VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of the UGC Act, 1956)

CRYPTOGRAPHY FUNDAMENTALS
(CSE 1011)

J-COMPONENT REPORT

RSA ENCRYPTION AND IMAGE STEGANOGRAPHY USING LEAST SIGNIFICANT BIT METHOD

Submitted by:
ABHINAV YADAV 18BCI0103
AMRIT KANOI 18BCI0108

Submitted to:
Prof. RAMANI S

Video Demonstration and Presentation: https://vimeo.com/426534166

Abstract:

Steganography and Cryptography are two popular ways of sending vital information in a secret way. One hides the existence of the message and the other distorts the message itself. This project implements the advanced LSB (least significant bit) and RSA algorithm to make the message highly secure. By matching data to an image, there is less chance of an attacker being able to use stego analysis to recover data. Before hiding the data in an image, the application first encrypts it.

Introduction:

The growing use of Internet among public masses and availability of public and private digital data and its sharing has driven industry professionals and researchers to pay a particular attention to information security. Internet users frequently need to store, send, or receive private information and this private information needs to be protected against unauthorized access and attacks. Presently, three main methods of information security being used: watermarking, cryptography and steganography. In watermarking, data are hidden to convey some information about the cover medium such as ownership and copyright. Cryptography techniques are based on rendering the content of a message garbled to unauthorized people. Steganography techniques are based on hiding the existence of information by embedding the secret message in another cover medium. While all three are information security techniques cryptography and steganography are having wide application as watermarking is limited to having information particularly about the cover medium. With the growth of computer network, security of data has become a major concern and thus data hiding technique has attracted people around the globe. Steganography techniques are used to address digital copyrights management, protect information, and conceal secrets.

Proposed System:

The aim of proposed scheme is to make a more secure and robust method of information exchange so that confidential and private data must be protected against attacks and illegal access. To order in achieve the required robustness and security cryptography and steganography is combined. Image is taken as a cover medium for steganography and **RSA algorithm is used for encryption**. In this proposed method our **advanced LSB bit manipulation method is used for embedding the message in the image file** and the message is itself encrypted using the existing RSA encryption method. For embedding the text in image file firstly both the text and image file are converted into binary equivalent and then text is encrypted using RSA. The encrypted text is then embedded into the image file using our advanced LSB algorithm.

RSA Algorithm:

The algorithm was given by three MIT's Rivest, Shamir & Adelman.
RSA algorithm is a message encryption cryptosystem in which two prime numbers are taken initially and then the product of these values is used to create a public and a private key, which is further used in encryption and decryption. The RSA algorithm could be used in combination with advanced LSB in a way that original text is embedded in the cover image in the form of cipher text. By using the RSA algorithm, we are increasing the security to a level above. In case of steganalysis only cipher text could be extracted which is in the encrypted form and is not readable, therefore will be secure.

RSA algorithm procedure can be illustrated in brief as follows:

1. Choose two large prime no. p & q.
2. Calculate N=p*q
3. Calculate f(z)=(p-1)*(q-1)Find a random number e satisfying 1 < e < f (n) and relatively prime to f (n) i.e., gcd (e, f (z)) = 1
4. Calculate a number d such that d = e-1 mod f (n).
5. Encryption: Enter message to get cipher text.
   Ciphertext c= mod ((message. ^e), N).
6. Decryption: The cipher text is decrypted by: Message=mod ((c. ^d), N) [5]

Steganography

Steganography is a very old technique of information hiding. Steganography refers to the science of invisible communication. Unlike cryptography, where the goal is to secure communications from an eavesdropper, steganographic techniques strive to hide the very presence of the message itself from an observer. The term Steganography is forked from the Greek words ―steganos meaning ―cover and ―graphia meaning ―writing defining it as ―covered writing. Before performing steganography, we need three primary accessories which are Secret message, cover medium and one or more embedding algorithm(s) besides these we can also use secret key for better security purpose. In the process of steganography, the cover medium can be a text file, an image, an audio file or it can be a video file but among these most popular is the Image steganography, so here are some advantages of
using images as cover medium in performing steganography.

LSB Technique:

A simple approach for embedding information in cover image is using Least Significant Bits (LSB). The simplest steganography techniques embed the bits of the message directly into least significant bit plane of the cover image in a deterministic sequence. Modulating the least significant bit does not result in human-perceptible difference because the amplitude of the change is small. To hide a secret message inside an image, a proper cover image is needed. Because this method uses bits of each pixel in the image, it is necessary to use a lossless compression format, otherwise the hidden information will get lost in the transformations of a lossy compression algorithm. When using a 24-bit color image, a bit of each of the red, green and blue color components can be used, so a total of 3 bits can be stored in each pixel. For example, the following grid can be considered as 3 pixels of a 24-bit color image, using 9 bytes of memory:

(00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)

When the character A, which binary value equals 10000001, is inserted, the following grid results:

(00100111 11101000 11001000)
(00100110 11001000 11101000)
(11001000 00100111 11101001)

In this case, only three bits needed to be changed to insert the character successfully. On average, only half of the bits in an image
will need to be modified to hide a secret message using the maximal cover size. The result changes that are made to the least
significant bits are too small to be recognized by the human visual system (HVS), so the message is effectively hidden.

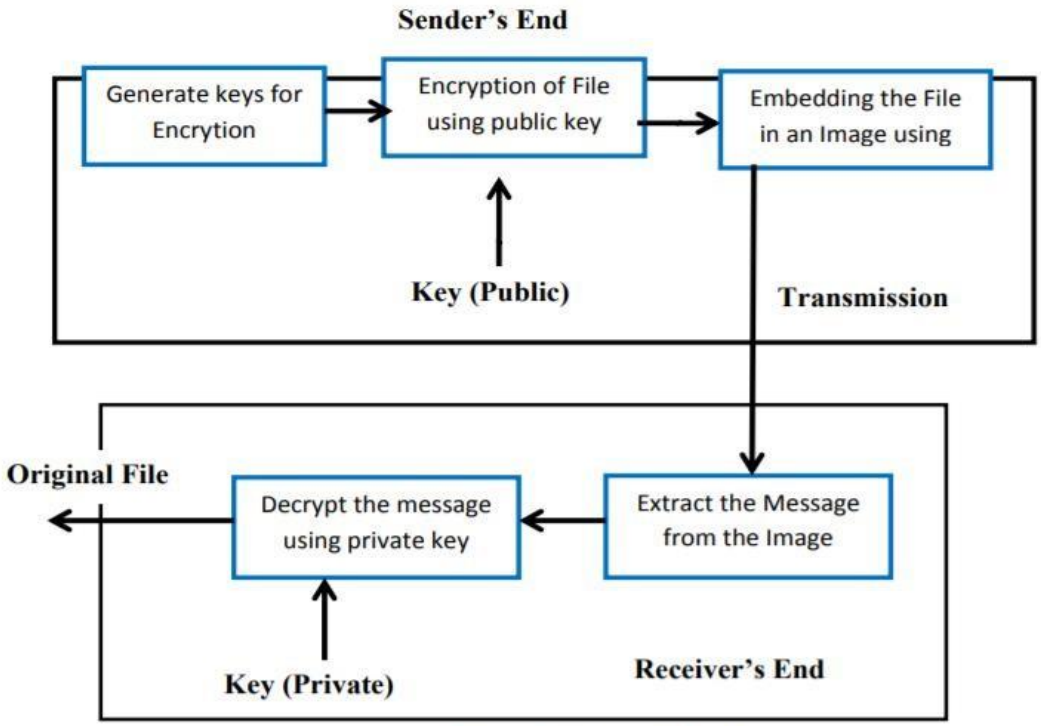


Figure 1: The Proposed Model

CODE IMPLEMENTATION:

RSA_encryption.py

```
import random
from PIL import Image
from Steganography import Encode, Decode
import math

# Encrypting Scheme used

letter = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q",
          "r","s","t","u","v","w","x","y","z",",",".","!","?"," "]

number = ["01","02","03","04","05","06","07","08","09","10","11","12","13",

"14","15","16","17","18","19","20","21","22","23","24","25","26","27",
          "28","29","30","31"]


def cipher(num,e):
    for i in range(len(num)):
        X.append((int(num[i])**e)%n)
```

```
def decipher(num,d):
    for i in range(len(num)):
        Y.append((int(num[i])**d)%n)

def gcd(a, b):
    while b != 0:
        (a, b) = (b, a % b)
    return a

def phi(n):
    amount = 0
    for k in range(1, n + 1):
        if math.gcd(n, k) == 1:
            amount += 1
    return amount

def Decrypt():

    global i,j,Y
    Y=[]

    # Encoded Image
    encoded_image_file = "enc_2.png"


    img2 = Image.open(encoded_image_file)
    print(img2, img2.mode)
    hidden_text = Decode(img2)
    print(hidden_text)

    decipher(hidden_text,d)

    numD=[]


    for i in range(len(Y)):
        for j in range(len(number)):
            if(Y[i]==int(number[j])):
                numD.append(letter[j])


    for i in numD:
        print(i,end="")
    print("\n")
```

```
def Encrypt():
    # encrypts a plaintext message using the current key
    global plaintext, numC, j, X
    X=[]

    plaintext = (input("Enter Plaintext :"))
    plaintext = (plaintext.lower())
    numC = []


    for i in range(len(plaintext)):
        forjinrange(len(letter)):
            if(plaintext[i]==letter[j]):
                numC.append(number[j])



    cipher(numC,e)
    print("Ciphertext:", X)
    print("Number of Ciphertext blocks:", len(X))

    original_image_file = "2.png"

    img = Image.open(original_image_file)

    print(img, img.mode)

    encoded_image_file = "enc_" + original_image_file

    # don't exceed 255 characters in the message
    img_encoded = Encode(img, plaintext, X)

    if img_encoded:
        img_encoded.save(encoded_image_file)
        print("{} saved!".format(encoded_image_file))



# setup()
n = 2537
e = 13
d = 937

print("To redefine n,e, or d, type "n","e",... etc.")
print("To encrypt a message with the current key, type "Encrypt"")
print("To decrypt a message with the current key, type "Decrypt"")
```

```
print("Type quit to exit")
print('\n')
print('\n')

mm = str()

mm = str()
while mm != "quit":
    mm = input("Enter Command: ")

    if mm.lower() == 'encrypt':
        Encrypt()

    elif mm.lower() == 'decrypt':
        Decrypt()

    elif mm.lower() == 'n':
        try:
            print("current n = ", n)
            n1 = int(input(" Enter a value for n:"))
            if  n1<2 :

                print('Invalid input')
            else :
                n=n1
                print('n set to :',n)
        except ValueError:
            print("please enter a number")

    elif mm.lower() == 'help':
        print("To redefine n,e, or d, type "n","e",... etc.")
        print("To encrypt a message with the current key, type "Encrypt"")
        print("To decrypt a message with the current key, type 'Decrypt"')
        print("Type quit to exit")
        print('\n')
        print('\n')

    elif mm.lower() == 'e':
        try:
            print("current e = ", e)
            e1 = int(input(" Enter a value for e :"))
            if  e1<= 2 or gcd(phi(n),e1)!=1 :

                print('Invalid input')
            else :
                e=e1
                print('e set to :', e)
        except ValueError:
```

```
            print('please enter a number')

    elif mm.lower() == 'd':
        try:
            print('current d = ', d)
            d1 = int(input(" Enter a value for d :"))
            if d1 <= 0 and (e*d1)%phi(n)!=1:

                print('Invalid input')
            else :
                d=d1
                print('d set to :',d)
        except ValueError:
            print('please enter a number')
    else:
        if mm != 'quit':
            ii = random.randint(0, 6)
            statements = ["This cannot be done", "Read the directions again",
                          "Didnt say the magic word", "This input is
UNACCEPTABLE!",
                          "Was that even a word???", "Please follow the
directions",
                          "Just type 'help' if you are really that lost"]
            print(statements[ii])
```

Steganography.py

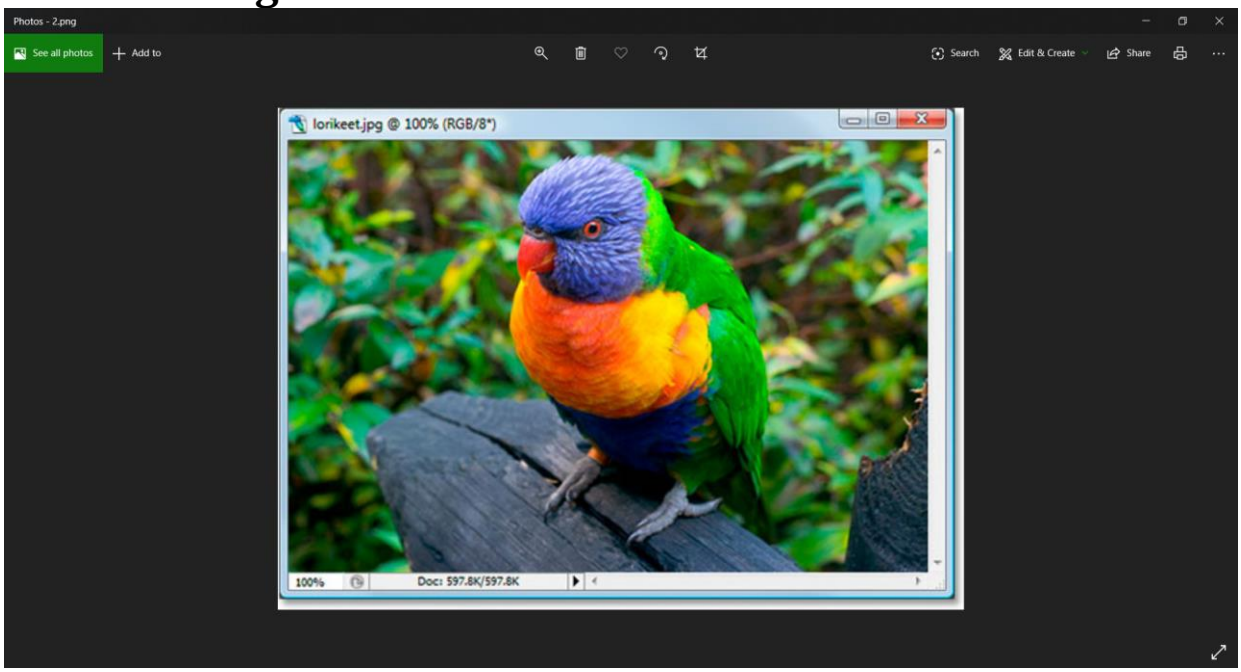
```
from PIL import Image

# Encode function of Steganography

def Encode(img, msg, lst):

    global L
    # List used to store remainder  value during encoding
    L = []

    # Limit length of message to 255
    length = len(msg)

    if length > 255:
        print("Text too long! (don't exeed 255 characters)")
        return False

    # Checking if Image is in RGB color format
```

```
    if img.mode != "RGB":
        print("Image mode needs to be RGB")
        return False

    # Making deep copy of image to make encryption! with original image intact
    encoded_img = img.copy()
    width, height = img.size
    index = 0

    for row in range(height):
        for col in range(width):
            r, g, b = img.getpixel((col, row))

            if row == 0 and col == 0 and index < length:
                asc = length
            elif index <= length:
                c = lst[index - 1]
                asc = c // 255

                # Storing remainder value, used for decoding
                L += [c % 255]
            else:
                asc = r
            encoded_img.putpixel((col, row), (asc, g, b))
            index += 1

    return encoded_img


# Decode function of Steganography

def Decode(img):

    global L
    width, height = img.size
    msg = ""
    lst = []
    index = 0
    length = 0

    for row in range(height):
        for col in range(width):
            try:
                r, g, b = img.getpixel((col, row))
            except ValueError:
                r, g, b, a = img.getpixel((col, row))

            # Runs only once to get length
            if row == 0 and col == 0:
```

```
                length = r

            elif index <= length:

                # L list is used to get the correct value encoded
                lst += [r * 255 + L[index - 1]]

            index += 1

    return lst
```

Input/output Snapshots:

Normal Image:



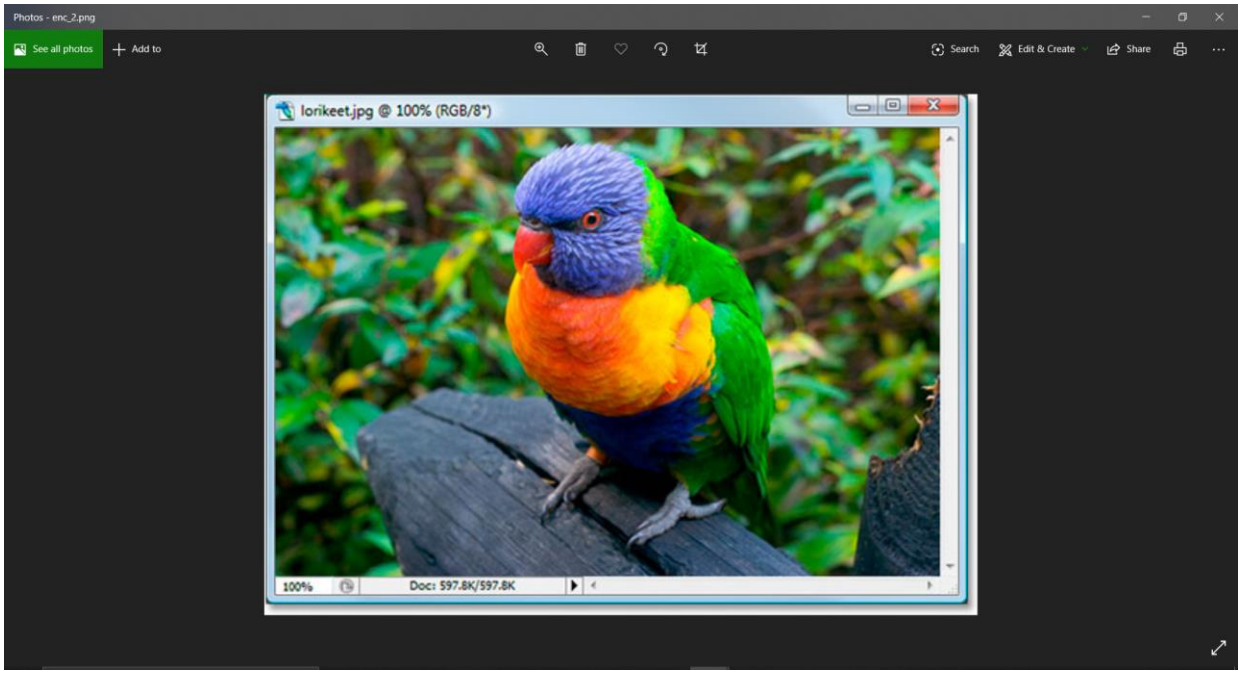
ENCRYPTION:

INPUT: Text Message & Cover Image

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:\Users\abhin\OneDrive\Desktop\RSA.py ==============
To redefine n,e, or d, type 'n','e',... etc.
To encrypt a message with the current key, type 'Encrypt'
To decrypt a message with the current key, type 'Decrypt'
Type quit to exit



Enter Command: Encrypt
Enter Plaintext :ABCDEFGH
Ciphertext: [1, 581, 1087, 140, 205, 2371, 1125, 156]
Number of Ciphertext blocks: 8
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=508x371 at 0x26F20B2C788>
 RGB
enc_2.png saved!
Enter Command:
```

Command: Enter encrypt

OUTPUT: Stego Image Created(with encrypted text behind the image)



DECRYPTION:

```
>>>
============== RESTART: C:\Users\abhin\OneDrive\Desktop\RSA.py ==============
To redefine n,e, or d, type 'n','e',... etc.
To encrypt a message with the current key, type 'Encrypt'
To decrypt a message with the current key, type 'Decrypt'
Type quit to exit



Enter Command: Encrypt
Enter Plaintext :ABCDEFGH
Ciphertext: [1, 581, 1087, 140, 205, 2371, 1125, 156]
Number of Ciphertext blocks: 8
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=508x371 at 0x26F20B2C788>
 RGB
enc_2.png saved!
Enter Command: Decrypt
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=508x371 at 0x26F20B2C788> R
GB
[1, 581, 1087, 140, 205, 2371, 1125, 156]
abcdefgh

Enter Command:
```

Command: Enter decrypt

Conclusion:

To hide confidential information cryptography and steganography can be effectively used. The objective of any Steganographic method is to hide maximum secret information which is immune to external attacks and also should not convey the fact that the cover medium is carry secret information. This project has used LSB substitution as a Steganographic method. Along with RSA algorithm the over system becomes more secure. This project helped us understand how data can be passed secretly through an image and how image processing and cryptographic algorithms can be applied for achieving this. Steganography is in the nascent stage of development. Several new techniques are being discovered and implemented. It is analysed that time is not far away when its importance would be realized by organizations in general and armed forces in particular.

……..XXXXXXX……

Video Demonstration and Presentation Link:

https://vimeo.com/426534166