CONCEALING OF DATA IN DIGITAL IMAGES

A PROJECT REPORT

Submitted by

ABISHEK A (715517104004)

AKASH M (715517104006)

KARTHIKEYAN S (715517104031)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH, COIMBATORE 641 062

ANNA UNIVERSITY: CHENNAI - 600 025

APRIL 2020

ANNA UNIVERSITY: CHENNAI - 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"CONCEALING OF DATA IN DIGITAL IMAGES"** is the bonafide work of **"ABISHEK A (715517104004) , AKASH M (715517104006), KARHIKEYAN S (715517104031)"** who carried out the project work under my supervision.

----------	**----------**
SIGNATURE	**SIGNATURE**
Dr. R. Manimegalai	Dr. I Kala
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Computer Science and Engineering	Associate Professor Department
PSG Institute of Technology and	Computer Science and Engineering,
Applied Research,	PSG Institute of Technology and
	Applied Research,
Coimbatore – 641 062	Coimbatore – 641 062

Submitted for the project viva-voce Examination held on _____

----------	**----------**
INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ABSTRACT

Generally security in data transmission is the most important for a successful data transmission. So if the data is not secure then arises the opportunity for a hacker to hack into your details. To prevent such hackers we have many algorithms and methods for secure data transmission. Among which we have Cryptography and Steganography. Security of Cryptography and Steganography is minimum when individual and is maximum when they both are simultaneousy used in designing an application

Steganography is the art of hiding the fact that communication is taking place, by hiding information in some Multimedia. Many different carrier formats can be used ,but digital images are the most popular because of their frequency on the Internet . For hiding the secret information in images , there exists a large variety of stenographic techniques some are more complex than other and all of them have respective strong and weak points. Different applications have different requirements of the steganography technique used. For Example some applications may require absolute invisibility of the secret information, while others require a larger secret message to be hidden. This project intends to give an overview of Image Steganography, its uses and techniques.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
BMP	Bitmap Image
CLI	Command Line Interface
GIF	Graphics Interchange Format
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
JPEG	Joint Photographic Experts Group
JSON	Javascript Object Notation
LSB	Least Significant Bit
MSB	Most Significant Bit
PDF	Portable Document Format
PNG	Portable Network Graphic
RAW	Raw Image Formats
TIFF	Tagged Image File

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

"Steganography is the art and science of communicating in a way which hides the existence of the communication. In contrast to Cryptography, where the enemy is allowed to detect, intercept and modify messages without being able to violate certain security premises guaranteed by the cover message with the embedded cryptosystem. The goal of Steganography is to hide messages inside other harmless images in a way that does not allow any enemy to even detect that there is a second message present".

In image steganography the information is hidden exclusively in images. In Histories the Greek historian Herodotus writes of a nobleman, Histaeus, who needed to communicate with his son-in-law in Greece. He shaved the head of one of his most trusted slaves and tattooed the message onto the slave's scalp. When the slave's hair grew back the slave was dispatched with the hidden message.

In the Second World War the Microdot technique was developed by the Germans. Information, especially photographs, was reduced in size until it was the size of a typed period. Extremely difficult to detect, a normal cover message was sent over an insecure channel with one of the periods on the paper containing hidden information.

The below represents various Data Hiding Techniques

- Data Hiding Techniques in Still Images
- Data Hiding Techniques in IPv4 Header
- Data Hiding Techniques in Video Sequences
- Data Hiding Techniques using DNA Sequences

1.2 OBJECTIVE

The main objective of this project is to perform secret data transmission through images. And the other challenges are lack of proper communication between the higher authorities, is to hide a secret message within a cover-media in such a way that others cannot discern the presence of the hidden message and transferring secret messages which will also be fulfilled providing an interface in the application.

1.3 PROBLEM STATEMENT

In today's scenario security of data is a very big challenge in any forms of communication. The Concealing of data in Digital Images is a science of hiding sensitive and secret information in another transmission medium to achieve secure and secret communication. The lack of secure communication between people made the hackers take their private data very easily. Concealing of data in Digital Images Framework is a Python based application which aims to develop a user interface between the end users and authorities. It provides an interface for data assembling between the users and the authorities. Framework implementation is done using four step processing.

1.4 PROJECT OVERVIEW

In today's scenario security of data is a very big challenge in any forms of communication. The Concealing of data in Digital Images is a science of hiding sensitive and secret information in another transmission medium to achieve secure and secret communication. The lack of secure communication between people made the hackers take their private data very easily. Concealing of data in Digital Images Framework is a Python based application which aims to develop a user interface between the end users and authorities. It provides an interface for data

assembling between the users and the authorities. Framework implementation is done using four step processing.

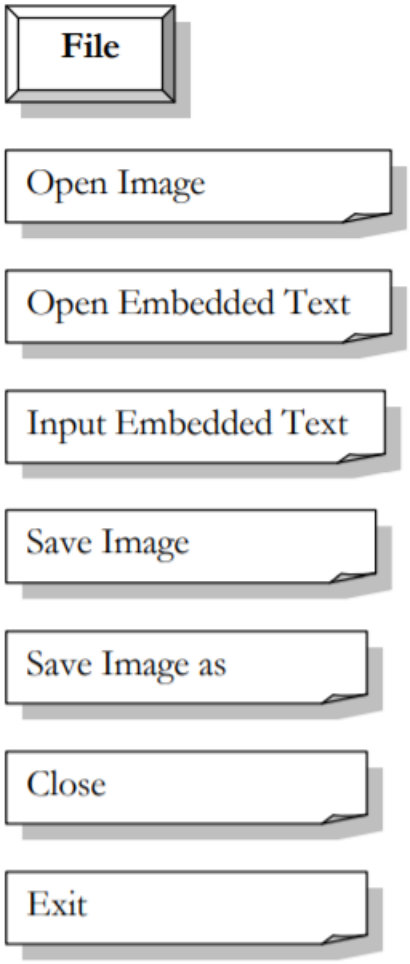

Fig 1.1 Flow of data in Data Hiding

1.5 SCOPE AND MOTIVATION

Without Concealing of data in Digital Images Framework, there is a leakage of data almost 70-80 GB of data in day to day life. Upon implementing the Framework, the leakage can be reduced up to 20%. Concealing of data in digital Images is an application interface for the end users to transmit their data based on their willingness. It brings out the liberty of transferring data. A single application

is developed with two GUI's one on the sender side and the other on the receiver side be deployed for both sender and receiver .

1.6 DIFFERENT APPROACHES

There are various approaches in concealing of data. Some of them are given as follows:

1. Cryptography
2. Steganography
3. Watermarking
4 . Reversible data hiding

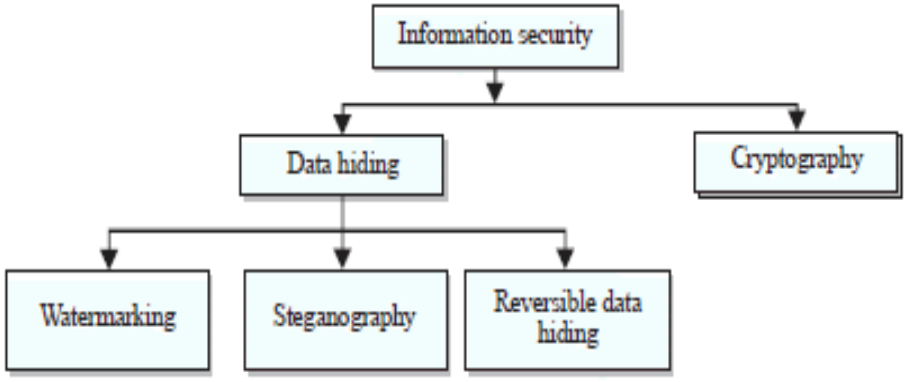

Fig 1.2 Different approaches of Data Hiding

- **Cryptography**

  Cryptography is technique of securing information and communications through use of codes so that only those person for whom the information is intended can understand it and process it. Thus preventing unauthorized access to information. The prefix "crypt" means "hidden" and suffix graphy means "writing".

  In Cryptography the techniques which are use to protect information are obtained from mathematical concepts and a set of rule based calculations known as algorithms to convert messages in ways that

make it hard to decode it. These algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on internet and to protect confidential transactions.

- **Steganography**

The word Steganography is derived from two Greek words- ‘stegos’ meaning ‘to cover’ and ‘grafia’, meaning ‘writing’, thus translating to ‘covered writing’, or ‘hidden writing’. Steganography is a method of hiding secret data, by embedding it into an audio, video, image or text file. It is one of the methods employed to protect secret or sensitive data from malicious attacks.

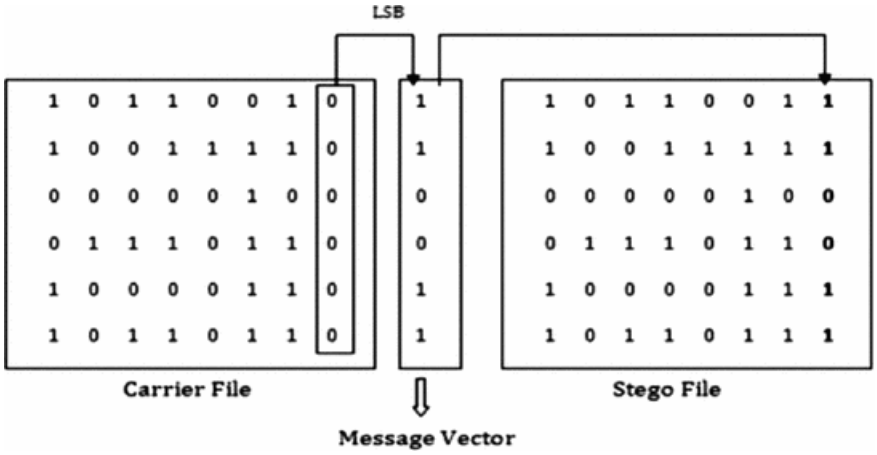

Fig 1.3 LSB Steganography

- **Watermarking**

Digital watermarking is the method of embedding data into digital multimedia content. This is used to verify the credibility of the content or to recognize the identity of the digital content's owner. Digital watermarking can be employed for multiple purposes, such as: Copyright protection

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING SYSTEMS

The number of papers dealing with Data hiding Framework for security in literature is growing exponentially. Several researchers have played a significant role in the development of this application.

Hideki Noda et al. [1] The JPEG compression using the discrete cosine transform (DCT) is still the most common compression standard for still images. QIM(Quantization Index Modulation) is applied in DCT(Discrete Cosine Transformation) Domain. DCT based steganography techniques are immune to Histogram based attacks. Two different quantizes are used with QIM, one for embedding '0' and another for embedding '1'. Another method called HM-JPEG(Histogram Matching JPEG) Steganography method is also presented along with QIM-JPEG Steganography. In these two methods embedding of secret message takes place during quantization of DCT coefficients only, not by modifying quantized DCT coefficients.

H. Motameni et al. [2] the authors have proposed a novel technique for hiding text message in a grayscale image. In this method different colors in the cover image are labeled in order to identify dark regions in the image. Data embedding in the these darker regions results in high quality stego images. This method offers more security than other LSB techniques

Chung-Ming Wang et al. [3] this work is an improvement over Wu and Tsai scheme of pixel value differencing (2003). In this method the image is divided in to the blocks of two consecutive pixels and the number of bits that can be embedded is determined from the width of the range table. The reminder of sum of two pixel values with width of suitable range is calculated and modulus of pixel values is adjusted to the decimal value of binary string to be embedded in the block of two consecutive pixels. This method also addresses the falling-off boundary problem and produces high quality stego images than any other technique of spatial domain steganography. But the hiding capacity is low in this method when compared to other methods.

Adem Orsdemir et al. [4] this method is based on the Higher Order Statistics Steganalysis. Generally any steganographer focuses more on undetectability and payload but not about the statistical difference between the stego image and cover image. When the steganographer is well aware of the steganalysis methods HOS steganalyzer and by formulating statistical in distinguish ability requirement, visual quality requirement, and detect ability requirement the method of steganography can withstand the steganalysis methods based on statistical differences.

Ahmad T. Al-Taani and Abdullah M. AL-Issa [5] the proposed method provides good quality and high embedding capacity of stego image. Here the carrier image is divided into blocks of equal sizes and then stuffs the original data bits in the edge of the block depending on the number of ones in left four bits of the pixel. Experimental results of this method are compared with Pixel Value Differencing method and Gray Level Modification Method.

C.-C.Chang and W.-C. Wu [6] this paper provides a technique to improve the embedding capacity without reducing the quality of cover file. That technique is called an adaptive VQ-based data hiding scheme based on a codeword clustering technique. Adaptive embedding method is superior to the fixed embedding method in terms of embedding capacity and stego-image quality.

Mei-Yi Wu et al. [7] this paper presents a new iterative method of image steganography based on palette which reduces the Root Mean Square error between an original image and its corresponding stego-image. Based on a palette modification scheme, which can embed one message bit into each pixel in a palette-based image iteratively. The cost of removing an entry color in a palette and the profit of generating a new color to replace the old color are calculated. If the maximal profit exceeds the minimal cost, an entry color is replaced in iteration.

Sorina Dumitrescu et al.[8] This paper proposes a new steganalysis technique to detect LSB steganography in digital signals such as image and audio. This technique is based on statistical analysis of sample pairs. By this technique the length of hidden message embedded via LSB steganography can be estimated with high precision.

Andrew D. Ker [9] Detecting LSB matching steganography is quiet difficult compared to the LSB replacement steganography. In this paper Histogram characteristic function (HCF) is used for the detection of steganography in color images, but it cannot be used for gray scale images.

Po-Yueh Chen and Hung-Ju Lin [10] this paper proposes a new image steganographic method based on frequency domain embedding. The frequency

domain transform applied in this method is Haar-DWT. There are three regions i.e., low frequency region, middle frequency region and high frequency region. And embedding occurs in Middle frequencies.

2.2 DRAWBACKS IN THE EXISTING SYSTEM

- The current framework makes the data retrieval very difficult because of huge volume of document.
- Data security and Integration are major concern in current system
- The current framework does not have a PC based database for storing of documents.
- The entire process is tedious and a waste of human and material assets.
- Record management and searching is a very difficult job.
- The time is taken to encode and decode the secret message is very high
- During transmission there is a possibility of Data Loss

2.3 PROPOSED SYSTEM

The original aims of the paper are to introduce a technique for hiding a text file, which techniques hide a secret text file inside an image file, and the modified image must be similar to the original image, in other words the changes that happen on the modified image mustn't be visible, or the human eye would be unable to notice it, The project application loads 24-bit BMP, GIF, and JPG image format, embed data into them using Sunflower system and saves the images. Encryption can be used before embedding the data to provide robustness. Finally the application can also extract data that was previously embedded. The

application runs in a user friend Windows environment where the user can view the image, before and after the embedding

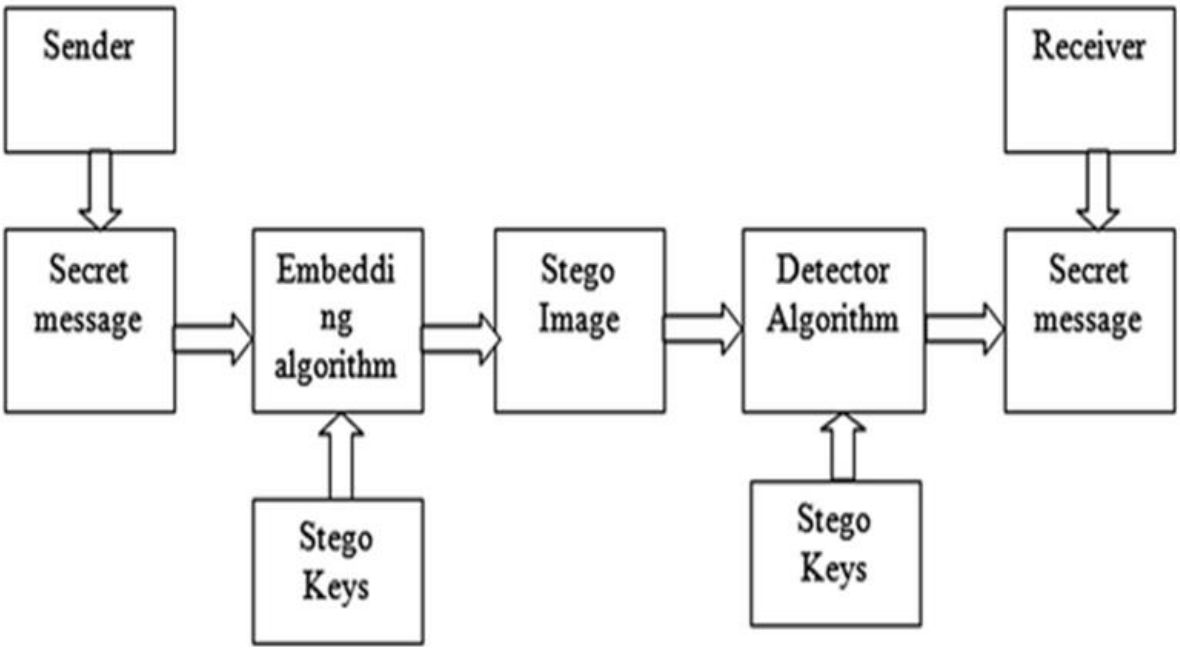

Fig 2.1 Block Diagram of proposed system

2.4 BENEFITS OF THE PROPOSED SYSTEM

- Power supply does not affect the operation of the current system.
- The current system can be used by both computer literates and non-computer literates.
- The security will help in easy retrieval of information and control data concurrency.
- The use of password will be incorporated to maintain and ensure data security and integrity.

CHAPTER 3

SYSTEM ANALYSIS AND DESIGN

3.1 Software and Hardware Requirements

Software Requirements

- Operating System : Windows 7 and higher versions, Any Linux distributions.
- Language : Python 3.6
- Packages Used : pandas, tkinter, matplotlib

Hardware Requirements

- RAM : 8GB (recommended) and more
- Processor : Intel Core i5 and more
- Disk capacity : 100 GB (Minimum) and more
- Speed : 1GHZ and more

3.2 Brief details of the Requirements

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.

Microsoft .Net framework prepares a huge amount of tool and options for programmers that they simples programming. One of .Net tools for pictures and images is auto-converting most types of pictures to BMP format

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 first layer 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 take the image file as an output, and give two file at destination folder, one is the same image file and another is the message file that is hidden it that.

Before encrypting file inside image we must save name and size of file in a definite place of image. We could save file name before file information in LSB layer and save file size and file name size in most right-down pixels of image. Writing this information is needed to retrieve file from encrypted image in decryption state.

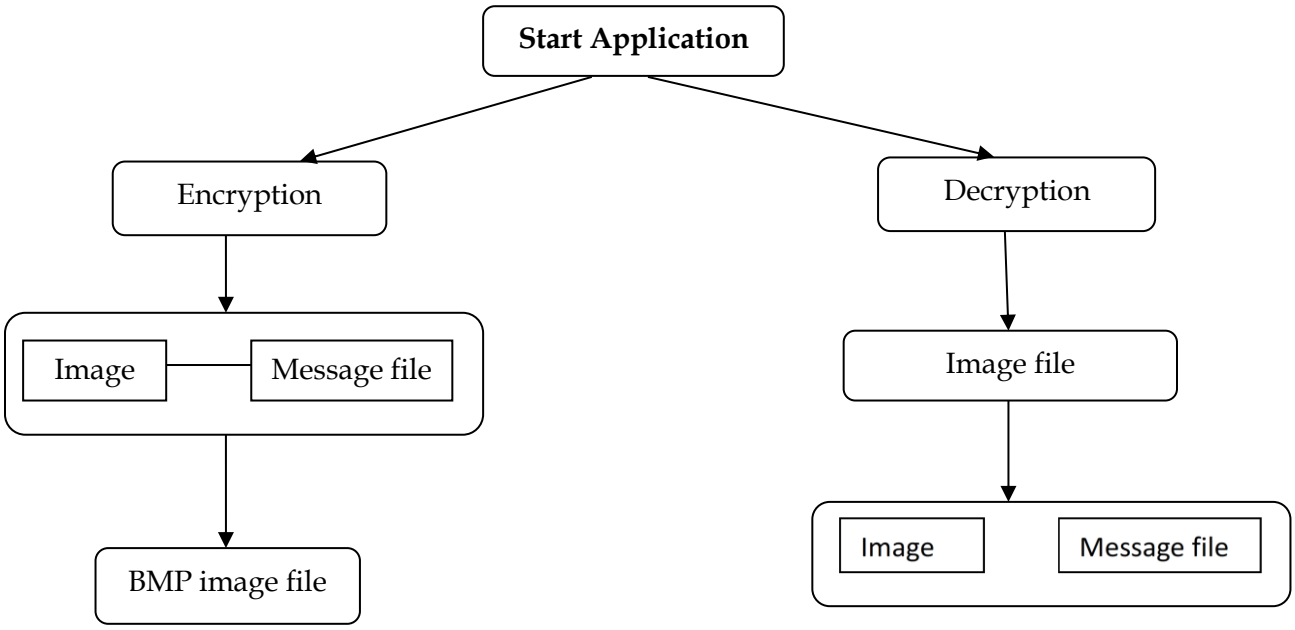

Fig 3.1 Level view of Application

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 SYSTEM FLOW

The entire work-flow of Concealing of data in digital images Application is as follows:

The sender initiates the request from the sender's application. Then receiver processes the request by generating the keys and then shares the pubic key with the sender then the sender encrypts and encodes the secret data using the public keys and generates an image. That image is sent to the receiver and using his private key the secret data from the image can be decoded and decrypted from the image. The possibility of the intruders hacking the system is minimal.

4.2 SETTING UP ENVIRONMENT

The python software is downloaded, installed and worked with its Command Line Interface (CLI). Using Tkinter library the graphical user interface is developed for the application on both sides of the application namely sender side and receiver side. Preprocessing of image is done using SciKit library in python. A module called intruder introspect is included which generates a checksum on both sender and receiver side information in order to avoid unauthorized access by hackers.

4.3 MODULE IMPLEMENTATION

Proper communication or exchange of information is maintained between the sender and the receiver.

SENDER MODULE

The sender is the initiator to the request. Then the sender receives a public key from the receiver. The public key is mainly used to encrypt using Rivest Shamir Adleman Algorithm. And then the encrypted message is encoded into the image using Least Significant bit Algorithm of encoding. LSB encoding is preferred because the error rate is about 0.9% and the MSB algorithm is not preferred because the error rate is about 90%. The encoded image is then generates a duplicate image and sends it to the receiver.

RECEIVER MODULE

The receiver module gets the request from the sender for the key. Then, the receiver generates the public and private keys and sends the public keys to the user and then the sender encrypts the data using public key and then encodes the data using LSB algorithm and is preferred as the error ratio is minimal when compared to MSB encoding . The image developed as a result is sent to the receiver and the receiver decodes the message from the images and then decrypts the message which leads to a final actual message.

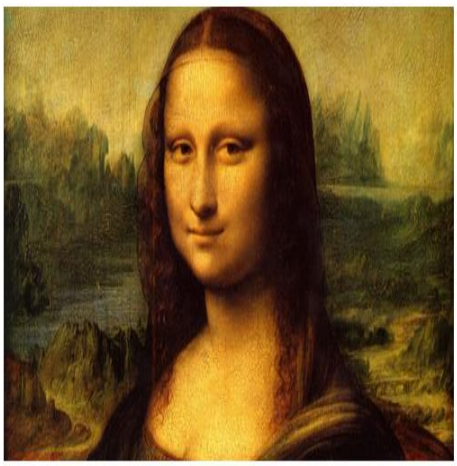

Fig 5.1 Original Image

Fig 5.2 Stego Image

4.4 CODE IMPLEMENTATION

The implementations involve the files namely the sender.py and receiver.py

- **Sender.py** – The sender.py is a set of python code when encryption and encoding is done and then sends the image to the receiver. Here in the sender.py the encryption is done using RSA Algorithm and encoding of secret data is done using LSB encoding technique. A GUI is developed using python Tkinter library for better user experience.

- **Reciever.py –** The reciever.py is a set of python code when decryption and decoding is done and then retrieves the secret data from the image. Here in the reciever.py the decryption is done using RSA Algorithm and decoding of secret data is done using LSB decoding technique. A GUI is developed using python Tkinter library for better user experience.

Sender.py

```
from tkinter import *

from PIL import ImageTk, Image

from decimal import Decimal

import random

import math

filename , photovar  = "" , "3.jpg" #Assigning Values

def count_largest(s):  #function for finding largest in an array

        m = 0
```

```
            for i in s :
                    if(m < len(i)):     m = len(i)
            return m
def gcd(x, y):  # function for calculating gcd
  while(y):      x, y = y, x % y
  return x
def getpubkey():   #function for getting public key
   f=open("pubkey.txt","r")
   key=f.read()
   text1.insert(0.0,str(key))
def nearest_roundoff(n):   #function for rounding off to nearest value
  if(n > 0):     return math.ceil(n/3.0) * 3
  elif( n < 0):     return math.floor(n/3.0) * 3
  else:     return 3
def isPrime(n) :    #function to validate wheather a number is prime or not a prime
   if(n in [2,3]):       return True
   else:       k=0
      while(True):
         if(n in [((6*k)-1),((6*k)+1)]):              return True
         else:
            if(( ((6*k) - 1) or ((6*k) + 1) ) > n):                return False
            k = k + 1
```

```
def get_num_pixels(filename):   # function to get number of pixels from an image
    width, height = Image.open(filename).size
    return width*height
def binary_to_decimal(v):  # function for binary to decimal conversion
    v = v.strip()
    v , dec_value , i = v[::-1] , 0 , 0
     for j in v:
        dec_value = dec_value + (int(j)*(2**i))
        i = i + 1
    return dec_value
def browse():       #Function for Opening Dialog
    global filename
    global photovar
    global firstimg
    global image3
    filename = filedialog.askopenfilename()
    var = ""
    for i in filename:
        var = var + i
        if (i == "/"):          var = ""
    filename , photovar = var , var
    image3 = ImageTk.PhotoImage(Image.open(photovar))
```

```
    canvas.itemconfig(firstimg, image=image3)
def encode():       #Encoding
    p,q=53,71
    n , z =p*q , (p-1)*(q-1)
    msg = text.get(0.0, END)
    publickey = text1.get(0.0, END)
    file=filename
    imag = Image.open(file,'r')
    hieght , width = Image.open(file,'r').size
    pixel_value = list(imag.getdata())
    imdata = iter(imag.getdata())
    pub_key = int(publickey)
    enc=""
    for i in msg:
        ctt = Decimal(0)
        ctt =pow(ord(i),pub_key)
        ct = ctt % n
        enc = enc + chr(ct)
    a=[]
for i in enc:
        w=bin(ord(i))[2:]
        w='0'*(8-len(w)) + w
```

```
        a.append(w)
maxlen = count_largest(a)
maxlen = nearest_roundoff(maxlen)
for  i in range(len(a)):
        a[i] = '0'*(maxlen - len(a[i]))  + a[i]
img = imag.load()
track , lb , bb , datadone , it , it1 = 0 , 0 , 0 , 0 , 0 , 0
r , g , b = img[lb,bb]
binary_value = (bin(len(msg)))[2:]
temp = '0'*(16 - len(binary_value)) + binary_value
r , g , b= binary_to_decimal(temp[0:8]) , binary_to_decimal(temp[8:]) , maxlen
img[lb,bb] = (r,g,b)
bb , rarr , garr , barr = bb + 1 , [] , [] , []
 for i in range(maxlen):
        if(i % 3 == 0):           rarr.append(i)
        elif(i % 3 == 1):           garr.append(i)
        elif(i % 3 == 2):           barr.append(i)
    while(lb <= hieght - 1):
        while(bb <= width - 1 ):
            r , g , b = img[lb,bb]
            if(it >= len(a)):
                    print("ExitPoint")
```

```
        datadone = 1
        break
t = a[it]
t = list(t)
if(it1 in rarr):
   if(t[it1] == '0' ):
      if(r % 2 != 0):
         r = r - 1
         img[lb,bb] = (r,g,b)
   elif(t[it1] == '1'):
      if(r % 2 != 1):
         r = r - 1
         img[lb,bb] = (r,g,b)
   it1 = it1 + 1
track = track + 1
if(it1 >= len(t)):
   it1 , t = 0 , a[it]
   t , it = list(t) , it + 1
   if(track > maxlen * len(msg)):
   datadone = 1
   break
if(it1 in garr):
```

```
            if(t[it1] == '0' ):
                if(g % 2 != 0):
                    g = g - 1
                    img[lb,bb] = (r,g,b)
            elif(t[it1] == '1'):
                if(g % 2 != 1):
                    g = g - 1
                    img[lb,bb] = (r,g,b)
            it1 = it1 + 1
    track = track + 1
  if(it1 >= len(t)):
        it1 , t = 0 , a[it]
        t , it = list(t) , it + 1
        if(track > maxlen * len(msg)):
        datadone = 1
        break
if(it1 in barr):
        if(t[it1] == '0' ):
            if(b % 2 != 0):
                b = b - 1
                img[lb,bb] = (r,g,b)
        elif(t[it1] == '1'):
```

```
                if(b % 2 != 1):
                    b = b - 1
                    img[lb,bb] = (r,g,b)
                it1 , track = it1 + 1 , track + 1
            if(it1 >= len(t)):
                it1 = 0
                t = a[it]
                t , it = list(t) , it + 1
                if(track > maxlen * len(msg)):
                datadone = 1
                break
            bb = bb + 1
        bb , lb = 0 , lb + 1
       if(datadone == 1):
           break
r,g,b = img[0,0]
    lb , bb = 0 , 0
  for i in range(20):
     bb = bb + 1
     Imag.save(file, "png")
def close_window():   # function for closing a window
    val = text.get(0.0,END)
```

```
    obj.destroy()
obj = Tk()       #Creating Window
obj.title("Concealing Images")
photo = PhotoImage(file="download.png")
obj.iconphoto(False, photo)
obj.resizable(0, 0)
canvas = Canvas(obj, width=900, height=780)      #Creating Canvas to draw
canvas.pack()
image=ImageTk.PhotoImage(Image.open("1.jpg"))         #Displaying Background Image
canvas.create_image(0,0,anchor=NW,image=image)
image1=ImageTk.PhotoImage(Image.open(photovar))
firstimg = canvas.create_image(100,50,anchor=NW,image=image1)
text = Text(obj, width=40, height=1, wrap=WORD)
button2 = canvas.create_window(200, 500, anchor='nw', window=text)
text.insert(0.0, "Enter Secret Message")
text1 = Text(obj, width=40, height=1, wrap=WORD)
key = canvas.create_window(200, 590, anchor='nw', window=text1)
dialog = Button(obj, text="OPEN", width=10, height=1, borderwidth=2, command=browse, bg="pink",
fg="black")
d_button = canvas.create_window(560, 500 , anchor='nw', window=dialog)
key_button = Button(obj, text="GET KEY",width=10, height=1, borderwidth=2, command=getpubkey,
bg="pink", fg="black")
```

```
gkey = canvas.create_window(560, 590, anchor='nw', window=key_button)

encode_button = Button(obj, text="ENCODE",width=12, height=1, borderwidth=3, command=encode, bg="pink", fg="black")

encode = canvas.create_window(300, 650, anchor='nw', window=encode_button)

decode_button = Button(obj, text="SEND",width=12, height=1, borderwidth=3, command=close_window, bg="pink", fg="black")

decode = canvas.create_window(480, 650, anchor='nw', window=decode_button)

width , height = 900 , 780

screenw = obj.winfo_screenwidth()

screenh = obj.winfo_screenheight()

x , y = (screenw / 2) - (width / 2) , (screenh / 2) - (height / 2)

obj.geometry("%dx%d+%d+%d" % (width, height, x, y))

obj.mainloop()
```

Reciever.py

```
def decode():

 file=filename

   imageval = Image.open(file,'r')

   hieght , width = Image.open(file,'r').size

   pri_key = text1.get(0.0, END)

   pri_key = int(pri_key)

   dec , lb , bb ="" , 0 , 0

   img = imageval.load()
```

```
 for i in range(20):
    bb = bb + 1
 bb , lb , b1 , datarecovered , tup, rec  = 0 , 0 , “” , 0 , 0 , []
r , g , b = img[lb,bb]
 msg_len,encoded_offset = r+g,b
 bb = bb + 1
 track_val =0
 max_track_val = msg_len * encoded_offset
 iter_val = 0
if(encoded_offset % 3 == 0):
    while(lb <= hieght - 1):
        while(bb <= width - 1 ):
          r , g , b = img[lb,bb]
          if(r % 2 == 0):                b1 = b1 + '0'
          if(r % 2 != 0):                b1 = b1 + '1'
         iter_val , track_val = iter_val + 1 , track_val +1
          if(g % 2 == 0):                b1 = b1 + '0'
          if(g % 2 != 0):                b1 = b1 + '1'
          iter_val , track_val = iter_val + 1 , track_val + 1
          if(b % 2 == 0):                b1 = b1 + '0'
          if(b % 2 != 0):                b1 = b1 + '1'
          iter_val , track_val = iter_val + 1 , track_val + 1
```

```
            if(iter_val == encoded_offset):
              iter_val = 0
              rec.append(b1)
              b1=""
            if( track_val == max_track_val):              datarecovered = 1
              break
            bb = bb + 1
          bb , lb = 0 , lb +1
     if(datarecovered == 1):            break
decoded_string = ""
   for i in rec:
      val = binary_to_decimal(i)
      decoded_string += chr(val)
for i in decoded_string:
      dtt = Decimal(0)
      dtt = pow(ord(i),pri_key)
      dt = dtt % n
      dec = dec + chr(dt)
   text.insert(0.0, dec)


def nearest_roundoff(n):   #function for rounding off to nearest value
  if(n > 0):     return math.ceil(n/3.0) * 3
```

```
  elif( n < 0):      return math.floor(n/3.0) * 3
  else:      return 3
def isPrime(n) :    #function to validate wheather a number is prime or not a prime
   if(n in [2,3]):       return True
   else:       k=0
      while(True):
         if(n in [((6*k)-1),((6*k)+1)]):              return True
         else:
            if(( ((6*k) - 1) or ((6*k) + 1) ) > n):                return False
            k = k + 1
obj = Tk()       #Creating Window
obj.title("Concealing Images")
photo = PhotoImage(file="download.png")
obj.iconphoto(False, photo)
obj.resizable(0, 0)
canvas = Canvas(obj, width=900, height=780)      #Creating Canvas to draw
canvas.pack()
image=ImageTk.PhotoImage(Image.open("1.jpg"))         #Displaying Background Image
canvas.create_image(0,0,anchor=NW,image=image)
image1=ImageTk.PhotoImage(Image.open(photovar))
firstimg = canvas.create_image(100,50,anchor=NW,image=image1)
text = Text(obj, width=40, height=1, wrap=WORD)
```

```
button2 = canvas.create_window(200, 500, anchor='nw', window=text)

text.insert(0.0, "Enter Secret Message")

text1 = Text(obj, width=40, height=1, wrap=WORD)

key = canvas.create_window(200, 590, anchor='nw', window=text1)

dialog = Button(obj, text="OPEN", width=10, height=1, borderwidth=2, command=browse, bg="pink", fg="black")

d_button = canvas.create_window(560, 500 , anchor='nw', window=dialog)

key_button = Button(obj, text="GET KEY",width=10, height=1, borderwidth=2, command=getpubkey, bg="pink", fg="black")

gkey = canvas.create_window(560, 590, anchor='nw', window=key_button)

encode_button = Button(obj, text="ENCODE",width=12, height=1, borderwidth=3, command=encode, bg="pink", fg="black")

encode = canvas.create_window(300, 650, anchor='nw', window=encode_button)

decode_button = Button(obj, text="SEND",width=12, height=1, borderwidth=3, command=close_window, bg="pink", fg="black")

decode = canvas.create_window(480, 650, anchor='nw', window=decode_button)

width , height = 900 , 780

screenw = obj.winfo_screenwidth()

screenh = obj.winfo_screenheight()

x , y = (screenw / 2) - (width / 2) , (screenh / 2) - (height / 2)

obj.geometry("%dx%d+%d+%d" % (width, height, x, y))

obj.mainloop()
```

CHAPTER 5

TESTING

Testing is the process of evaluating a system or its component's with the intent to find that whether it satisfies the specified requirements or not. This activity results in the actual, expected and difference between their results. i.e. testing is executing a system in order to identify any errors or missing requirements in contrary to the actual desire or requirements.

5.1 Testing strategies

In order to make sure that system does not have any errors, the different levels of testing strategies that are applied at different phases of software development is shown in Fig 5.1.

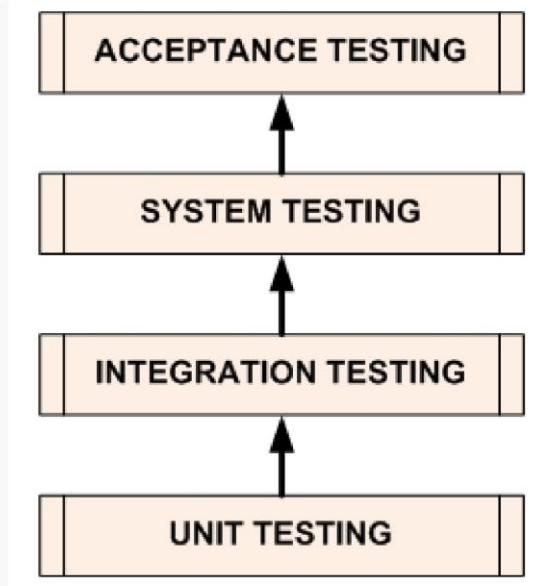

Fig 5.1 Phases of Testing in software development

5.1.1 Unit Testing

The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality. Each module in the tool is tested separately with the help of various users social media accounts.

5.1.2 Integration Testing

The testing of combined parts of an application to determine if they function correctly together is Integration testing. The approach followed here is Bottom-up Integration testing where the smallest modules are tested first and then combined together.

5.1.3 System Testing

This is the next level in the testing and tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards. The complete system is tested with thousands of records from different users and measured accuracy.

5.1.4 Acceptance Testing

The main purpose of this Testing is to find whether application meets the intended specifications and satisfies the client's requirements. Model has been subjected to both alpha and beta testing and gathered their feedbacks.

5.2 Validation

All the levels in the testing (unit, integration, system) are implemented in our application successfully and the results obtained as expected.

5.3 Limitations

Though the lexicon based approach gives reasonable accuracy, machine learning approach will give much better accuracy in some use cases since it learns through the training dataset.

5.4 Test Results

The testing is done among the team members and by the end users. It satisfies the specified requirements and finally we obtained the results as expected.

CHAPTER 6

RESULT ANALYSIS

6.1 Graphical User Interface

From the Graphical User interface part the application is developed using four different modules namely:

6.1.1 Home Screen

6.1.2 Encryption Screen

6.1.3 Decryption Screen

6.1.4 Output Screen

6.1.1 Home screen

The first screen will ask the user to choose which image and choice him/her over encryption or decryption . The entire GUI was created with Python Tkinter. The home screen of our application is shown in Fig 6.1.

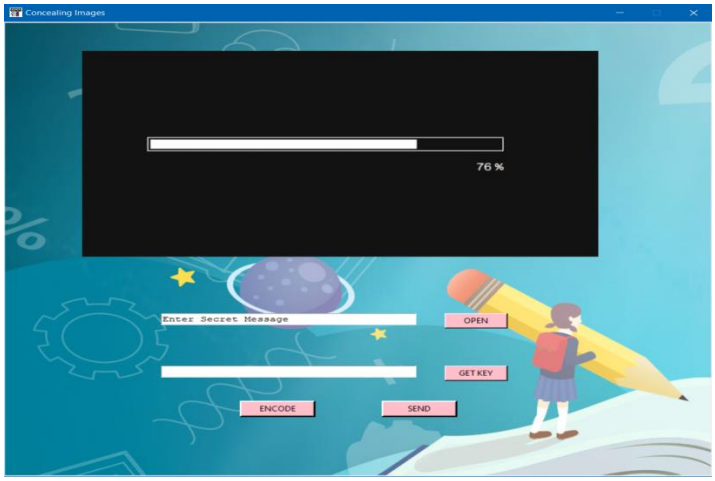

Fig 6.1 Homescreen

6.1.2 Encryption screen

The next screen from the home screen is based on the user's choice in the homescreen. Encryption screen will ask the user to load their image which was downloaded in before for transfer and keys.

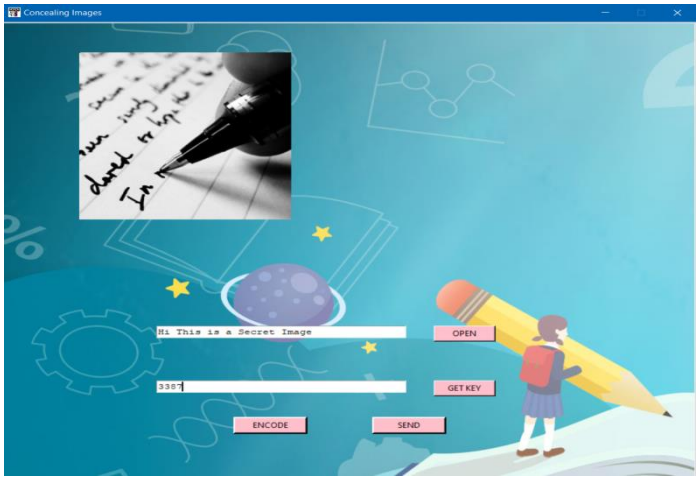

Fig 6.2 Encryption screen

6.1.3 Decryption screen

If the user's choice is twitter, he/she is asked to give the image and find the image which is concealed under the image with his generated private key

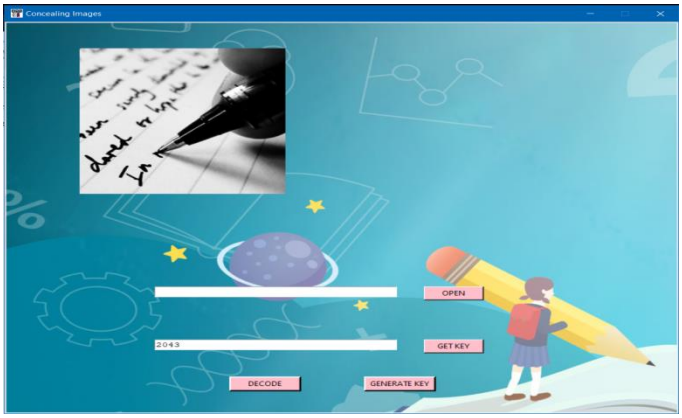

Fig 6.3 Decryption screen

6.2 Output screen

The final output of the analyzed data will be shown in a same Decryption Wimdow. This was done with the help of tkinter library. The Output screen of our application is shown in Fig 6.4.

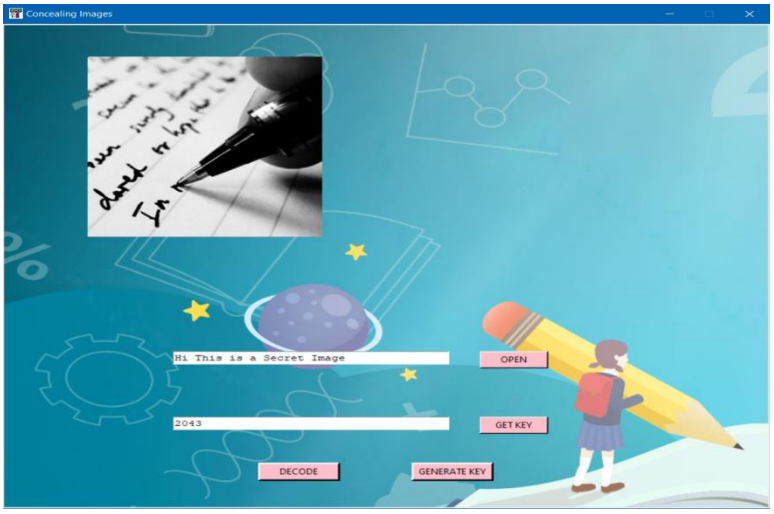

Fig 6.4 Output screen

CHAPTER 7

CONCLUSION AND FUTURE WORKS

7.1 CONCLUSION

The world of Android and Internet has changed the pace at which the world normally used to work and now we can access multiple systems at a click or a touch of a screen. Online transaction for banks, shopping at e-commerce sites and booking travel tickets is now possible from the smartphone itself. Keeping up to speed is the need of the hour in this fast-paced environment for a student, so security is main concern now. So , the private information could be exchanged very easily at the tips.

7.2 FUTURE ENHANCEMENT

The system can be made to generate a unique ID (QR Code) for each user and the authorities can verify the users at the time of use. This unique ID will provide all the details of the user which is secured. By having the identity if any illegal activity is performed using this software it could be avoided. An android application can be built with voice to text converter so that with the help of speech given it is encoded into an image and sent to the receiver.

REFERENCES

[1]. Noda, Hideki & Niimi, Michiharu & Kawaguchi, Eiji. (2006). High-performance JPEG steganography using quantization index modulation in DCT domain. Pattern Recognition Letters.

[2]. Norouzi, Mostafa & Motameni, Homayun. (2007). Labeling Method in Steganography. Proc. World Acad. Sci. Eng. Technol.. 24.

[3]. Wang, Chung-Ming & Wu, Nan-I & Tsai, Chwei-Shyong & Hwang, Min-Shiang. (2008). A high quality steganographic method with pixel-value differencing and modulus function. Journal of Systems and Software.

[4]. Orsdemir, Adem & Altun, Hilmi & Sharma, Gaurav & Bocko, Mark. (2008). Steganalysis-aware steganography: Statistical indistinguishability despite high distortion.

[5]. Al-Taani, Ahmad & Al-Issa, Abdullah. (2008). A new approach for data hiding in gray-level images.

[6]. Chang, C.-C & Wu, W.-C. (2006). Hiding secret data adaptively in vector quantisation index tables. Vision, Image and Signal Processing, IEE Proceedings.

[7]. Wu, Mei-Yi & Ho, Yu-Kun & Lee, Jia-Hong. (2004). An iterative method of palette-based image steganography. Pattern Recognition Letters.

[8]. Dumitrescu, Sorina & Wu, Xiaolin & Wang, Zhe. (2003) . Detection of LSB steganography via sample pair analysis. Signal Processing, IEEE Transactions.

[9]. Ker, Andrew. (2004). Improved Detection of LSB Steganography in Grayscale Images. Lecture Notes in Computer Science: 6th International Workshop on Information Hiding, Toronto, Canada

[10]. Chen, Po-Yueh & Lin, Hung-Ju. (2006). A DWT Based Approach for Image Steganography. International Journal of Applied Science and Engineering.