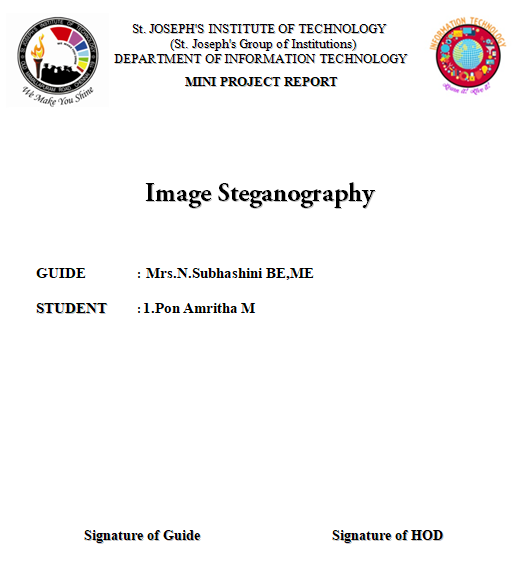
****

**Image Steganography**

Abstract:

Steganography is the process of hiding a secret message within a larger one in such a way that someone can not know the presence or contents of the hidden message. The purpose of Steganography is to maintain secret communication between two parties. This project will show how Steganography is used in a modern context while providing a practical understanding of what Steganography is and how to accomplish it.

**Introduction:**

Although related, Steganography is not to be confused with Encryption, which is the process of making a message unintelligible—Steganography attempts to hide the existence of communication. The basic structure of Steganography is made up of three components: the “carrier”, the message, and the key1 . The carrier can be a painting or a digital image. It is the object that will ‘carry’ the hidden message. A key is used to decode/decipher/discover the hidden message.

**Applications:**

Image Steganography has many applications, especially in today’s modern, hightech world. Privacy and anonymity is a concern for most people on the internet. Image Steganography allows for two parties to communicate secretly and covertly. It allows for some morally-conscious people to safely whistle blow on internal actions; it allows for copyright protection on digital files using the message as a digital watermark. One of the other main uses for Image Steganography is for the transportation of high-level or top-secret documents between international governments. While Image Steganography has many legitimate uses, it can also be quite nefarious. It can be used by hackers to send viruses and trojans to compromise machines, and also by terrorists and other organizations that rely on covert operations to communicate secretly and safely

Software Environment:

Software Environment is a technical specification of requirements of software product.This specifies the environment for development operation and maintenance of the product.

Technology used:

Python3

**Python3:**

**Python** an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language" \o "Interpreted language), [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), [general-purpose](https://en.wikipedia.org/wiki/General-purpose_programming_language) [programming language](https://en.wikipedia.org/wiki/Programming_language). Created by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) and first released in 1991. Its language constructs and [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help programmers write clear, logical code for small and large-scale projects. Python is [dynamically typed](https://en.wikipedia.org/wiki/Dynamic_programming_language) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). Python is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library). Python [interpreters](https://en.wikipedia.org/wiki/Interpreter_(computing)) are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility)

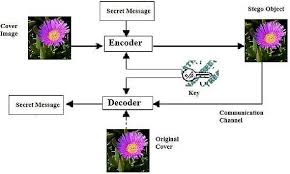
Packages Used:

tkinter

PIL

os

Architectural Design:



Implementation:

In my project I will first click on the open image button to choose the image from my device.Once the image is chosen it will display the path of the image in the output screen.Then click on the message button to encrypt a message now type any message on the text box.A duplicate image is produced along with the decoder button.Now click on the button it will automatically display the message.

**Source Code:**

from tkinter import \*

from PIL import ImageTk, Image

from tkinter import filedialog

import os

root = Tk()

root.title("IMG STENO GREAPHY")

root.geometry("550x300+300+150")

root.resizable(width=True, height=True)

s=StringVar()

def openf():

fname=filedialog.askopenfilename(title='open')

print(fname)

img=Image.open(fname)

img=img.resize((250,250),Image.ANTIALIAS)

img = ImageTk.PhotoImage(img)

panel = Label(root, image=img)

panel.image = img

panel.pack()

btn2 = Button(root, text='message',command=openf1).pack()

def openf1():

newwin = Toplevel(root)

L=Label(newwin,text="Enter Message:")

L.pack()

e=Entry(newwin,width=30,textvariable=s)

e.pack()

btn2 = Button(newwin, text='Encrypt', command=open2).pack()

def open2():

win1 = Toplevel(root)

btn1 = Button(win1, text='decoder',command=open3)

btn1.pack()

image=ImageTk.PhotoImage(Image.open("C:\\Users\\DELL\\Desktop \\Wallpaper\\bg1.jpg"))

panel = Label(win1, image = img)

panel.pack()

root.mainloop()

def open4():

win2 = Toplevel(root)

label=Label(win2,text="IMAGE ENCODER VALUES",bg="blue")

label.pack()

d=s.get()

lab2=Label(win2,text=d,font=20).pack()

root.mainloop()

def open3():

win1 = Toplevel(root)

d=s.get()

lab2=Label(win1,text=d,font=20).pack()

#im2 = Image.open(im1)

#stegoImage = stepic.decode(im2)

#stegoImage.show()

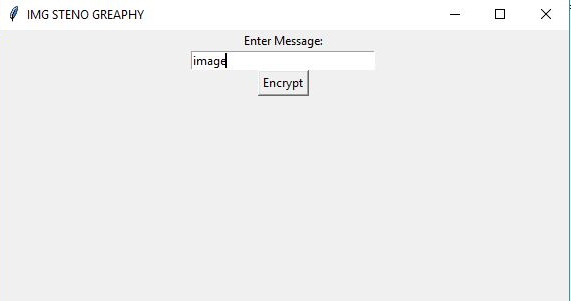
btn1 = Button(root, text='open image', command=openf).pack()

root.mainloop()

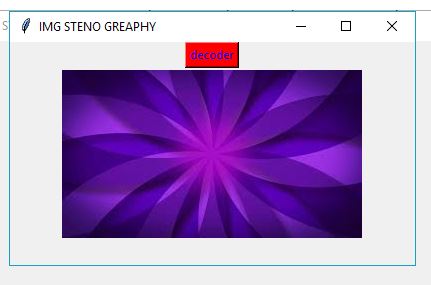
Snapshots:

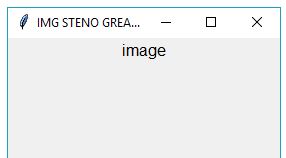
Encryption:





Decryption:





Conclusion:

I have learned that while implementing Image Steganography is important, thinking of how to detect and attack it and the methods to do so are far more complex than actually doing the Steganography itself.