Modules

PIL Module

Open, save and manipulate images

Import ImageTk

Create and modify Tkinter photoimage from PIL images

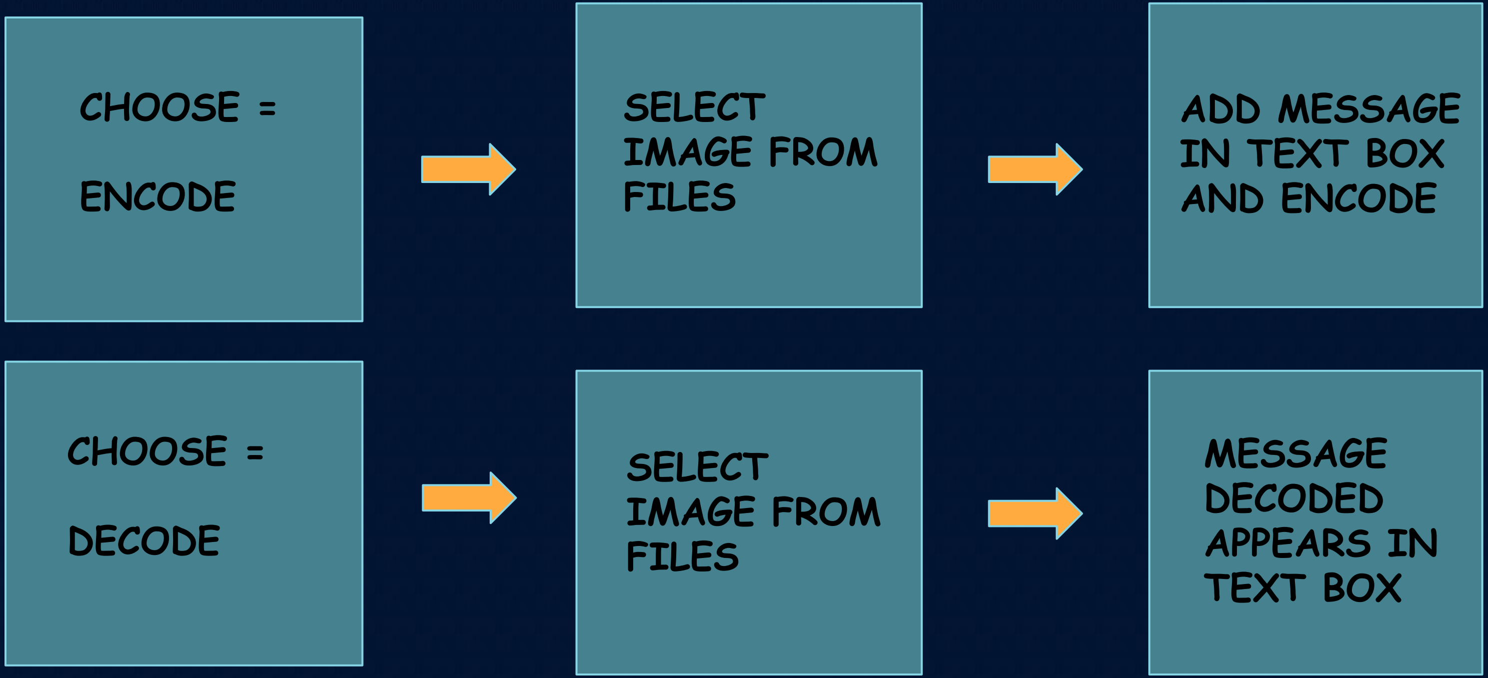
io import Bytesio

Bytes data in the memory

Import os

For creation and removal of directory

Output Diagram



CHOOSE =

ENCODE

SELECT IMAGE FROM FILES

ADD MESSAGE IN TEXT BOX AND ENCODE

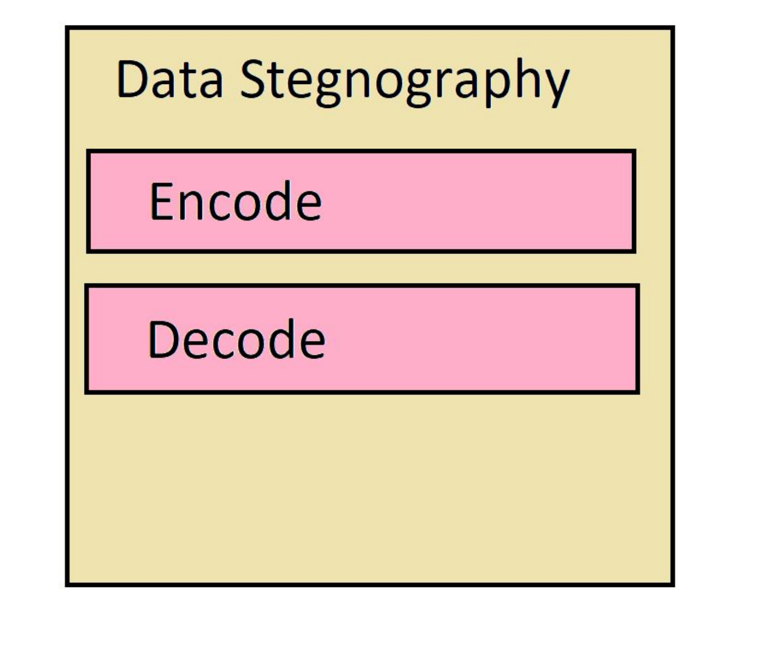
CHOOSE =

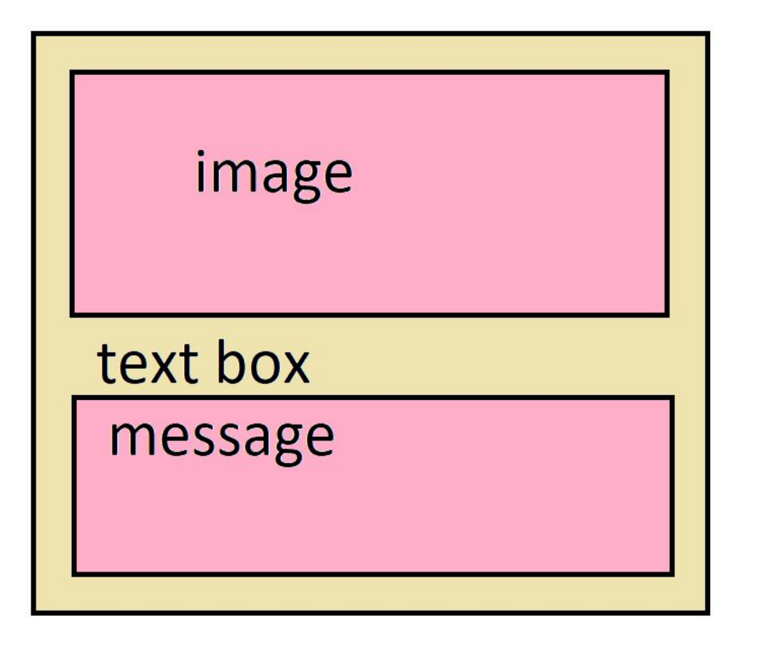
DECODE

SELECT IMAGE FROM FILES

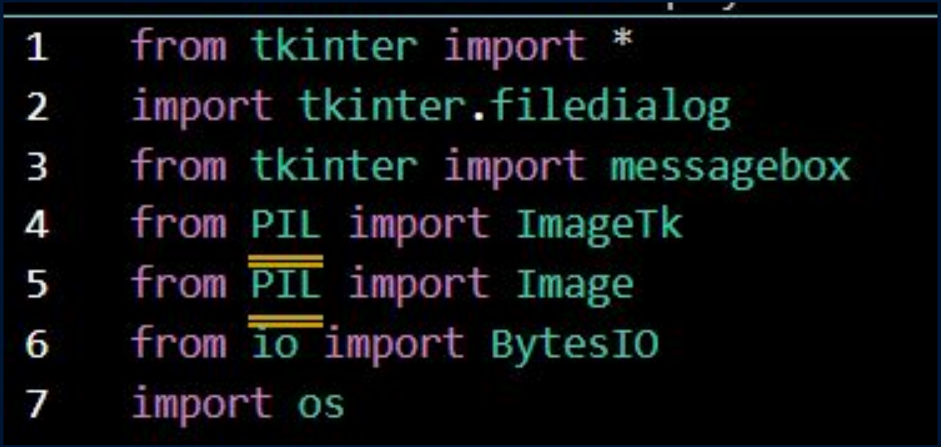
MESSAGE DECODED APPEARS IN TEXT BOX

Sample Output

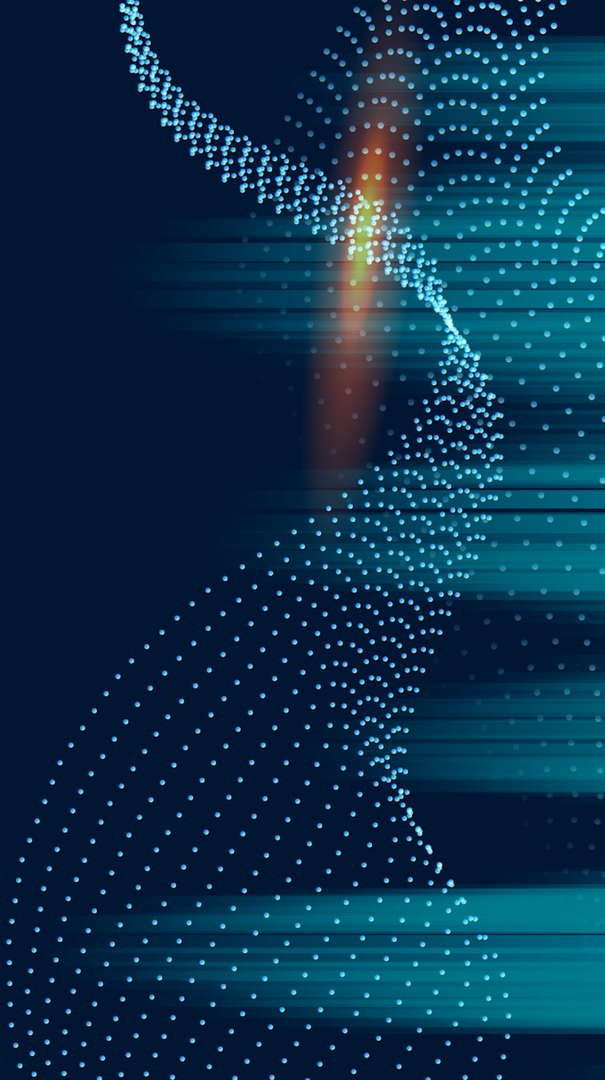




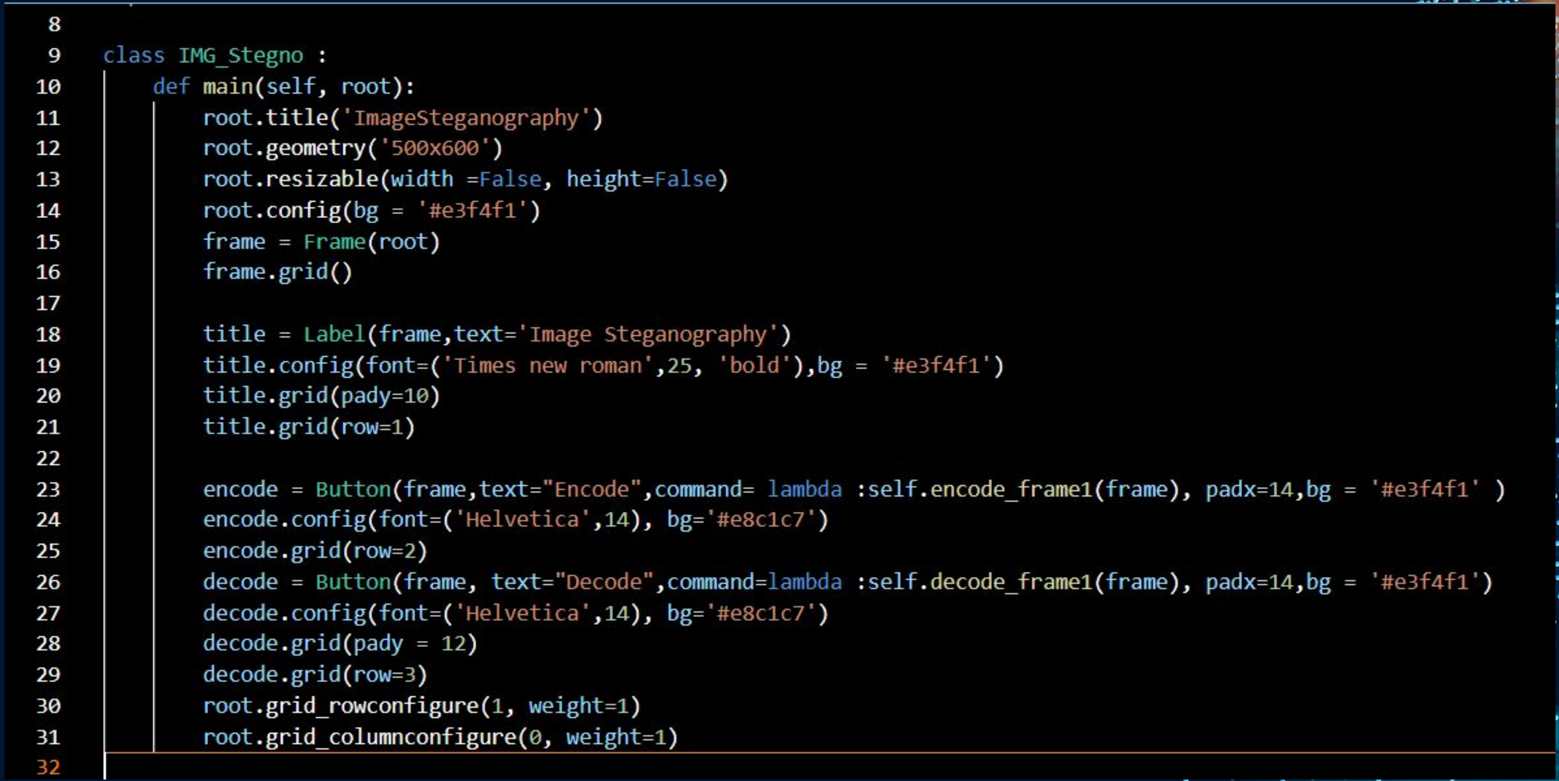
LIBRARIES CODE



```
from tkinter import *
import tkinter.filedialog
from tkinter import messagebox
from PIL import ImageTk
from PIL import Image
from io import BytesIO
import os
```

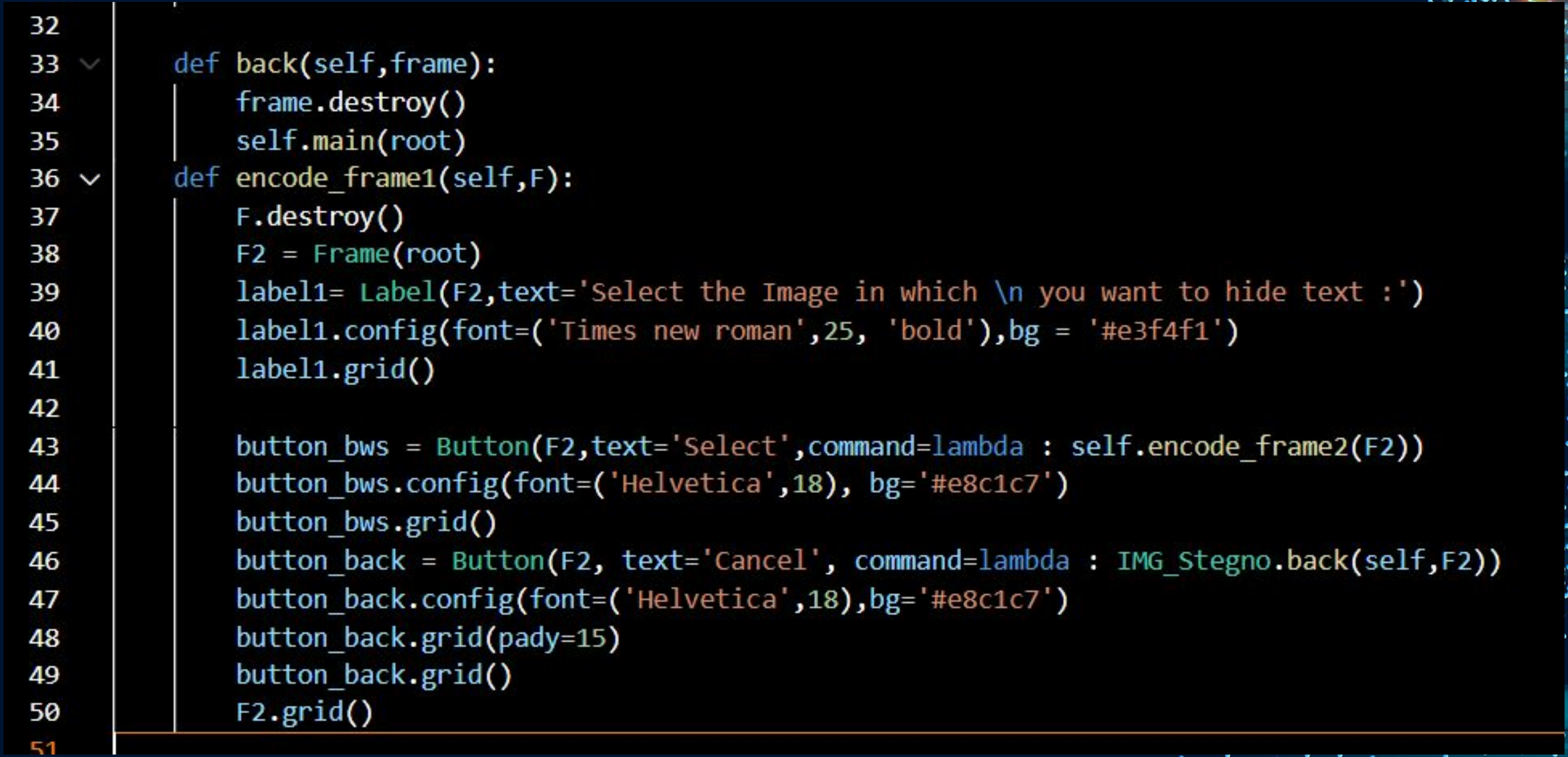


CODE



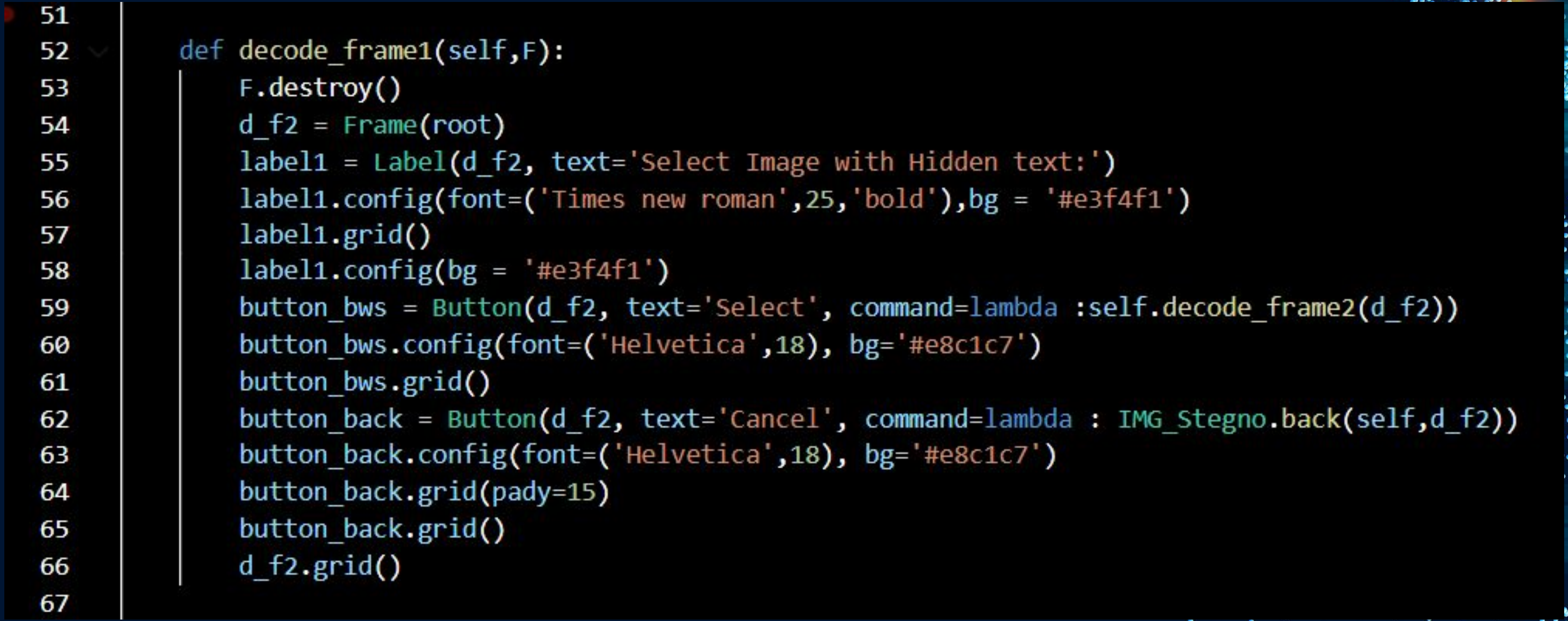
```
class IMG_Stegno :
    def main(self, root):
        root.title('ImageSteganography')
        root.geometry('500x600')
        root.resizable(width =False, height=False)
        root.config(bg = '#e3f4f1')
        frame = Frame(root)
        frame.grid()

        title = Label(frame,text='Image Steganography')
        title.config(font=('Times new roman',25, 'bold'),bg = '#e3f4f1')
        title.grid(pady=10)
        title.grid(row=1)

        encode = Button(frame,text="Encode",command= lambda :self.encode_frame1(frame), padx=14,bg = '#e3f4f1' )
        encode.config(font=('Helvetica',14), bg='#e8c1c7')
        encode.grid(row=2)
        decode = Button(frame, text="Decode",command=lambda :self.decode_frame1(frame), padx=14,bg = '#e3f4f1')
        decode.config(font=('Helvetica',14), bg='#e8c1c7')
        decode.grid(pady = 12)
        decode.grid(row=3)
        root.grid_rowconfigure(1, weight=1)
        root.grid_columnconfigure(0, weight=1)
```

CODE



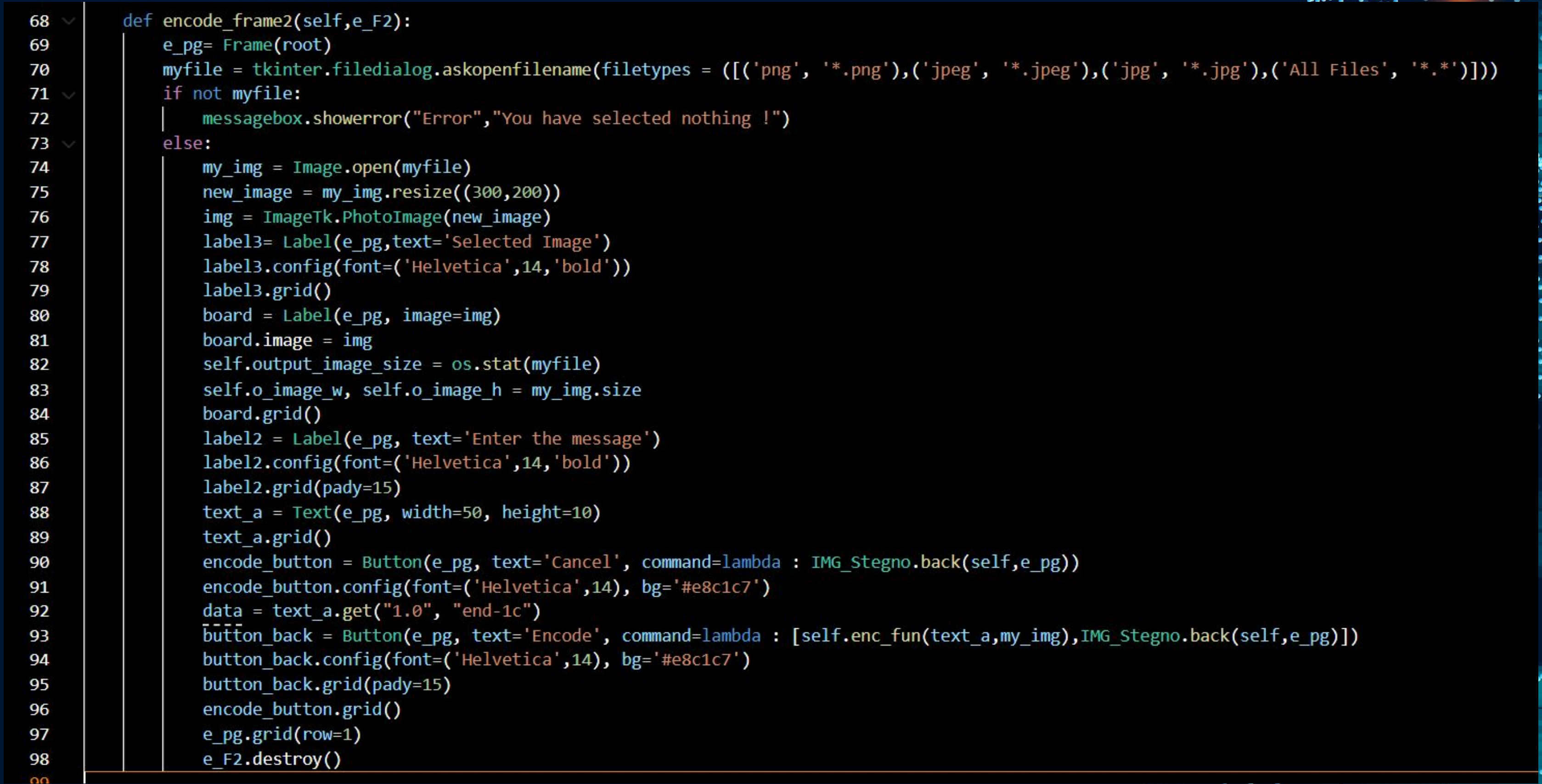
```python

    def back(self,frame):
        frame.destroy()
        self.main(root)
    def encode_frame1(self,F):
        F.destroy()
        F2 = Frame(root)
        label1= Label(F2,text='Select the Image in which \n you want to hide text :')
        label1.config(font=('Times new roman',25, 'bold'),bg = '#e3f4f1')
        label1.grid()

        button_bws = Button(F2,text='Select',command=lambda : self.encode_frame2(F2))
        button_bws.config(font=('Helvetica',18), bg='#e8c1c7')
        button_bws.grid()
        button_back = Button(F2, text='Cancel', command=lambda : IMG_Stegno.back(self,F2))
        button_back.config(font=('Helvetica',18),bg='#e8c1c7')
        button_back.grid(pady=15)
        button_back.grid()
        F2.grid()
```

CODE



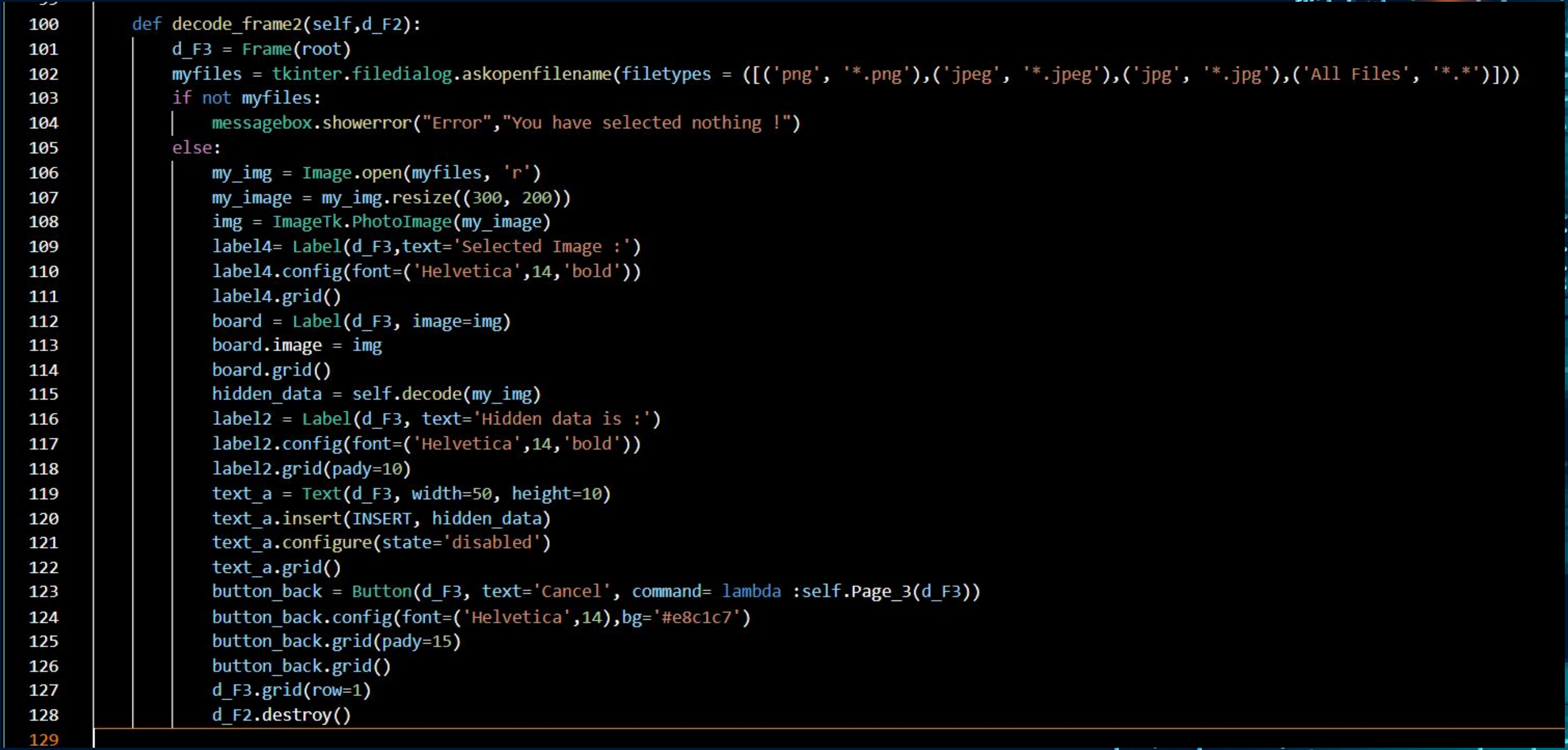
```
    def decode_frame1(self,F):
        F.destroy()
        d_f2 = Frame(root)
        label1 = Label(d_f2, text='Select Image with Hidden text:')
        label1.config(font=('Times new roman',25,'bold'),bg = '#e3f4f1')
        label1.grid()
        label1.config(bg = '#e3f4f1')
        button_bws = Button(d_f2, text='Select', command=lambda :self.decode_frame2(d_f2))
        button_bws.config(font=('Helvetica',18), bg='#e8c1c7')
        button_bws.grid()
        button_back = Button(d_f2, text='Cancel', command=lambda : IMG_Stegno.back(self,d_f2))
        button_back.config(font=('Helvetica',18), bg='#e8c1c7')
        button_back.grid(pady=15)
        button_back.grid()
        d_f2.grid()

```

CODE



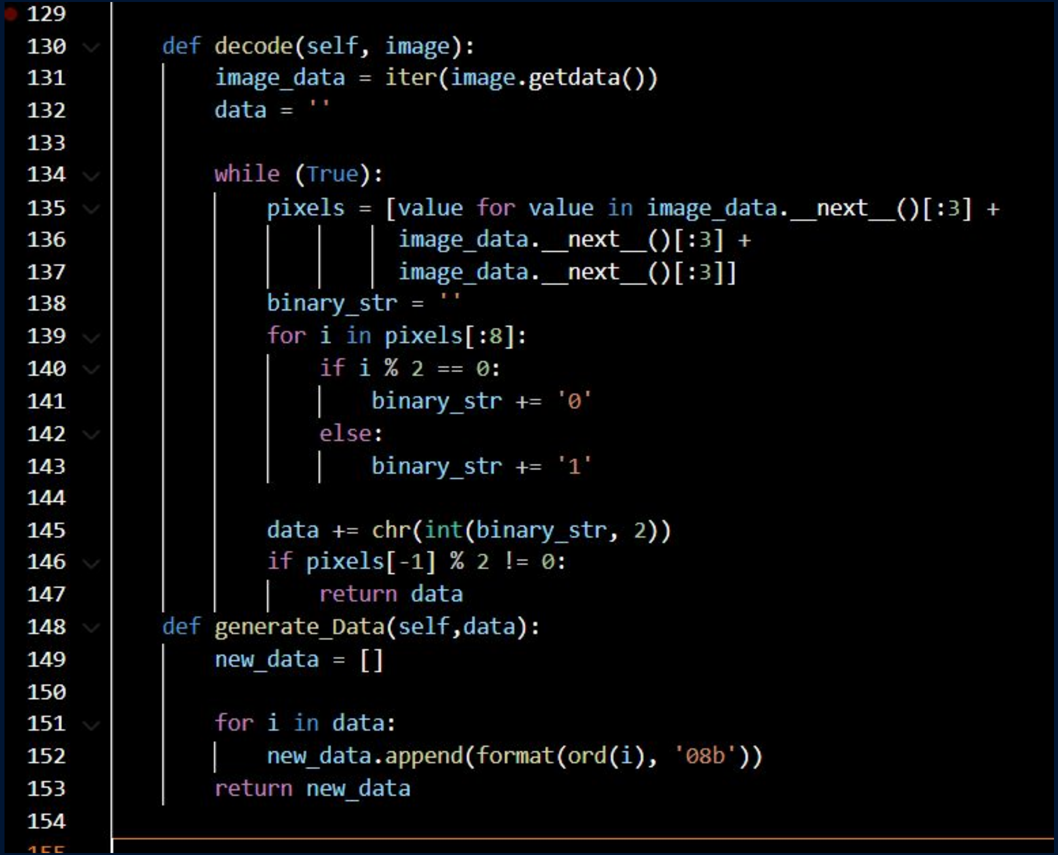
```python
    def encode_frame2(self,e_F2):
        e_pg= Frame(root)
        myfile = tkinter.filedialog.askopenfilename(filetypes = ([('png', '*.png'),('jpeg', '*.jpeg'),('jpg', '*.jpg'),('All Files', '*.*')]))
        if not myfile:
            messagebox.showerror("Error","You have selected nothing !")
        else:
            my_img = Image.open(myfile)
            new_image = my_img.resize((300,200))
            img = ImageTk.PhotoImage(new_image)
            label3= Label(e_pg,text='Selected Image')
            label3.config(font=('Helvetica',14,'bold'))
            label3.grid()
            board = Label(e_pg, image=img)
            board.image = img
            self.output_image_size = os.stat(myfile)
            self.o_image_w, self.o_image_h = my_img.size
            board.grid()
            label2 = Label(e_pg, text='Enter the message')
            label2.config(font=('Helvetica',14,'bold'))
            label2.grid(pady=15)
            text_a = Text(e_pg, width=50, height=10)
            text_a.grid()
            encode_button = Button(e_pg, text='Cancel', command=lambda : IMG_Stegno.back(self,e_pg))
            encode_button.config(font=('Helvetica',14), bg='#e8c1c7')
            data = text_a.get("1.0", "end-1c")
            button_back = Button(e_pg, text='Encode', command=lambda : [self.enc_fun(text_a,my_img),IMG_Stegno.back(self,e_pg)])
            button_back.config(font=('Helvetica',14), bg='#e8c1c7')
            button_back.grid(pady=15)
            encode_button.grid()
            e_pg.grid(row=1)
            e_F2.destroy()
```

CODE

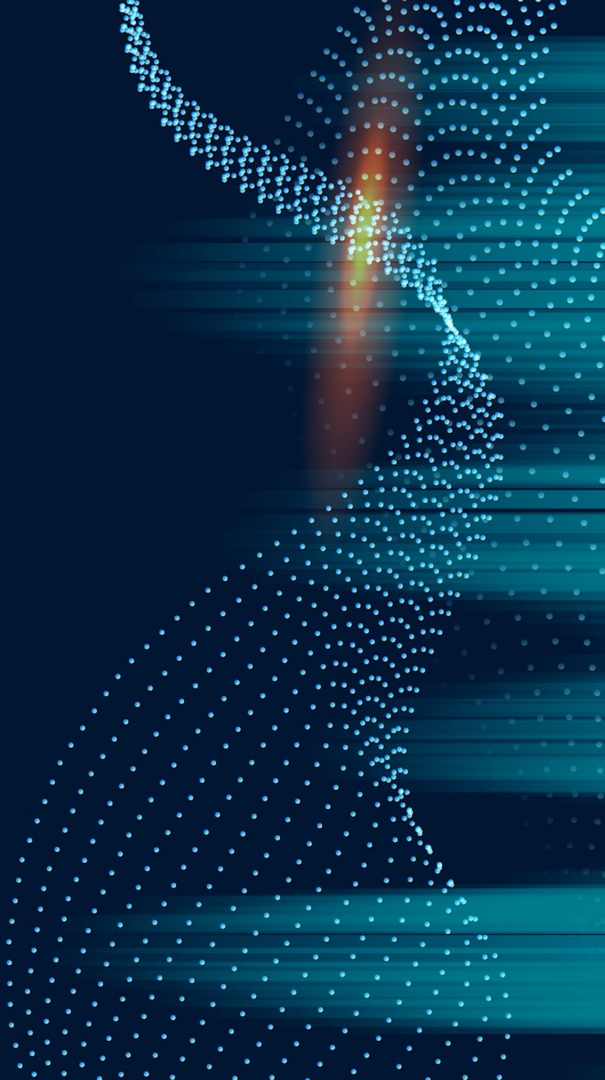


```
    def decode_frame2(self,d_F2):
        d_F3 = Frame(root)
        myfiles = tkinter.filedialog.askopenfilename(filetypes = ([('png', '*.png'),('jpeg', '*.jpeg'),('jpg', '*.jpg'),('All Files', '*.*')]))
        if not myfiles:
            messagebox.showerror("Error","You have selected nothing !")
        else:
            my_img = Image.open(myfiles, 'r')
            my_image = my_img.resize((300, 200))
            img = ImageTk.PhotoImage(my_image)
            label4= Label(d_F3,text='Selected Image :')
            label4.config(font=('Helvetica',14,'bold'))
            label4.grid()
            board = Label(d_F3, image=img)
            board.image = img
            board.grid()
            hidden_data = self.decode(my_img)
            label2 = Label(d_F3, text='Hidden data is :')
            label2.config(font=('Helvetica',14,'bold'))
            label2.grid(pady=10)
            text_a = Text(d_F3, width=50, height=10)
            text_a.insert(INSERT, hidden_data)
            text_a.configure(state='disabled')
            text_a.grid()
            button_back = Button(d_F3, text='Cancel', command= lambda :self.Page_3(d_F3))
            button_back.config(font=('Helvetica',14),bg='#e8c1c7')
            button_back.grid(pady=15)
            button_back.grid()
            d_F3.grid(row=1)
            d_F2.destroy()

```

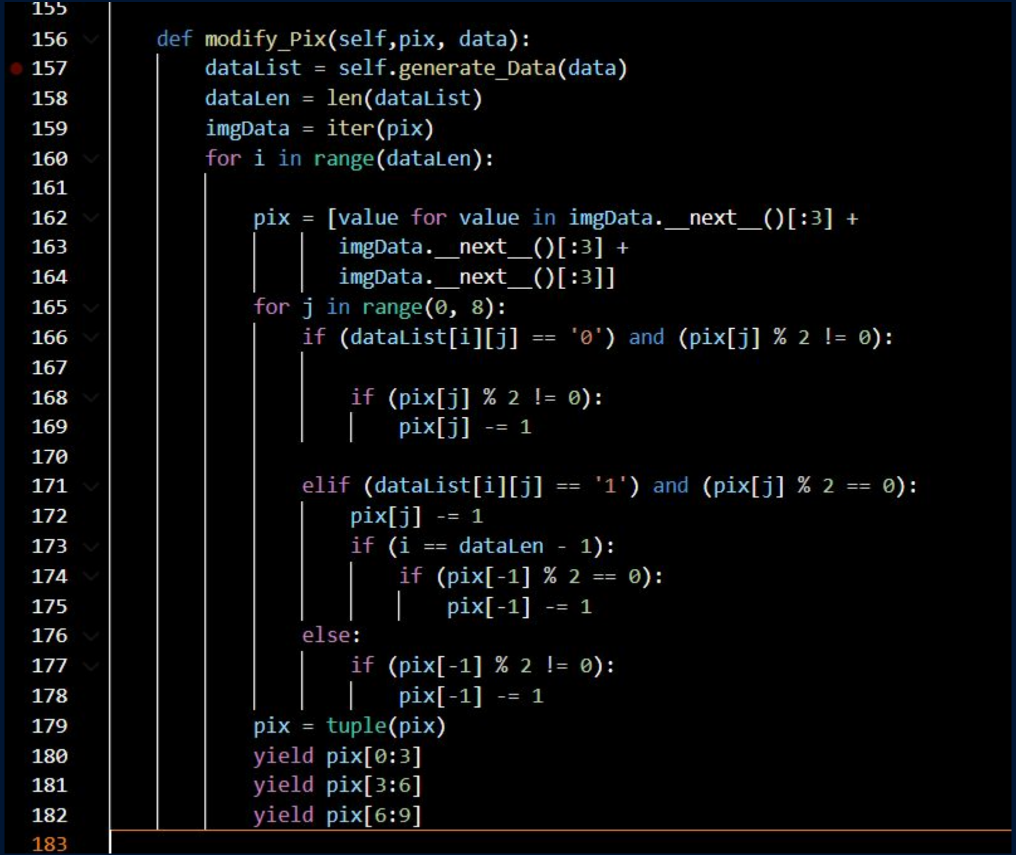
CODE



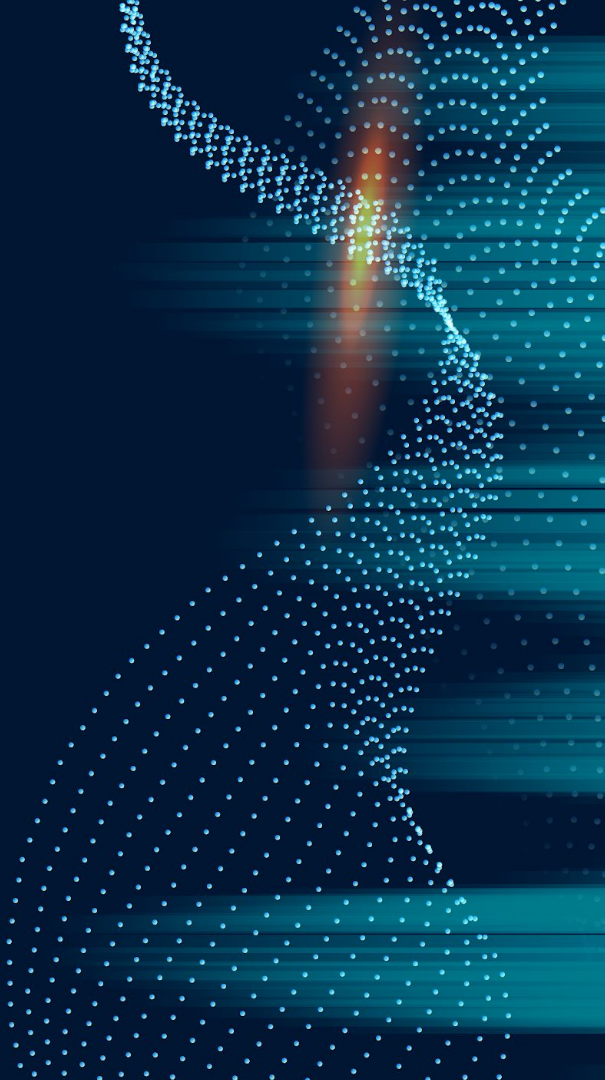
```python
    def decode(self, image):
        image_data = iter(image.getdata())
        data = ''

        while (True):
            pixels = [value for value in image_data.__next__()[:3] +
                      image_data.__next__()[:3] +
                      image_data.__next__()[:3]]
            binary_str = ''
            for i in pixels[:8]:
                if i % 2 == 0:
                    binary_str += '0'
                else:
                    binary_str += '1'

            data += chr(int(binary_str, 2))
            if pixels[-1] % 2 != 0:
                return data
    def generate_Data(self,data):
        new_data = []

        for i in data:
            new_data.append(format(ord(i), '08b'))
        return new_data
```



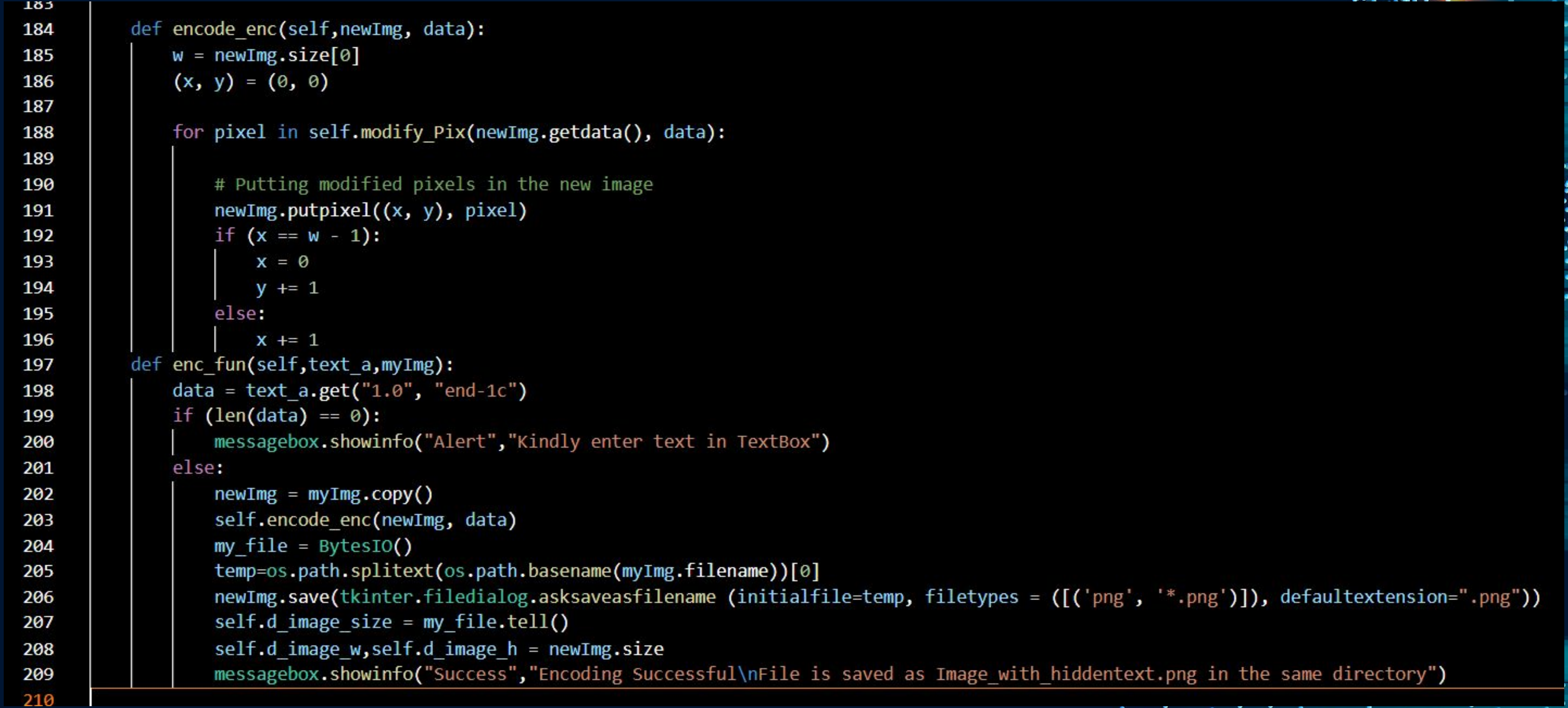
CODE



```
    def modify_Pix(self,pix, data):
        dataList = self.generate_Data(data)
        dataLen = len(dataList)
        imgData = iter(pix)
        for i in range(dataLen):

            pix = [value for value in imgData.__next__()[:3] +
                   imgData.__next__()[:3] +
                   imgData.__next__()[:3]]
            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
                    if (i == dataLen - 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]

```

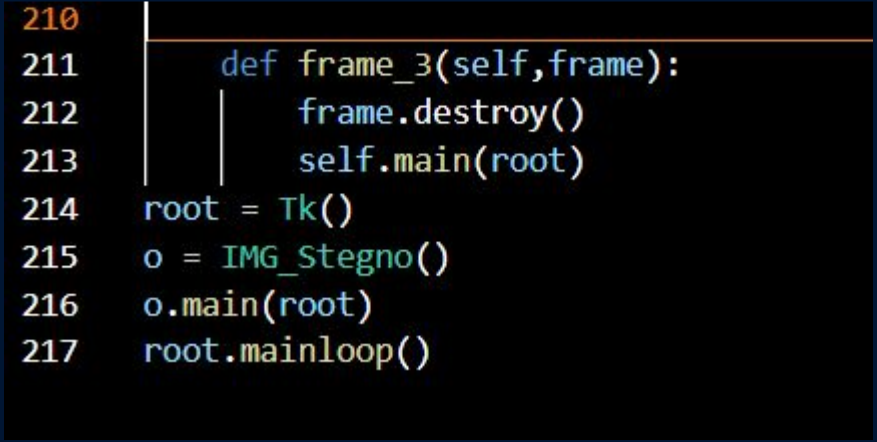


CODE

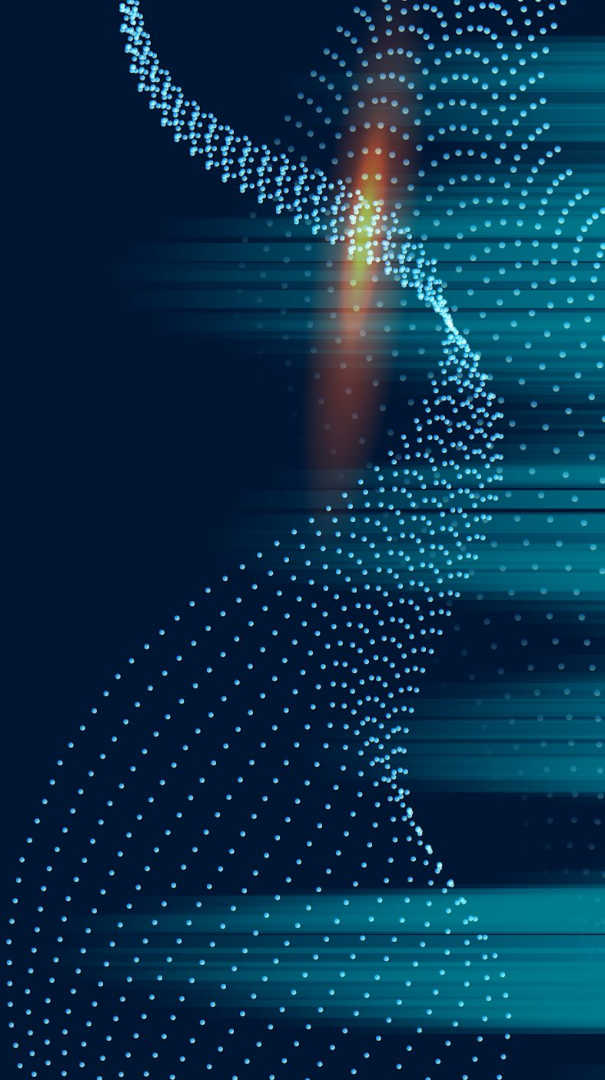


```
    def encode_enc(self,newImg, data):
        w = newImg.size[0]
        (x, y) = (0, 0)

        for pixel in self.modify_Pix(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
    def enc_fun(self,text_a,myImg):
        data = text_a.get("1.0", "end-1c")
        if (len(data) == 0):
            messagebox.showinfo("Alert","Kindly enter text in TextBox")
        else:
            newImg = myImg.copy()
            self.encode_enc(newImg, data)
            my_file = BytesIO()
            temp=os.path.splitext(os.path.basename(myImg.filename))[0]
            newImg.save(tkinter.filedialog.asksaveasfilename (initialfile=temp, filetypes = ([('png', '*.png')]), defaultextension=".png"))
            self.d_image_size = my_file.tell()
            self.d_image_w,self.d_image_h = newImg.size
            messagebox.showinfo("Success","Encoding Successful\nFile is saved as Image_with_hiddentext.png in the same directory")
```

CODE



```

    def frame_3(self,frame):
        frame.destroy()
        self.main(root)
root = Tk()
o = IMG_Stegno()
o.main(root)
root.mainloop()
```



OUTPUT IMAGES (encode)

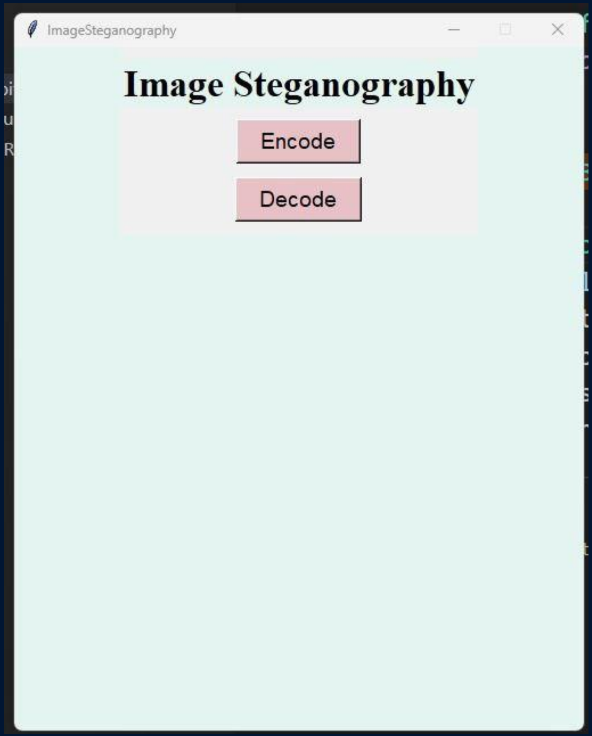
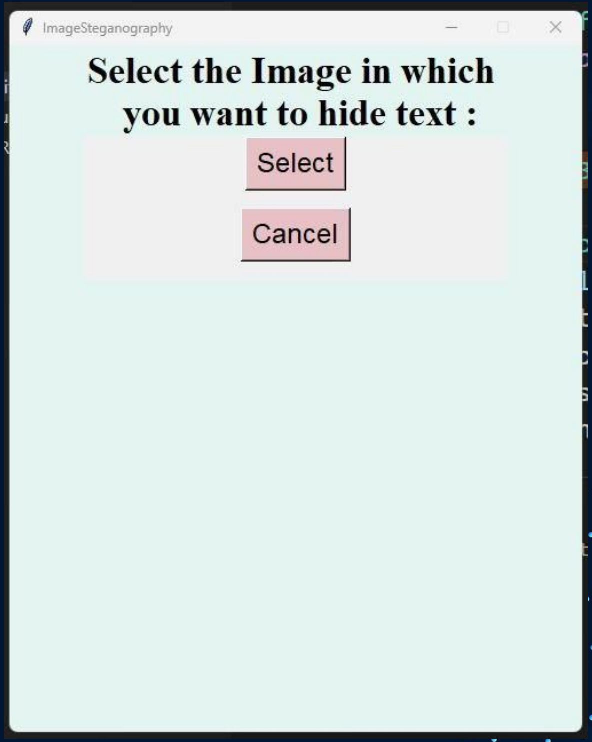
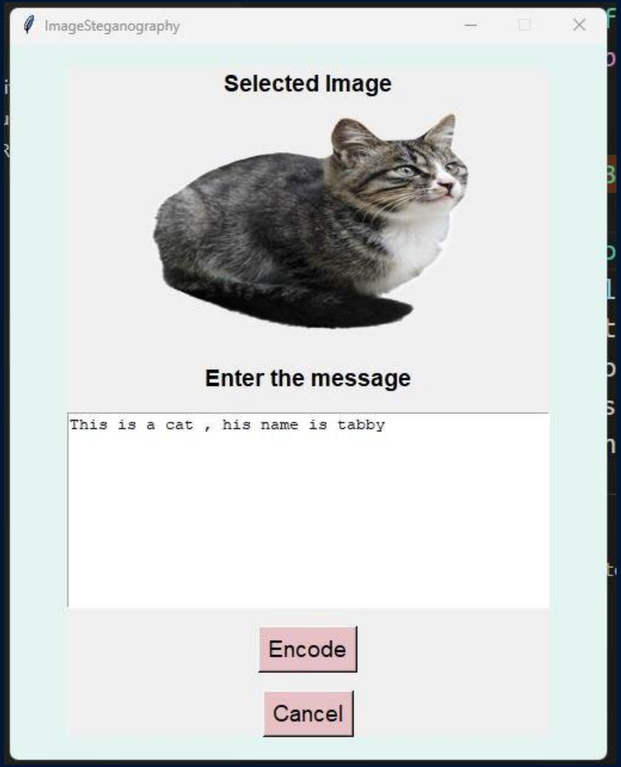


Image Steganography

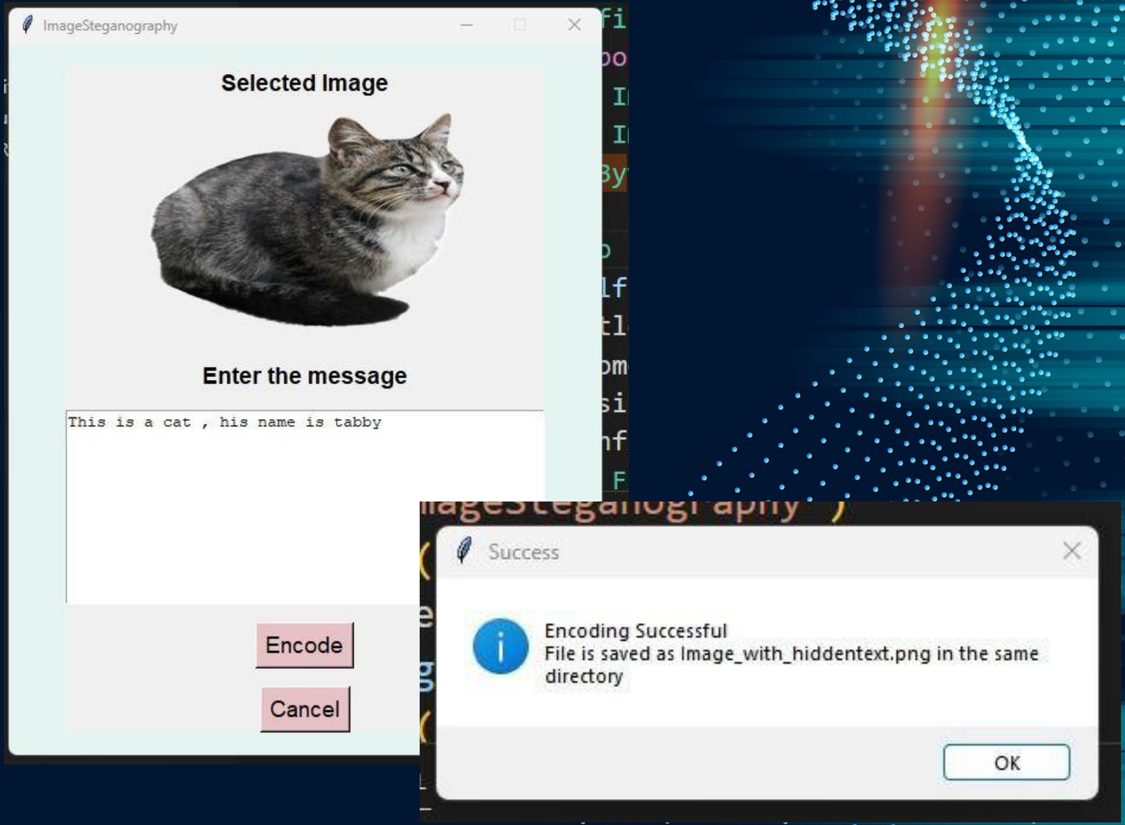


Select the Image in which you want to hide text :

OUTPUT IMAGES (encode)



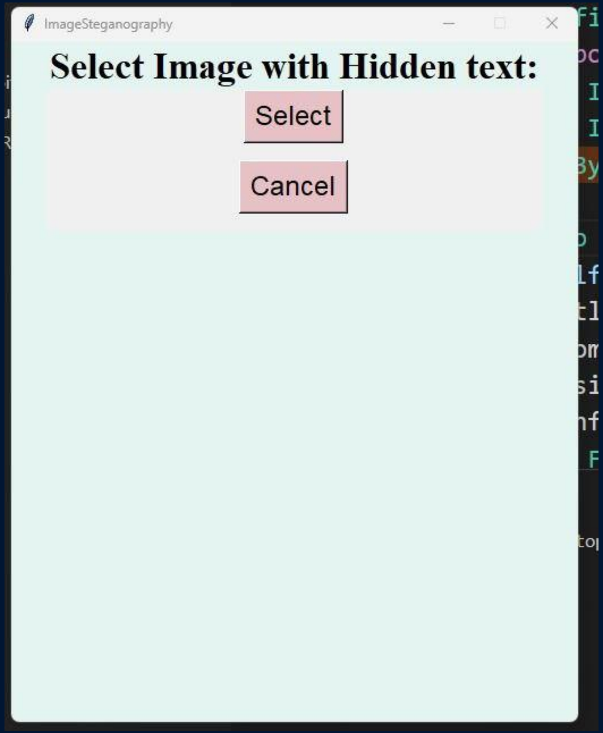
This is a cat , his name is tabby



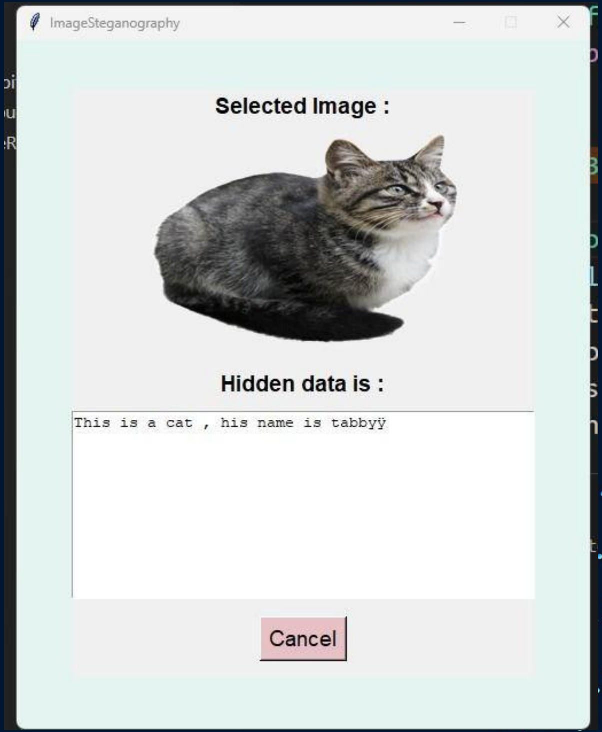
This is a cat , his name is tabby

Encoding Successful
File is saved as Image_with_hiddentext.png in the same directory

OUTPUT IMAGES (decode)



Select Image with Hidden text:



Selected Image :

Hidden data is :

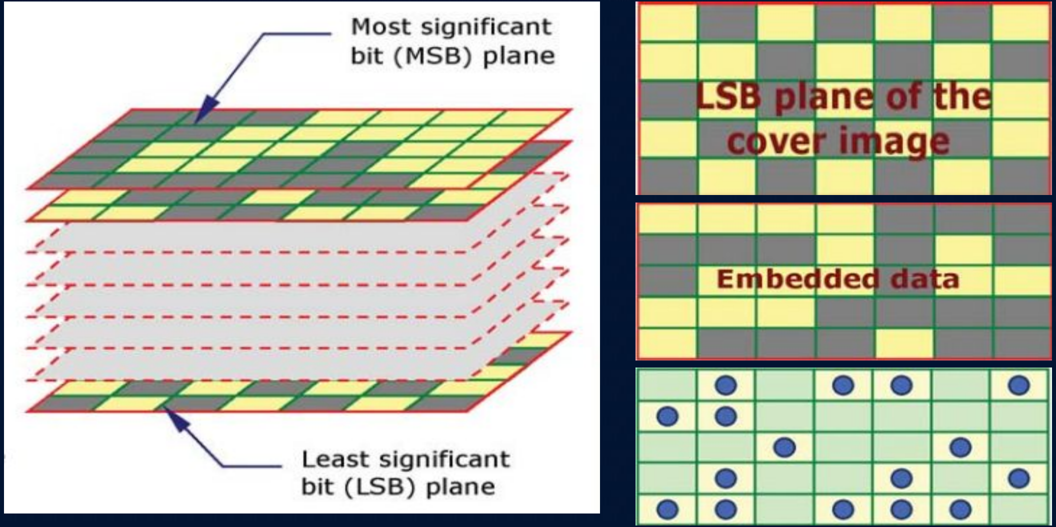
```
This is a cat , his name is tabbyÿ
```

Steganographic Techniques

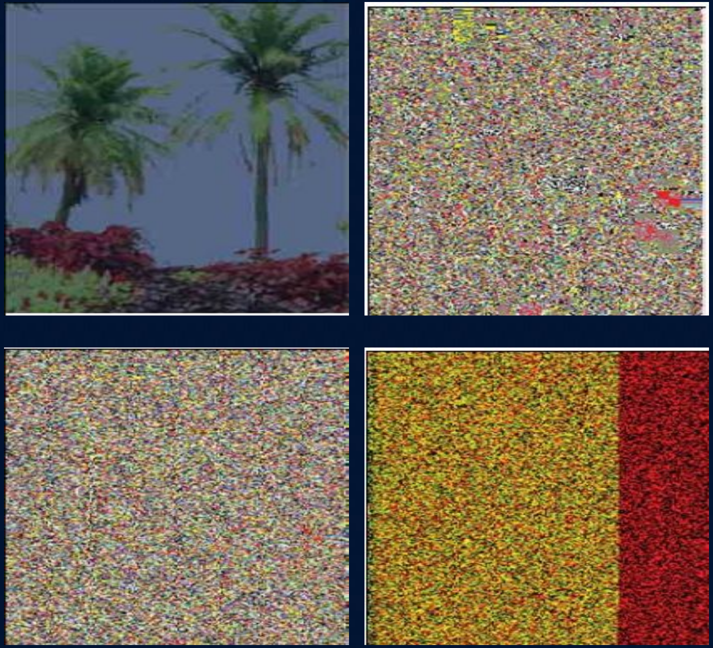
- **Quantization Index Modulation (QIM)**- which uses different quantizers to carry different bits of the secret data. Although a simple unified method for classifying these techniques does not exist, some popular approaches are used in downloadable steganographic tools or found in the literature.

- **LSB modification -** These techniques are based on modifying the least significant bits (LSBs), of the pixel values in the space domain. In a basic implementation, these pixels replace the entire LSB-plane with the stego-data; on average, 50% of the LSBs are flipped It can be shown that fidelity of the stego-image measured in peak-signal-to-noise ratio with respect to the cover is 51.1dB, representing a very high degree of imperceptibility compared to the lower bound of 39dB generally accepted by researchers of watermarking. With more sophisticated schemes in which embedding locations are adaptively selected, depending on human vision characteristics, even less distortion is achievable.

LSB Embedding


Most significant bit (MSB) plane
Least significant bit (LSB) plane
LSB plane of the cover image
Embedded data

Basic LSB Approach



Effects of LSB Embedding

Steganographic Techniques

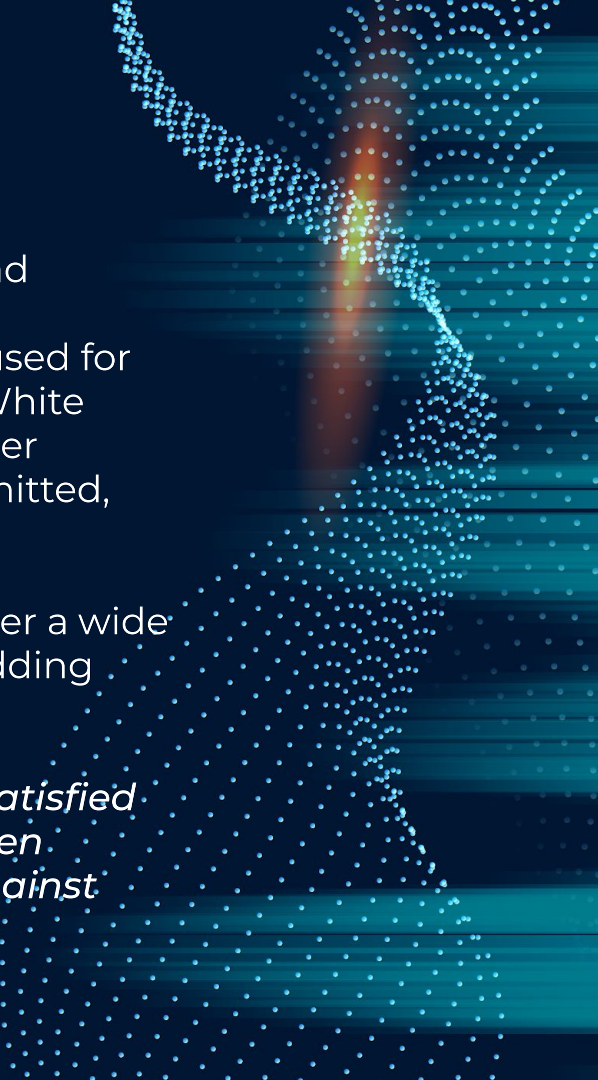
- **Masking approaches** -These techniques are similar to visible watermarking in which pixel values in masked areas are raised or lowered by some percentage. Reducing the increment to a certain degree makes the mark invisible. In the patchwork method, pairs of patches are selected pseudo-randomly; pixel values in each pair are raised by a small constant value in one patch and lowered by the same amount in the other.

- **Transform domain techniques -**Data embedding performed in the transform domain is widely used for robust watermarking. Similar techniques can also realize large-capacity embedding for steganography. Candidate transforms include discrete cosine transform (DCT), discrete wavelet transform (DWT), and discrete Fourier transform (DFT). By being embedded in the transform domain, the hidden data resides in more robust areas, spread across the entire image, and provides better resistance against signal processing. Various methods are available.

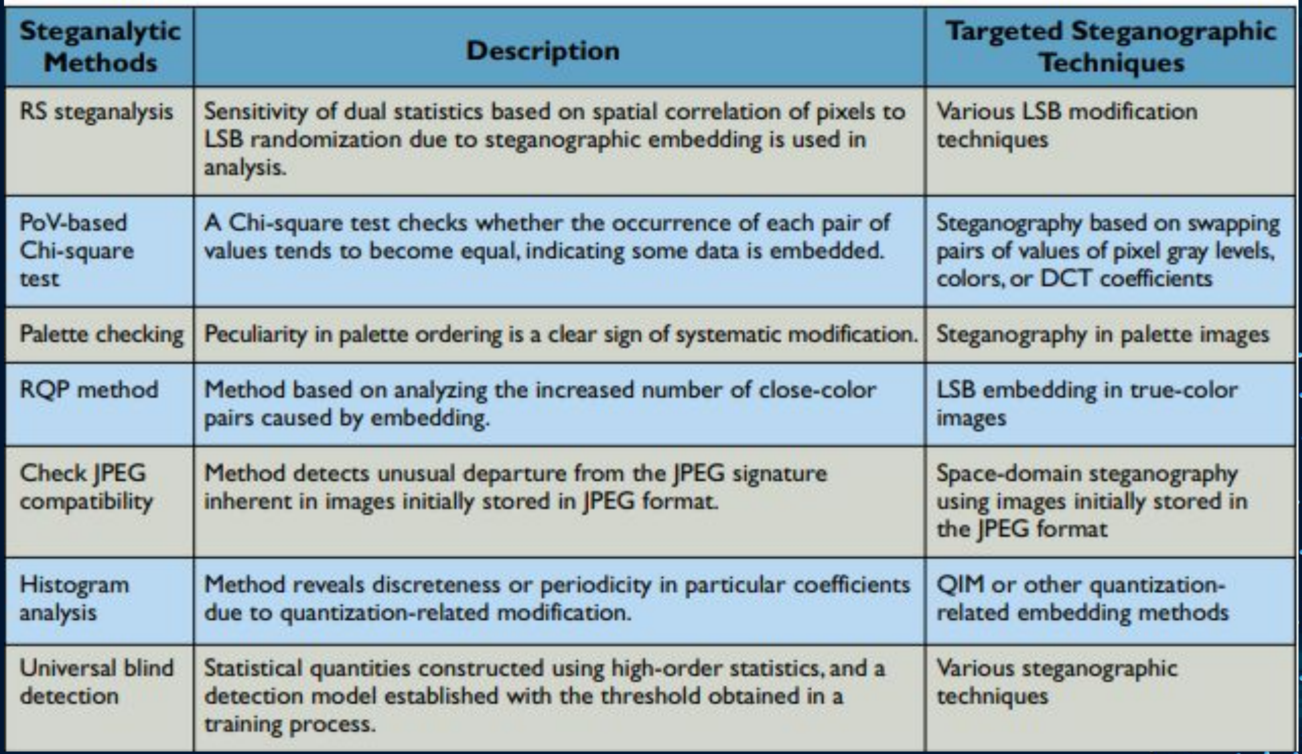
Steganographic Techniques

- **Spread-spectrum techniques -** The hidden data is spread throughout the cover-image based on spread-spectrum techniques (such as frequency hopping). A stego-key is used for encryption to randomly select the frequency channels. White Noise Storm is a popular tool using this technique. In other research, with embedded data as the object to be transmitted, the cover-image is viewed as interference in a covert communication framework. The embedded data is first modulated with pseudo-noise so the energy is spread over a wide frequency band, achieving only a very low level of embedding strength. This is valuable in achieving imperceptibility.

  The three most important requirements that must be satisfied for any steganographic system are: security of the hidden communication; size of the payload; and robustness against malicious and unintentional attacks.



Detection of Steganographic Contents



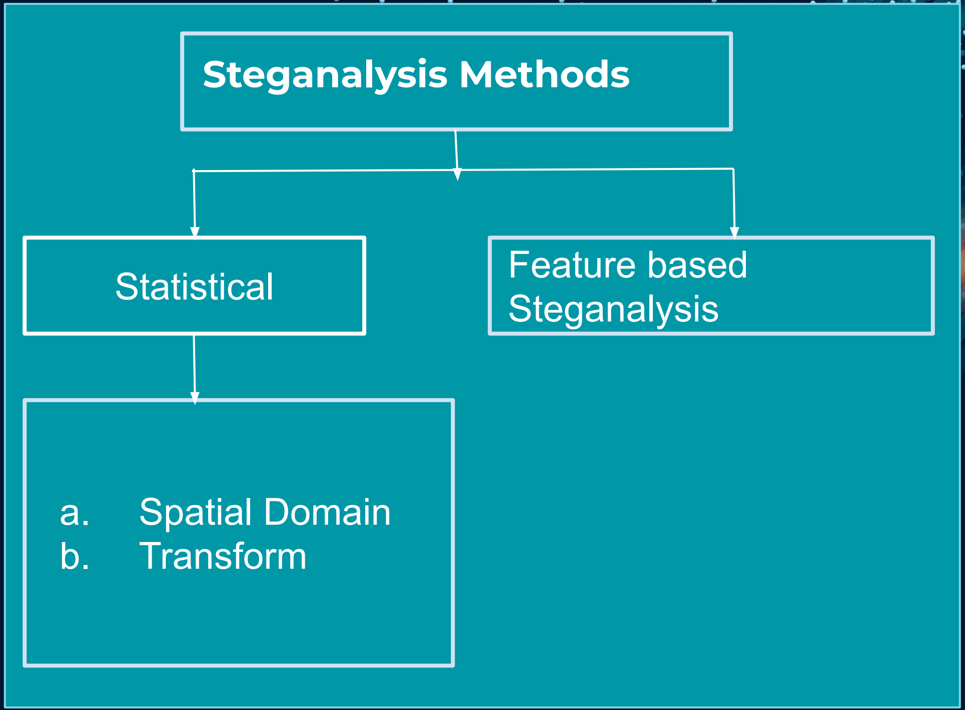
Steganalytic Methods	Description	Targeted Steganographic Techniques
RS steganalysis	Sensitivity of dual statistics based on spatial correlation of pixels to LSB randomization due to steganographic embedding is used in analysis.	Various LSB modification techniques
PoV-based Chi-square test	A Chi-square test checks whether the occurrence of each pair of values tends to become equal, indicating some data is embedded.	Steganography based on swapping pairs of values of pixel gray levels, colors, or DCT coefficients
Palette checking	Peculiarity in palette ordering is a clear sign of systematic modification.	Steganography in palette images
RQP method	Method based on analyzing the increased number of close-color pairs caused by embedding.	LSB embedding in true-color images
Check JPEG compatibility	Method detects unusual departure from the JPEG signature inherent in images initially stored in JPEG format.	Space-domain steganography using images initially stored in the JPEG format
Histogram analysis	Method reveals discreteness or periodicity in particular coefficients due to quantization-related modification.	QIM or other quantization-related embedding methods
Universal blind detection	Statistical quantities constructed using high-order statistics, and a detection model established with the threshold obtained in a training process.	Various steganographic techniques

Steganalysis

Steganography is a strong technique to hide a secure message, however, it also vulnerable when the technique used to spread malicious and harmful content by embedding. To identify such content, steganalysis has been performed. The counter technique of image steganography is known as image steganalysis, which begins by recognizing the object that exists in the embedded source file. The aim of this process is not to advocate the removal or disabling of valid hidden information such as copyrights, but to point out approaches that are vulnerable and may be exploited to investigate illegal and dishonest hidden information

Steganalysis Methods

Steganalysis is the method, which detects the presence of hidden data; this process can be categorized by different types such as Statistical steganalysis which contains spatial domain. Transform domain and Feature based steganalysis. The Statistical steganalysis helps to detect the existence of the hidden message, statistical analysis is done with the pixels and it is further classified as spatial domain steganalysis and transforms domain steganalysis. In spatial domain, the pair of pixels is considered and the difference between them is calculated. The pair may be any two neighboring pixels.



Steganalysis Methods

Statistical

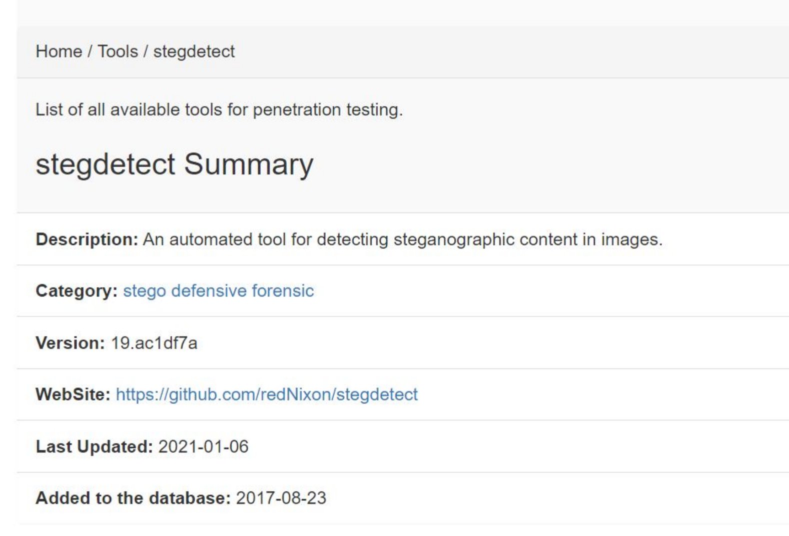
Feature based Steganalysis

a. Spatial Domain
b. Transform

Steganalysis tools

Steganalysis usually consist several processes like cropping, blurring, image resizing, noise removal and compression process. Various steganalysis tools are available to detect the presence of hidden information with the stego image.They are:

StegDetect: This software is an automated tool for detecting steganographic content in images. It is capable of detecting several different steganographic methods to embed hidden information in JPEG images. This software will run on the linux platform. Currently, the detectable schemes are jsteg, jphide, invisible secrets; OutGuess 01.3b, F5, appendX, and camouflage. Using linear discriminant analysis, it also supports detection of new stego systems. The main drawback of this software is it works only for JPEG images. Currently, there is no support for parameter training. The only exported knob is the sensitivity level. Future versions will export all detection parameters via a configuration file.



Home / Tools / stegdetect

List of all available tools for penetration testing.

stegdetect Summary

Description: An automated tool for detecting steganographic content in images.

Category: stego defensive forensic

Version: 19.ac1df7a

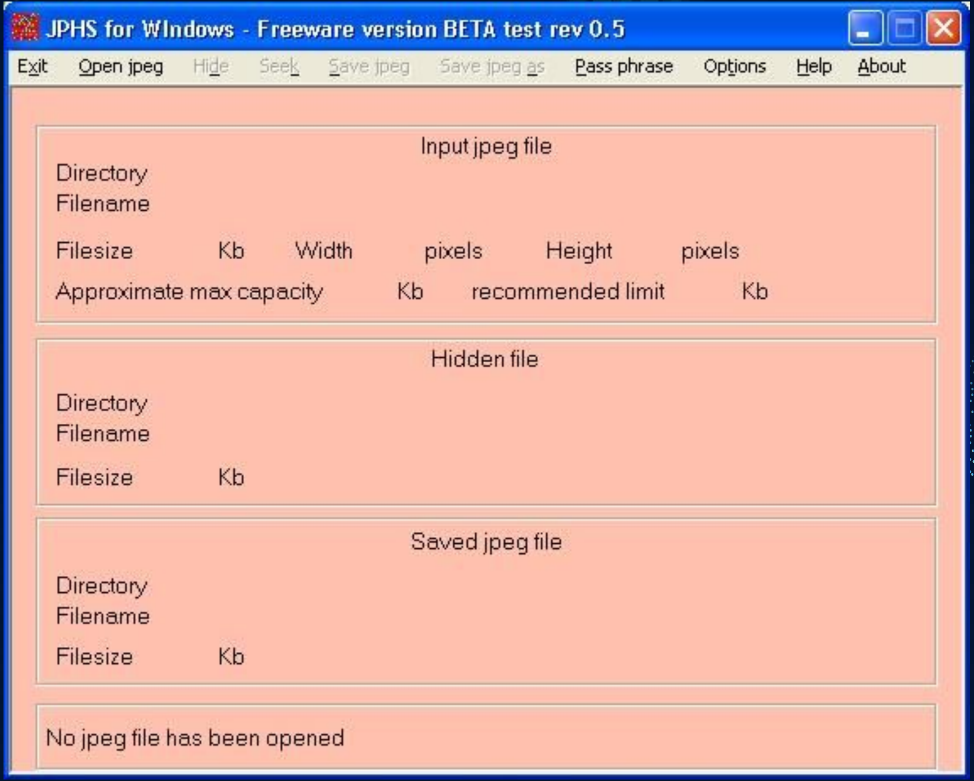
WebSite: https://github.com/redNixon/stegdetect

Last Updated: 2021-01-06

Added to the database: 2017-08-23

Description of StegDetect

JPSeek: It is a program that allows detecting the hidden message inside a jpeg image. There are various versions of similar programs available on the internet but JPSeek is rather special. The design objective is same as JPHide.



No jpeg file has been opened

StegSecret: StegSecret software aim is to gather, to implement and to make easier the usage of steganalysis techniques, especially in digital media such as images, audio and video. This software warns about the insecurity of several steganographic tools and steganographic algorithms available in Internet. It is a steganalysis open source project that makes possible the detection of hidden information in different digital media. StegSecret is java-based multiplatform steganalysis tool that allows the detection of hidden information by using the most known steganographic methods. It detects EOF, LSB, and DCT like techniques.

StegBreak: It launches brute-force dictionary attacks on JPG image. The StegBreak states a brute-force dictionary attack against the specified JPG images. And while comparing with the other tools, this is effectively work on JPG image formats.

REFERENCES

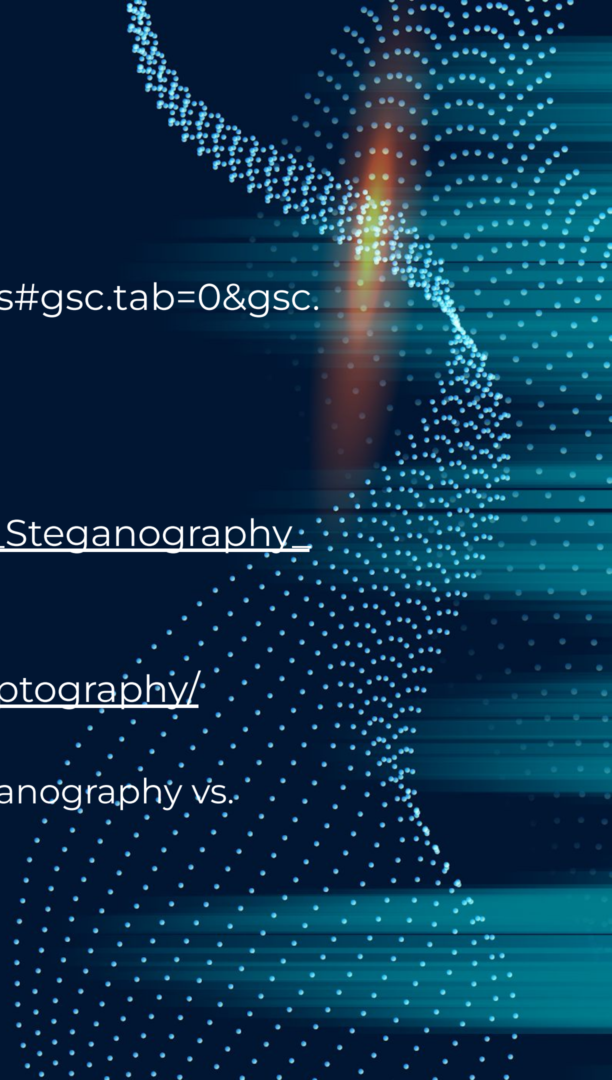
https://www.ieee.org/searchresults/index.html?q=steganalysis#gsc.tab=0&gsc.q=steganalysis&gsc.page=1

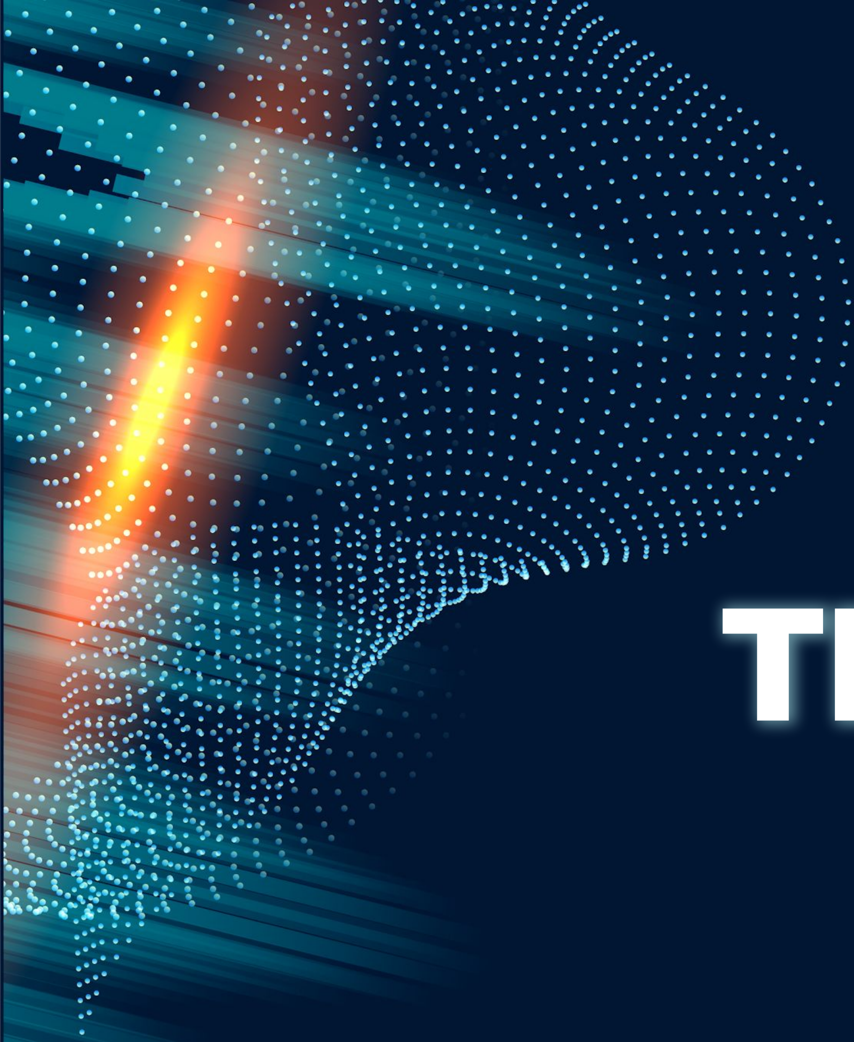
https://link.springer.com/search?query=steganalysis

https://www.researchgate.net/publication/292310394_Image_Steganography_Techniques_An_Overview

https://www.geeksforgeeks.org/image-steganography-in-cryptography/

Huaiqing Wang and Shuozhong Wang. 2004. Cyber warfare: steganography vs. steganalysis. Commun. ACM 47, 10 (October 2004), 76–82. https://doi.org/10.1145/1022594.1022597





THANK YOU