**INFORMATION CODING TECHNIQUES**

**SUBJECT CODE:IT220**

****

**MINI PROJECT ON**

Image Steganography in Python:

An Exploration of Data Security and Concealed Communication

|  |  |
| --- | --- |
| **Submitted by,** | **Submitted to,** |
| Bhuvaneshwari B  21EE1013  Siva Darshini N  21IT1051 | Dr.Maragathavalli.P  Assistant Professor  Department of Information Technology |

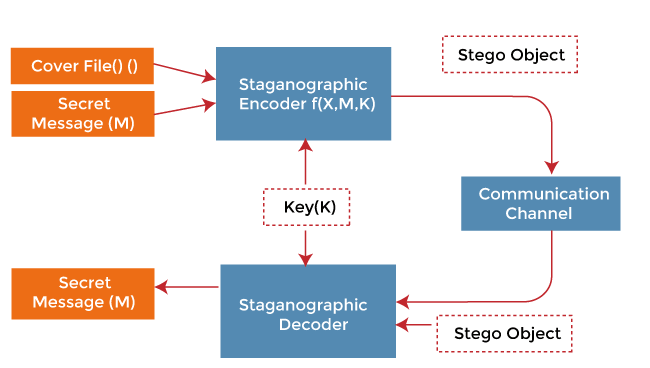
**AIM:**

The aim of this mini project is to implement a simple image encryption and decryption system using Python. The encryption process involves embedding text data into the least significant bits of the image pixels, while the decryption process extracts and reconstructs the original text from the encrypted image.

**INTRODUCTION:**

Image steganography is a type of information security method that involves concealing hidden messages in ordinary digital photographs to communicate secretly. In other words, image steganography project is the practice of concealing sensitive information within an image or video file.

**Basic Model of Steganography**



**Advantages and Disadvantages:**

a. **Advantages:**

1. **Educational Value:**
   * **Advantage:** The project serves as an educational tool for individuals learning about image processing and basic cryptographic techniques.
   * **Details:** Users can gain hands-on experience with image manipulation and understand the principles of hiding information within image pixels.
2. **Simple Implementation:**
   * **Advantage:** The implementation of image encryption and decryption is straightforward, making it accessible for beginners.
   * **Details:** The simplicity of the algorithm allows users to grasp the core concepts without the complexities of advanced cryptographic methods.
3. **User Interface (Tkinter):**
   * **Advantage:** The integration of Tkinter provides a user-friendly interface for image selection, encryption, and decryption.
   * **Details:** The graphical interface enhances user interaction, making the application more accessible and intuitive.

b. **Disadvantages:**

1. **Limited Security:**
   * **Disadvantage:** The encryption method used in the project is basic and may not provide robust security for highly sensitive data.
   * **Details:** Sophisticated cryptographic techniques are necessary for applications requiring a higher level of security.
2. **Pixel Alteration:**
   * **Disadvantage:** Image encryption involves altering pixel values, potentially leading to visible artifacts or degradation in image quality.
   * **Details:** Depending on the image content and the extent of pixel manipulation, the encrypted image may exhibit noticeable changes.
3. **Algorithm Complexity:**
   * **Disadvantage:** The simplicity of the encryption algorithm is a limitation for scenarios where more advanced and secure encryption methods are required.
   * **Details:** For applications demanding higher security standards, advanced cryptographic algorithms with proven security properties are recommended.
4. **Potential for Visible Patterns:**
   * **Disadvantage:** The basic nature of the encryption algorithm may result in predictable patterns in the encrypted images.
   * **Details:** This predictability could be exploited by attackers, emphasizing the need for more sophisticated encryption techniques to thwart potential threats.

**MODULES USED:**

**a. OpenCV (cv2):**

Purpose: OpenCV is a powerful computer vision library that provides tools for image processing, computer vision, and machine learning. In this project:

Used for reading images from file paths (cv2.imread).

Enables manipulation of image data in the form of NumPy arrays.

Essential for iterating through pixels and altering pixel values during the encryption process.

**b. PIL (Python Imaging Library):**

Purpose: PIL, now known as the Pillow library, is a versatile image processing library for Python. In this project:

Utilized for opening, resizing, and displaying images (Image.open, Image.thumbnail).

Facilitates the conversion between different image data representations (e.g., NumPy arrays to PIL images).

**c. Tkinter:**

Purpose: Tkinter is the standard GUI toolkit for Python. In this project:

Used to create a user-friendly graphical interface for the image encryption and decryption application.

Provides buttons (Button), labels (Label), and text boxes (Text) for user interaction.

Enables the display of images within the GUI (Label with ImageTk.PhotoImage).

**d. NumPy:**

Purpose: NumPy is a fundamental package for scientific computing with Python. In this project:

Used for efficient manipulation of image data represented as arrays.

Facilitates operations on images as numerical data, enhancing computational efficiency.

Enables conversion between Python lists and arrays for convenient data handling.

**ALGORITHM:**

**The algorithm for encrypting data into the image is as follows:**

1. Load the image and write the text in the text box provided below
2. Convert the message into an array representation of the ASCII letters.
3. Compute the number of pixels required, which is equal to the 3 times the length of the array of ASCII letters
4. Number of rows required = number of pixels required / width of the image
5. Traversing the image row-wise, we will check for the following conditions:
6. Check the number of pixels traversed. If the bit is 1 and the pixel value is an even number, make it an odd number by subtracting 1. Similarly, if the bit is 0 and the pixel value is an odd number, make it an even number by subtracting 1.
7. Keep a count of the number of letters using the count variable.
8. If the index is 7, check if the next character exists. If yes, mark the EOF bit as 0 and continue. Else, mark as 1 and end.
9. We have successfully encrypted the image into the file.

**The algorithm to decode the encrypted file is as follows:**

1. Open the encrypted image and convert it into a numpy array.
2. Obtain the data from the image by going through the encryption algorithm.

* Every pixel in every row has 1 bit of information, which is added into the data variable, using the for loop.
* Check if the EOF character is reached.
* If yes break from the for loop
* Otherwise, continue.
* The ASCII is stored serially in the data variable.

1. After obtaining the ASCII bits, bits are grouped into letters by making groups of 8.
2. The letters are stored in the message variable, which is linked using the join command in python.
3. Finally, the proper message is shown on the screen.

**CODE IMPLEMENTATION:**

**Encrypt.py**  
from tkinter import \*

from PIL import Image, ImageTk

from tkinter import filedialog

import cv2

import numpy as np

import math

global path\_image

image\_display\_size = 300, 300

def on\_click():

# Step 1.5

global path\_image

# use the tkinter filedialog library to open the file using a dialog box.

# obtain the image of the path

path\_image = filedialog.askopenfilename()

# load the image using the path

load\_image = Image.open(path\_image)

# set the image into the GUI using the thumbnail function from tkinter

load\_image.thumbnail(image\_display\_size, Image.ANTIALIAS)

# load the image as a numpy array for efficient computation and change the type to unsigned integer

np\_load\_image = np.asarray(load\_image)

np\_load\_image = Image.fromarray(np.uint8(np\_load\_image))

render = ImageTk.PhotoImage(np\_load\_image)

img = Label(app, image=render)

img.image = render

img.place(x=20, y=50)

def encrypt\_data\_into\_image():

# Step 2

global path\_image

data = txt.get(1.0, "end-1c")

# load the image

img = cv2.imread(path\_image)

# break the image into its character level. Represent the characyers in ASCII.

data = [format(ord(i), '08b') for i in data]

\_, width, \_ = img.shape

# algorithm to encode the image

PixReq = len(data) \* 3

RowReq = PixReq/width

RowReq = math.ceil(RowReq)

count = 0

charCount = 0

# Step 3

for i in range(RowReq + 1):

# Step 4

while(count < width and charCount < len(data)):

char = data[charCount]

charCount += 1

# Step 5

for index\_k, k in enumerate(char):

if((k == '1' and img[i][count][index\_k % 3] % 2 == 0) or (k == '0' and img[i][count][index\_k % 3] % 2 == 1)):

img[i][count][index\_k % 3] -= 1

if(index\_k % 3 == 2):

count += 1

if(index\_k == 7):

if(charCount\*3 < PixReq and img[i][count][2] % 2 == 1):

img[i][count][2] -= 1

if(charCount\*3 >= PixReq and img[i][count][2] % 2 == 0):

img[i][count][2] -= 1

count += 1

count = 0

# Step 6

# Write the encrypted image into a new file

cv2.imwrite("encrypted\_image.png", img)

# Display the success label.

success\_label = Label(app, text="Encryption Successful!",

bg='lavender', font=("Times New Roman", 20))

success\_label.place(x=160, y=300)

# Step 1

# Defined the TKinter object app with background lavender, title Encrypt, and app size 600\*600 pixels.

app = Tk()

app.configure(background='lavender')

app.title("Encrypt")

app.geometry('600x600')

# create a button for calling the function on\_click

on\_click\_button = Button(app, text="Choose Image", bg='white', fg='black', command=on\_click)

on\_click\_button.place(x=250, y=10)

# add a text box using tkinter's Text function and place it at (340,55). The text box is of height 165pixels.

txt = Text(app, wrap=WORD, width=30)

txt.place(x=340, y=55, height=165)

encrypt\_button = Button(app, text="Encode", bg='white', fg='black', command=encrypt\_data\_into\_image)

encrypt\_button.place(x=435, y=230)

app.mainloop()

**decrypt.py**import cv2

from tkinter import filedialog, Tk, Button, Label

from PIL import Image, ImageTk

import numpy as np

image\_display\_size = 500, 350

def decrypt():

# load the image and convert it into a numpy array and display on the GUI.

load = Image.open("./encrypted\_image.png")

load.thumbnail(image\_display\_size, Image.ANTIALIAS)

load = np.asarray(load)

load = Image.fromarray(np.uint8(load))

render = ImageTk.PhotoImage(load)

img = Label(app, image=render)

img.image = render

img.place(x=100, y=50)

# Algorithm to decrypt the data from the image

img = cv2.imread("./encrypted\_image.png")

data = []

stop = False

for index\_i, i in enumerate(img):

i.tolist()

for index\_j, j in enumerate(i):

if((index\_j) % 3 == 2):

# first pixel

data.append(bin(j[0])[-1])

# second pixel

data.append(bin(j[1])[-1])

# third pixel

if(bin(j[2])[-1] == '1'):

stop = True

break

else:

# first pixel

data.append(bin(j[0])[-1])

# second pixel

data.append(bin(j[1])[-1])

# third pixel

data.append(bin(j[2])[-1])

if(stop):

break

message = []

# join all the bits to form letters (ASCII Representation)

for i in range(int((len(data)+1)/8)):

message.append(data[i\*8:(i\*8+8)])

# join all the letters to form the message.

message = [chr(int(''.join(i), 2)) for i in message]

message = ''.join(message)

message\_label = Label(app, text=message, bg='lavender', font=("Times New Roman", 10))

message\_label.place(x=30, y=400)

# Defined the TKinter object app with background lavender, title Decrypt, and app size 600\*600 pixels.

app = Tk()

app.configure(background='lavender')

app.title("Decrypt")

app.geometry('600x600')

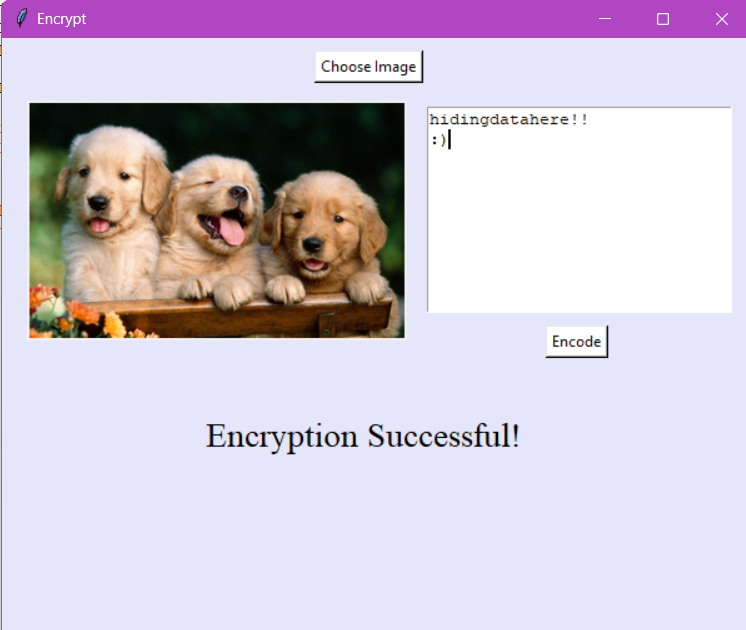
# Add the button to call the function decrypt.

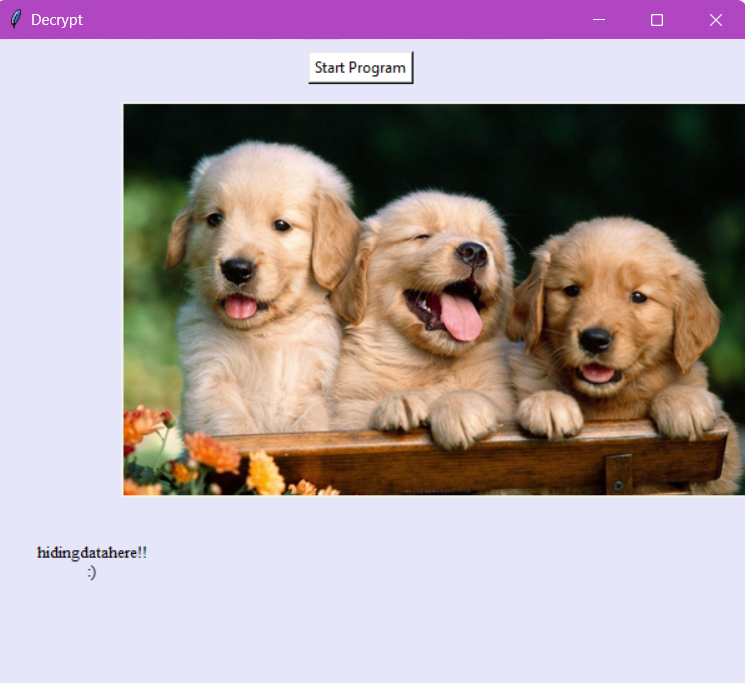
main\_button = Button(app, text="Start Program", bg='white', fg='black', command=decrypt)

main\_button.place(x=250, y=10)

app.mainloop()

**OUTPUT:**





**CONCLUSION:**

In conclusion, this mini project offers an entry-level exploration into image encryption and decryption using Python. While the implemented encryption algorithm is rudimentary, the project lays the foundation for understanding more advanced cryptographic techniques. The inclusion of a user interface enhances the accessibility and usability of the application, making it a valuable resource for educational purposes.

**REFERENCES:**

[**https://www.geeksforgeeks.org/python-gui-tkinter/**](https://www.geeksforgeeks.org/python-gui-tkinter/)

[**https://www.javatpoint.com/python-pillow**](https://www.javatpoint.com/python-pillow)

[**https://www.javatpoint.com/image-steganography-using-python**](https://www.javatpoint.com/image-steganography-using-python)

[**https://ieeexplore.ieee.org/document/9120935**](https://ieeexplore.ieee.org/document/9120935)