Enhancing Security of Image and Data Using AES, LSB steganography for Secure Transfer

A PROJECT REPORT

Submitted by

Laya Rathod (18BCE2162)

Course Code: **CSE3501**
Course Title: **Information security Analysis and Audit**

Under the guidance of

Prof. Kakelli Anil Kumar
School of Computer Science and Engineering
VIT University, Vellore.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

INDEX

Section **Page no**

ABSTRACT

Quite often, we communicate exceedingly confidential information over a communication network that we want only the recipient to know. Techniques like steganography, watermarking and cryptography can be used to keep our data confidential. But, a combination of these techniques would help us keep our data highly confidential. In this project, we address the problem of security breaches using the core algorithms in the said fields. We know that email service providers sell our data to third party companies for their usage and profits. In today's scenario, security is a major concern while transmitting any information over the network. Security provided by the network is insufficient with the increasing rates of cybercrimes. Therefore, we need employ other techniques to carefully send our data over the network. In this project, we explore a number of security techniques that can be combined together to ensure higher levels of security to our data.

Keywords: LSB Steganography, Image Segmentation, AES Encryption, Watermarking.

1. INTRODUCTION

In today’s tech savvy world security of data is of foremost importance. Security has become one of the most important factors in information and communication technology. Online stores, social media sites, entertainment and miscellaneous sites, hospital portals, bank portals and so on store sensitive and private information of people on a daily basis. It is important that the most secure algorithms be used to store their data.

In this project, we address the problem of enhancing security of a secret image and secret message that is to be sent over a network, by digitally processing it. We require that the secret image and secret message to be sent to the recipient in such a way that no one else suspects the existence of them. A cover image is used as a decoy in this technique in which the secret image as well as the secret message is embedded. On the sender’s side, the secret image is encrypted using AES Algorithm. In this encrypted secret image, the secret message is hidden using LSB Based Image Steganography. Furthermore, the encrypted secret image with the secret text is hidden in the cover image, using LSB Based Image Steganography. The stego image thus obtained is split into 16 parts, indexed and sent to the receiver. On the receiver side, these sub images are fetched one by one and merged based on their index. The encrypted image is obtained from the merged image. Next, we extract the secret text from the LSBs of this encrypted image. Additionally, decryption is performed to extract the original secret image from the encrypted secret image. Thus, the receiver obtains the secret image and the secret message from the cover image.

The reason why image steganography is preferred over other kinds of encryption techniques such as cryptography is because the disguised message is very easily recognizable. Whereas in Steganography the message is disguised by changing the original message in such a way that it becomes illegible. This project focuses on steganography using LSB (Least Significant Bit) algorithm in which the original message generated is previously encrypted using 256-bit AES (Advanced Encryption Standard algorithm) and it can also restore the data that was hidden previously. The best part about steganography is that the hidden information can just go unnoticed by the external world, unlike cryptography. In cryptography others at least get to know that there is some hidden data and in steganography no suspicion is generated at all. In this project, the technique of digital watermarking is also used to make sure that the sender has sent the image and no one else has done so ensuring the non-repudiation feature.

2. BACKGROUND OF THE WORK

2.1 EXISTING SYSTEM

In the current encryption systems, individual algorithms are used to secure data. Such as Facebook server use advanced hashing algorithm while some others use maybe AES or DES algorithms to encrypt their passwords. But each of these mentioned algorithms have been cracked some or the other time, which means they are not invincible and can be broken by a skilled hand.

The security of the data (passwords in many cases) is highly and threateningly compromised. All these algorithms are very famous all around the globe and are used by many, some are even open source. This means that the algorithm’s flaws are well known to all and in some cases, even the source code is well known to many.

There is no work with provides the feature of nonrepudiation as of now along with the said technique. We will work on providing all the basic features required.

2.2 PROPOSED SYSTEM

The methodology used is that we have introduced a new secret data communication system that employs the usage of state-of-the art cryptographic algorithm AES with symmetric key together with steganography. The joining of these techniques builds a robust steganography-based communication system capable of withstanding multiple types of attacks. Our system was designed in a way that offers a solution to the major flaws presented in other stenographic communication systems. The Least Significant Bit (LSB) is one of the frequently used techniques in spatial domain image steganography. At the receiver side the message is extracted from the 6 image and is then decoded using various encryption methods to get the original message.

Matlab as a simulator is being used to implement the techniques of encryption and steganography. Matlab provide highly computing environment and advanced in-built function for image processing. The methodology used is focused on image-based steganography. We also have used the technique of digital watermarking to make sure that the sender has sent the image and no one else has done so ensuring the non-repudiation feature.

2.3 RELATED WORK LITERATURE SURVEY

Septimiu Fabian Mare et al, introduced a new communication system that uses a unique combination of two state cryptographic algorithms i.e. asymmetric RSA and AES with symmetric key cryptography with steganography. The combination of these techniques generates a strong communication system resistant to multiple types of attacks, detection and reverse engineering processes. Our model is built in such a way that it offers a solution to the majority of defects in other stenographic communication systems. [1]

DNA is a medium for communication system for data security since it is a strong security method that achieves maximum protection with low modification rate and high capacity. A new security method can be built by taking the all benefits of DNA based steganography and AES cryptography. This method will provide a double layer security to the secret message. In this technique, the message is encrypted to DNA bases and then AES algorithm is applied on it. This unique methodology provides multi-layer data security to the message. [2]

Utsav Sheth et al., describe a unique technique in which data is hide in a cover image. The lower nibble of each cover image byte is changed 7 so that each nibble contains an input text. The steganography technique used in this implementation increases the data capacity of the cover image and also ensures high level protection. The Java libraries are used for easy implementation. A simple Graphic user interface has been developed using Java's applets and SWING packages. The AES cryptographic technique is further used for improving the data security of the model. [3]

This paper introduces a special technique in which the alice encodes the message using asymmetric cryptographic algorithm. This technique provides double layer protection by encoding the text using multiple algorithms such as Modified Vigenere Cipher algorithm and data is hidden in a cover image using Least Significant Bit steganography. The Least Significant Bit (LSB) is the most commonly implemented algorithms in spatial domain image steganography. At the bob side the secret data message is extracted from the cover image and is then decrypted using different decoding algorithms to extract the original message back.In this paper, G.Prashanti is using Matlab as a simulator to implement the techniques of encryption and steganography. Matlab provide highly computing environment and advanced in built function for image processing. [4]

The fast development of information technology in present time needs a strong data protection algorithm for exchanging of any kind of image or secret information. Steganography is a longtime methodology for hiding information from an unauthorized access thus providing confidentiality. Steganography is a technique that hides information in different file formats: audio, text, videos and images. Also, the given methodology is not only secure, but computationally efficient and fast [5].

In this paper Prof. Beenish Mehboob, compares the various different cryptographic techniques. The main goal of steganography is to 8 hide the data content during the transmission process. The success of stenographic technique depends on the confidentiality and data integrity. The information security also relies on the robustness of the implemented technique. The most commonly used is the Least Significant Bit algorithm for image steganography. Saving images in these particular formats provides lossless compression during transmission. This paper gives a complete overview on the art of Steganography and gives a unique technique to hide data in a RGB color image [6].

This paper proposes a new methodology of using AES algorithm, to improve the data protection of the hidden information in the two proposed techniques for steganography i.e. the genetic algorithm and path relinking. It also combines the two techniques forming a new hybrid approach that provides the benefits of both the algorithms. Also, all types of digital information from text and compressed files and even the executable programs can be hidden inside the cover image. This increases the scope of various application of the technique for exchanging information across different networks and hiding the data from attackers [7]

Dipti Kapoor Sarmah et al., developed a system where they build a new technique in which Cryptography and Steganography are combined together and implement it as an integrated part along with newly build enhanced security system. In steganography we are using AES algorithm to encode a secret data and a part of the data is hidden in DCT of an image and the rest of the message is used to generate two secret keys which provides multilayer security to this model. [8]

The proposed method is summed up by embedding data files into digital media with steganalysis done by Reddy, V. L., 9 Subramanyam et al. [9]

G. G. Rajput et al., focus on image based steganography for hiding data in this paper. He uses LSB technique for embedding text information i.e. secret image in digital RGB color images is proposed. Secret text is encrypted using Least Significant Bits (LSB) of three components of color image namely, red, green and blue (RGB) channels using the angular transformation concept. [10]

Table 2.1

Author	Methodology	Solution	Advantages	Disadvantage
Septimiu Fabian Mare et al [1]	RSA with asymmetric keys and AES with symmetric key	Provided high level security for data and images using the method of steganography	Robust steganography-based communication system capable of withstanding multiple types of attacks	High data size after steganography, takes high time
K S Sajisha et al. [2]	DNA based AES cryptography and DNA steganography	Gave multiple level security for data using the method of cryptography and steganography	Provide multilayer security to the secret message.	High data size after steganography , takes high time, DNA hard to implement.
Utsav Sheth et al. [3]	AES encryption algorithm.	Used cryptography for high level security	Maximize on data capacity and also ensures Security	No multilayer security
G.Prashanti and B.V.Jyothirmai,	Data Confidentiality Using	Provided high level confidentiality	It is simple to understand, easy to	It is not very secure as

K.Sai Chandana [4]	Steganography and Cryptographic Techniques (Modified Vigenere Cipher)	using the cryptography method	implement. The biggest advantage is the relative frequencies of individual letters exhibit a much greater range than of diagrams, making frequency analysis difficult.	compared to AES
Ammad Ul Islam, Faiza Khalid, Mohsin Shah, Zakir Khan [5]	An Improved Image Steganography Technique based on MSB using Bit Difference	Provided high level security for data and images using the method of steganography	Provides better security than LSB steganography algorithm. Increased security with reduced distortion rate.	Less secure as just MSB is used, no multilevel security
Beenish Mehboob and Rashid Aziz Faruqui [6]	A Steganography Implementation	Used Steganography for high level security	Better image quality and security. Error detection and noise free transmission	For better image quality the file size increases drastically
Aura Conci', Andre Luiz Brazil', Simone	AES Cryptography in Color Image	Provided high level security for data and	More robust as algorithm is integrated with	Data size after steganography

Bacellar Leal Ferreira [7]	Steganography by Genetic Algorithms	images using the method of steganography	AES. Embedding capacity is more.	is high, takes high time.
Dipti Kapoor Sarmah et al. [8]	AES algorithm to encrypt a message and a part of the message is hidden in DCT of an image	DCT used to hide data and AES for providing high level security for encryption of messages	Enhanced security, integration	Integration is very hard, high data size after steganography takes high time
Reddy, V.L., Subramanyam, A., & Reddy, P.C. [9]	Embedding digital media with steganalysis	Embedded high level security for MEDIA using the method of steganography	Robust, secure	Hard to embedded media
Rajput G. G., & Chavan, R. [10]	Secret text is encoded in LSB of three components of color image namely. RGB channels using the angular transformation concept	Provided high level security for RGB images using the method of steganography	Better image quality and security	RGB usage takes a lot of computation time compared to gray scale, High data size after steganography

3. OVERVIEW OF THE WORK

3.1 PROBLEM DESCRIPTION

Based on the background mentioned before, the problem statement pertaining to this project is how the LSB method can be used as one of the stenographic methods combined with AES cryptographic methods and Digital Watermark to conceal messages into a digital image and add non-repudiation feature. This system uses the Advanced Encryption Standard (256-bit) encryption to initially encrypt the secret message and input image. Followed by image steganography for even more security of data while transmitting in networks. Further DCT-DWT Watermarking techniques have been discussed for image security enhancement.

3.2 PROPOSED DESIGN

The methodology used is that we have introduced a new secret data communication system that employs the usage of state-of-the art cryptographic algorithm AES with symmetric key together with steganography. The joining of these techniques builds a robust steganography-based communication system capable of withstanding multiple types of attacks. Our system was designed in a way that offers a solution to the major flaws presented in other stenographic communication systems. The Least Significant Bit (LSB) is one of the frequently used techniques in spatial domain image steganography. At the receiver side the message is extracted from the image and is then decoded using various encryption methods to get the original message. I am using Matlab as a simulator to implement the techniques of encryption and steganography. Matlab provides highly computing environment and advanced in-built function for image processing. Further I have also implemented a GUI based python application for an AES + LSB based secret message transfer system. The necessary libraries and modules used have been mentioned later in this report. The methodology used in this project is focused on image-based steganography. The techniques of digital watermarking to make sure that the sender has sent the image and no one else has done so ensuring the non-repudiation feature.

This project provides all basic necessities for a secure system:

1. **Confidentiality:** Confidentiality refers to personal information shared with other individual that generally cannot be divulged to third parties without the express consent of the client. The secret message is encrypted using AES algorithm before transmission, so confidentiality is ensured.
2. **Data Integrity:** Data security refers to the protection of data against unauthorized access or corruption and is necessary to ensure data integrity. The secret message as well as the secret image are encrypted using cryptographic algorithm AES
3. **Non-Repudiation:** Nonrepudiation is the assurance that someone cannot deny something. Typically, non-repudiation refers to the ability to ensure that a party to a contract or a communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated. In our project, we are adding a digital watermark to ensure non repudiation.
4. **Embedding Effectiveness:** Probability to embed secret messages successfully in an arbitrary cover.
5. **Privacy:** Privacy assures that personal information are collected, processed, protected and destroyed legally and fairly.
6. **Robustness:** Ability to detect the secret message after the processing operations
7. **Security:** Secure the cover message and statistically undetectable scheme.

3.3 DESIGN DESCRIPTION

From the Sender Side:

1. The private key, sender’s message and decryption of the receiver’s message are given as options to the user.
2. The secret message of the sender is taken as input and encrypted using AES algorithm to provide confidentiality.
3. An image is taken as an input from the user in which the user wants to embed the secret message
4. The image is also encrypted using AES algorithm.
5. The encrypted and digitally signed secret message is then embedded in the encrypted image using Least Significant Bit algorithm in which the LSB of the pixels of the image are replaced with the secret code.

6. The encoded image is then again embedded into another cover image using embedding algorithm.
7. The resulted stego image is then divided into multiple parts (16 to be specific).
8. Each part is then separately sent to the recipient.

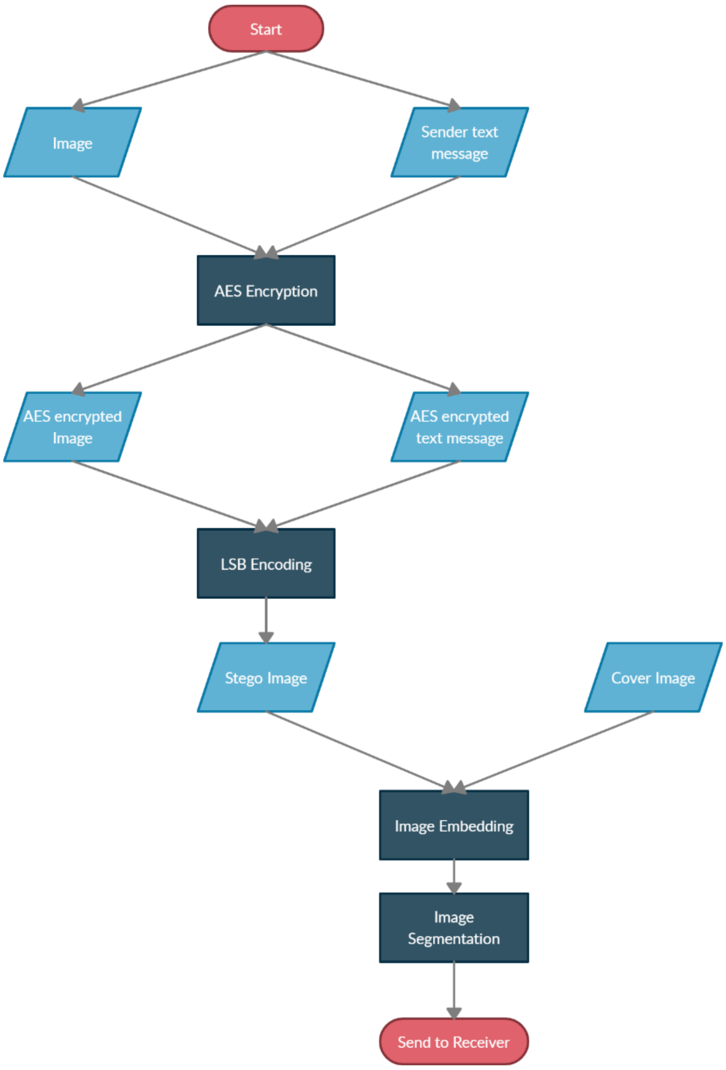


Fig 3.1 Sender Side Process

At the Receiver Side:

1. On the receiver’s end, the sub images are stitched back based on their index values (0 to 15) to regain the original cover image.
2. The embedded image is now extracted to get back the original stego image further the encrypted image with the encrypted text from the stego cover image, we extract the LSB of all red, green and blue components.

3. The encrypted image is then decrypted using the AES decryption algorithm.
4. Now, to extract the encrypted text from the image, we again extract the LSB bits of the red, blue and green components.
5. The encrypted text is then decrypted using AES algorithm and hence the receiver can view the secret code.

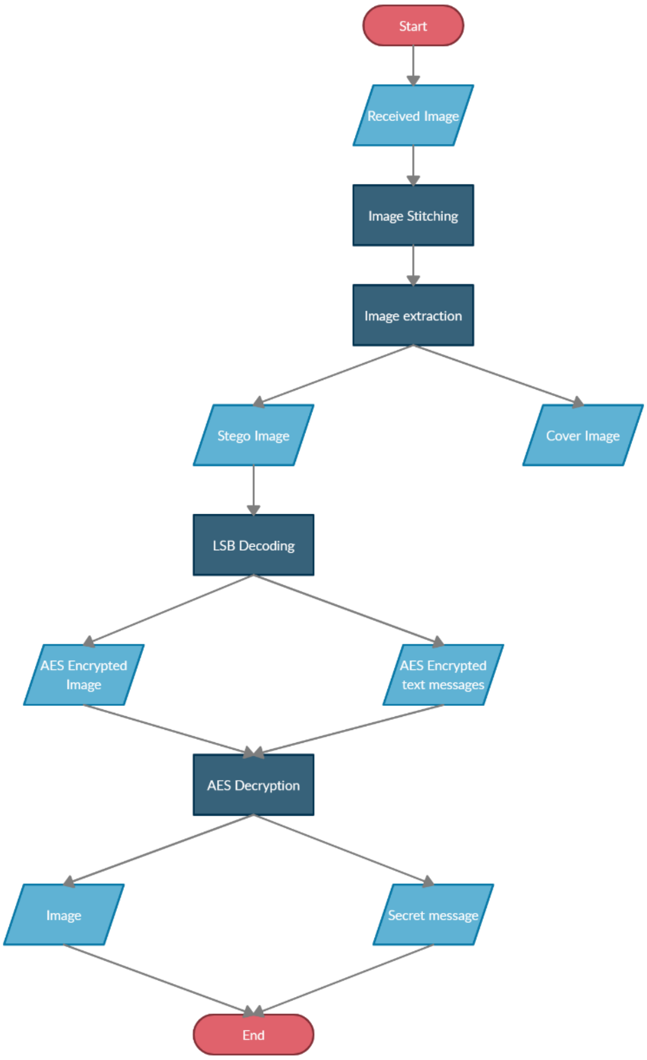


Fig 3.2 Receiver Side Process

4. IMPLEMENTATION

4.1 MODULES AND TOOLS USED

The tools used for this project:

1. MATLAB R2015a
2. Creatly Tool
3. Python 3

Python Libraries used:

- Python pip
- numpy==1.17.1
- opencv-python==4.1.1.26
- Pillow==6.2.0
- pycryptodome==3.9.0
- cv2
- tkinter

4.2 SOURCE CODE

Matlab Code for LSB Steganography:

```
%Encoding the message

original=imread('dog.jpg'); //upload required file

cover=rgb2gray(original);

[row,column]=size(cover);

L=256;

stego=cover;

message=input('Enter the message to be hidden: ','s');

len=strlength(message)*8;  %Each character will take 8 bits so total number of bits in the message will be len
```

```
ascii=uint8(message);   %ascii is a vector having the ascii value of each character

binary_separate=dec2bin(ascii,8);   %binary_separate is an array having the decimal
representation of each ascii value

binary_all=";  %binary_all will have the entire sequence of bits of the message

for i=1:strlength(message)

    binary_all=append(binary_all,binary_separate(i,:));

end

count=1;    %initializing count with 1

for i=1:row

    for j=1:column

        %for every character in the message

        if count<=len

            %Obtain the LSB of the grey level of the pixel

            LSB=mod(cover(i,j),2);

            %Convert the bit from the message to numeric form

            a=str2double(binary_all(count));

%Perform XOR operation between the bit and the LSB

temp=double(xor(LSB,a));

%Change the bit of the stego image accordingly

stego(i,j)=cover(i,j)+temp;

count=count+1;

end

end
```

```
end
subplot(1,2,1);
imshow(cover);
title('Cover Image');
subplot(1,2,2);
imshow(stego);
title('Stego Image');
%Decoding the message
count=1;
message_in_bits=";
for i=1:row
for j=1:column
%For all the characters in the message
if count<=len
%Retrieve the LSB of the intensity level of the pixel
LSB=mod(stego(i,j),2);
%Append into message_in_bits to get bit sequence of message
message_in_bits=append(message_in_bits,num2str(LSB));
count=count+1;
end
end
end
```

```
%Converting the bit sequence into the original message
i=1;
original_message=";
while i<=len
%Take a set of 8 bits at a time
%Convert the set of bits to a decimal number
%Convert the decimal number which is the ascii value to its corresponding character
%Append the obtained character into the resultant string
original_message=append(original_message,char(bin2dec(message_in_bits(1,i:i+7))));
i=i+8;
end
disp(['The original message is: ',original_message]);
```

Python Codes for AES + LSB Implementation:

aes.py:

```
from Crypto.Cipher import AES
# AESCipher used to do text manipulation/cryptography
# key length : 16 character
# message length : multiple of 16
class AESCipher:
  def __init__(self, key):
    self.key = str.encode(key)
  # encrypt encript message in msg using key
  # and return ciphertext result
  def encrypt(self, msg):
    cipher = AES.new(self.key, AES.MODE_ECB)
    cipherText = cipher.encrypt(str.encode(msg))
    return cipherText.hex()
  # decrypt try decrypt cipher text in cipherText using key
  # and return secret message as result
  def decrypt(self, cipherText):
    decipher = AES.new(self.key, AES.MODE_ECB)
    msg = decipher.decrypt(bytes.fromhex(cipherText))
    return msg
if __name__ == "__main__":
  c = AESCipher("abcdefghijklmnop")
  secret = "SepertiYangBiasa"
  print(secret)
  cipherText = c.encrypt(secret)
  print(cipherText)
  secret = c.decrypt(cipherText)
  print(secret)
```

lsb.py:

```
import cv2

class AppError(BaseException):

  pass

def i2bin(i, l):

  actual = bin(i)[2:]

  if len(actual) > l:

    raise AppError("bit size is larger than expected.")

  while len(actual) < l:

    actual = "0"+actual

  return actual

def char2bin(c):

  return i2bin(ord(c), 8)

# LSB used to do image manipulation especially embedding secret message

# using LSB method

class LSB():

  # before embedding secret message on image, we need

  # to know which cell is used or will be used to store

  # secret message, to achive that, we will use 16 first cell

  # to store length, this value will be converted to binary

  # and no more that 16 bit which means max length of message is

  # 2^16 = 65536

  MAX_BIT_LENGTH = 16
```

```
def __init__(self, img):
  self.size_x, self.size_y, self.size_channel = img.shape
  self.image = img
  # pointer used to refer which cell on image will be read or write
  self.cur_x = 0
  self.cur_y = 0
  self.cur_channel = 0
# move pointer to next cell
def next(self):
  if self.cur_channel != self.size_channel-1:
    self.cur_channel += 1
  else:
    self.cur_channel = 0
    if self.cur_y != self.size_y-1:
      self.cur_y += 1
    else:
      self.cur_y = 0
      if self.cur_x != self.size_x-1:
        self.cur_x += 1
      else:
        raise AppError("need larger image")
```

```
# replace last bit from value of cell referred by pointer
# and move pointer to next cell
def put_bit(self, bit):
  v = self.image[self.cur_x, self.cur_y][self.cur_channel]
  binaryV = bin(v)[2:]
  # replace last bit if different
  if binaryV[-1] != bit:
    binaryV = binaryV[:-1]+bit
  self.image[self.cur_x, self.cur_y][self.cur_channel] = int(binaryV,2)
  self.next()
# put_bits put array of bit to designated cell respectively
def put_bits(self, bits):
  for bit in bits:
    self.put_bit(bit)
# read_bit read last bit from value of cell referred by pointer
# return bit as result
def read_bit(self):
  v = self.image[self.cur_x, self.cur_y][self.cur_channel]
  return bin(v)[-1]
# read_bits read last bit for every cell referred by pointer until length
# return array of bit as result
def read_bits(self, length):
  bits = ""
  for _ in range(0, length):
    bits += self.read_bit()
    self.next()
  return bits
# embed embed text to image
def embed(self, text):
  # calculate text length and convert it to binary with length 16 bit
  text_length = i2bin(len(text), self.MAX_BIT_LENGTH)
  # put length to first 16 cell
  self.put_bits(text_length)
  # put every character on text to image
  for c in text:
    # convert character into binary with 8 length
    bits = char2bin(c)
    # put every bit to cell respectively
    self.put_bits(bits)
# extract extract text from image
def extract(self):
  # read 16 first cell as length of text that contained on image
  length = int(self.read_bits(self.MAX_BIT_LENGTH), 2)
  text = ""
  for _ in range(0, length):
    # read every 8 bit as a character
```

```
      c = int(self.read_bits(8), 2)
      # convert binary as a character
      text += chr(c)
    return text
  # save save image to dstPath
  def save(self, dstPath):
    cv2.imwrite(dstPath, self.image)
if __name__ == "__main__":
  # obj = LSB(cv2.imread('src.jpg'))
  # obj.embed("ku yakin pasti suatu saat semua kan terjadi, kau kan mencintaiku dan tak akan
 pernah melepasku aku mau mendampingi dirimu, aku mau cintai kekuranganmu, s'lalu berse
dia bahagiakanmu apapun terjadi, kujanjikan aku ada...")
  obj = LSB(cv2.imread('dst.png'))
  text = obj.extract()
  print(text)
```

app.py:

```
import cv2
import tkinter as tk
import numpy as np
from tkinter.filedialog import askopenfilename, asksaveasfilename
from tkinter import messagebox
from PIL import Image, ImageTk
from lsb import LSB

from aes import AESCipher

# Activity handle all user interaction like:

# 1. preview image

# 2. handle button click interaction

# 3. program lifecycle from start and exit

# 4. how User Interface looks like

class Activity:

  # root window object

  master = tk.Tk()

  # store image on cv2 object to be able to image manipulation

  image = None

  # store image on Imagetk object to be able to preview on window

  imgPanel = None

  keyInput = None

  messageInput = None

  path = "./dst.png"

  def __init__(self):

    self.master.title('AES + Steganography')

    # use blank image when program started

    self.image = np.zeros(shape=[100, 100, 3], dtype=np.uint8)

    self.updateImage()
```

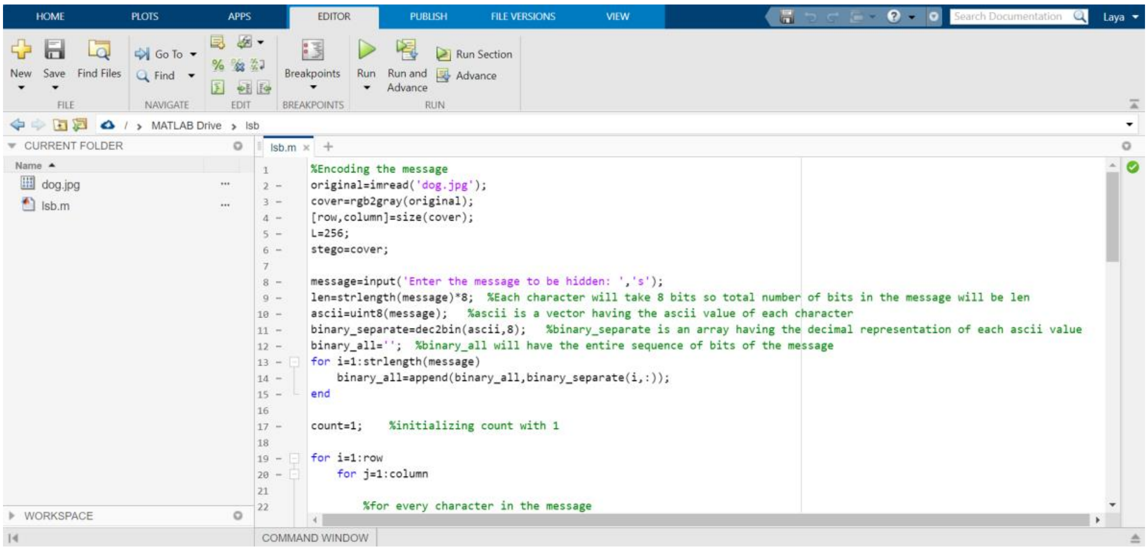
```
# configure open button

openBtn = tk.Button(self.master, text = 'Open', command = self.openImage)

openBtn.pack()

btnFrame = tk.Frame(self.master)

btnFrame.pack()

# configure encode button

encodeBtn = tk.Button(btnFrame, text = 'Encode', command = self.encode)

encodeBtn.pack(side = tk.LEFT)

# configure decode button

decodeBtn = tk.Button(btnFrame, text = 'Decode', command = self.decode)

decodeBtn.pack(side = tk.LEFT)

savebtnFrame = tk.Frame(self.master)

savebtnFrame.pack()

# configure save button

saveBtn = tk.Button(savebtnFrame, text = 'Save Image', command = self.saveImage)

saveBtn.pack(side = tk.LEFT)

# configure save value button

saveValueBtn = tk.Button(savebtnFrame, text = 'Save Value', command = self.saveValue)

saveValueBtn.pack(side = tk.LEFT)

# configure input box for key

tk.Label(self.master, text='Key').pack()

self.keyInput = tk.Entry(self.master)

self.keyInput.pack()
```

```
  # configure input box for secret message
  tk.Label(self.master, text='Secret Message').pack()
  self.messageInput = tk.Text(self.master, height=10, width=60)
  self.messageInput.pack()
# updateImage read image from cv2 object and preview on image window
def updateImage(self):
  image = cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB)
  image = Image.fromarray(image)
  image = ImageTk.PhotoImage(image)
  if self.imgPanel == None:
    self.imgPanel = tk.Label(image=image)
    self.imgPanel.image = image
    self.imgPanel.pack(side="top", padx=10, pady=10)
  else:
    self.imgPanel.configure(image = image)
    self.imgPanel.image = image
# cipher create AESCipher object to encode message with inputed key as secret key
def cipher(self):
  key = self.keyInput.get()
  # key length must 16 character
  if len(key) != 16:
    messagebox.showwarning("Warning","Key must be 16 character")
    return
  return AESCipher(self.keyInput.get())
# encode encode message using AESCipher and embed cipher text to image
def encode(self):
  message = self.messageInput.get("1.0",'end-1c')
  # message length will forced to be multiple of 16 by adding extra white space
  # at the end
  if len(message)%16 != 0:
    message += (" " * (16-len(message)%16))
  cipher = self.cipher()
  if cipher == None:
    return
  cipherText = cipher.encrypt(message)
  obj = LSB(self.image)
  obj.embed(cipherText)
  self.messageInput.delete(1.0, tk.END)
  self.image = obj.image
  # preview image after cipher text is embedded
  self.updateImage()
  messagebox.showinfo("Info", "Encoded")
# decode extract cipher text from image and try decode it using provided secret key
def decode(self):
  cipher = self.cipher()
  if cipher == None:
    return
  obj = LSB(self.image)
  cipherText = obj.extract()
  msg = cipher.decrypt(cipherText)
  # show decoded secret message to message input box
  self.messageInput.delete(1.0, tk.END)
  self.messageInput.insert(tk.INSERT, msg)
# openImage ask user to select image
def openImage(self):
  path = askopenfilename()
  if not isinstance(path, str):
    return
  self.image = cv2.imread(path)
  self.updateImage()
# saveValue export int value for every color channel (RGB)
# on csv format
def saveValue(self):
  path = asksaveasfilename(title = "Select file")
  if path == '':
    return
  np.savetxt(path+'_blue.csv', self.image[:, :, 0], delimiter=',', fmt='%d')
  np.savetxt(path+'_green.csv', self.image[:, :, 1], delimiter=',', fmt='%d')
```

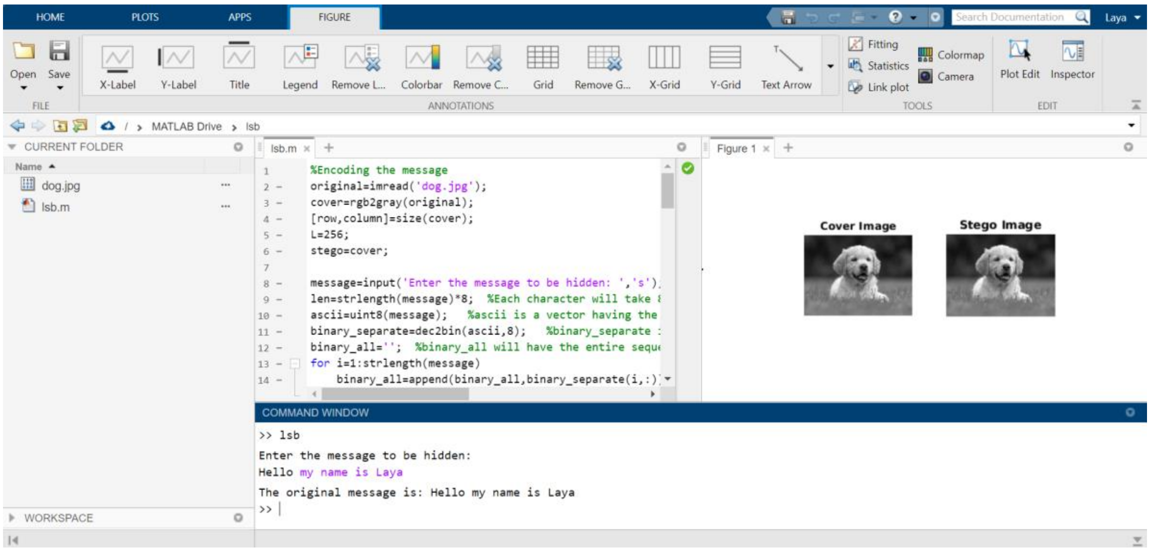
```
    np.savetxt(path+'_red.csv', self.image[:, :, 2], delimiter=',', fmt='%d')
    messagebox.showinfo("Info", "Saved")
  # saveImage save image on png format
  def saveImage(self):
    path = asksaveasfilename(title = "Select file",filetypes=[("png files", "*.png")])
    if path == '':
      return
    if ".png" not in path:
      path = path + ".png"
    obj = LSB(self.image)
    obj.save(path)
    messagebox.showinfo("Info", "Saved")
  def startLoop(self):
    self.master.mainloop()
if __name__ == "__main__":
  app = Activity()
  app.startLoop()
```

4.3 EXECUTION SNAPSHOTS

LSB Matlab:

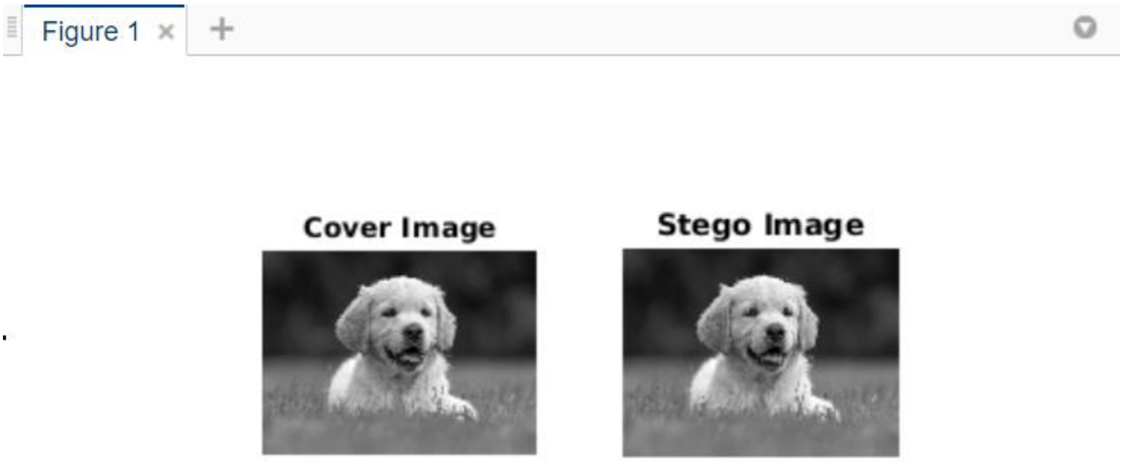


```
%Encoding the message
original=imread('dog.jpg');
cover=rgb2gray(original);
[row,column]=size(cover);
L=256;
stego=cover;

message=input('Enter the message to be hidden: ','s');
len=strlength(message)*8;  %Each character will take 8 bits so total number of bits in the message will be len
ascii=uint8(message);   %ascii is a vector having the ascii value of each character
binary_separate=dec2bin(ascii,8);   %binary_separate is an array having the decimal representation of each ascii value
binary_all='';  %binary_all will have the entire sequence of bits of the message
for i=1:strlength(message)
    binary_all=append(binary_all,binary_separate(i,:));
end

count=1;    %initializing count with 1

for i=1:row
    for j=1:column

        %for every character in the message
```

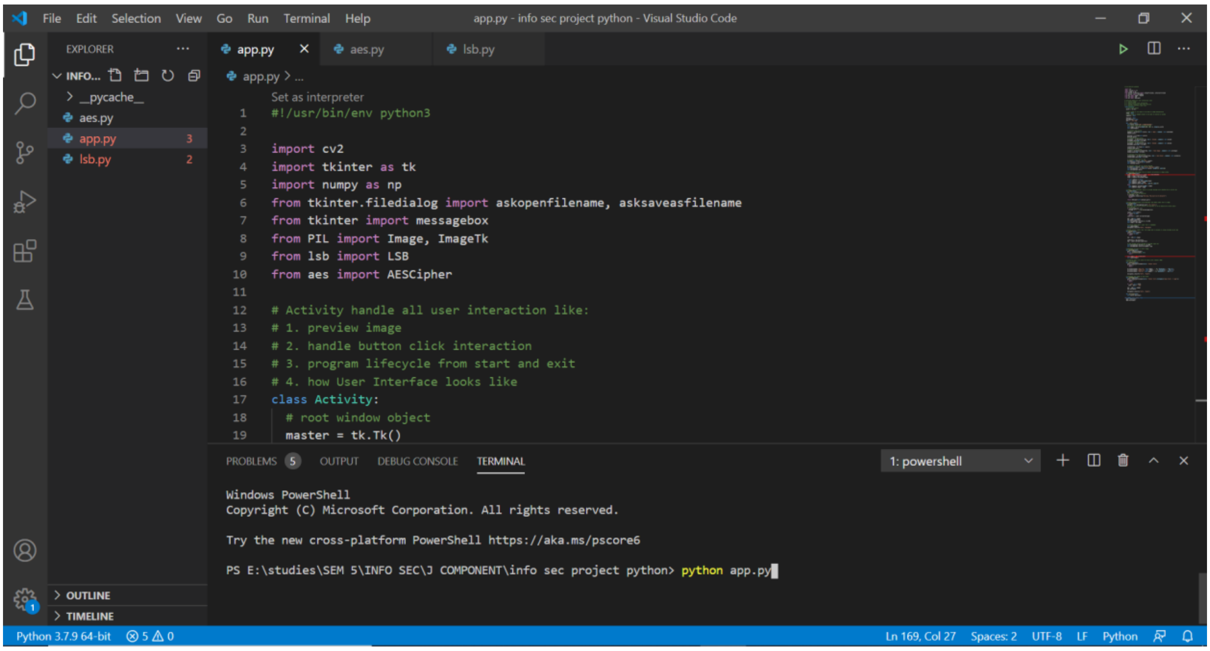


```
%Encoding the message
original=imread('dog.jpg');
cover=rgb2gray(original);
[row,column]=size(cover);
L=256;
stego=cover;

message=input('Enter the message to be hidden: ','s')
len=strlength(message)*8;  %Each character will take 8
ascii=uint8(message);   %ascii is a vector having the
binary_separate=dec2bin(ascii,8);   %binary_separate i
binary_all='';  %binary_all will have the entire seque
for i=1:strlength(message)
    binary_all=append(binary_all,binary_separate(i,:)
```

```
>> lsb
Enter the message to be hidden:
Hello my name is Laya
The original message is: Hello my name is Laya
>> |
```



AES + LSB Python application:

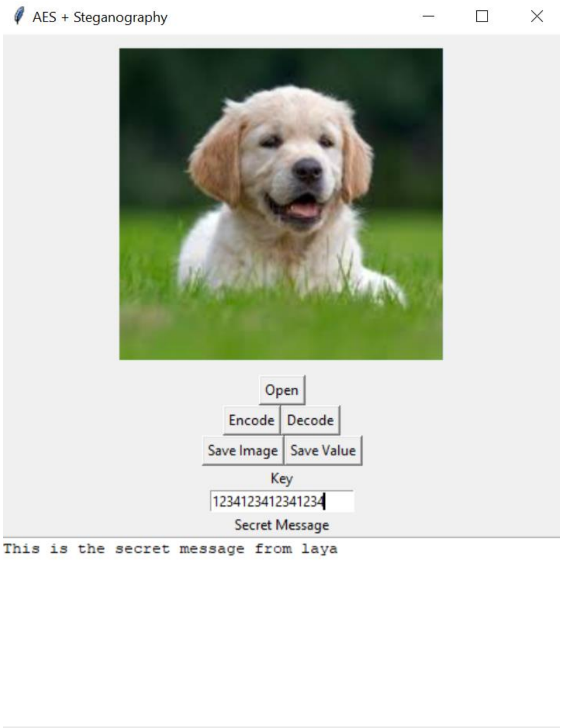


File Edit Selection View Go Run Terminal Help
app.py - info sec project python - Visual Studio Code
EXPLORER
INFO...
pycache
aes.py
app.py 3
lsb.py 2
app.py
aes.py
lsb.py
app.py > ...
OUTLINE
TIMELINE

```
Set as interpreter
#!/usr/bin/env python3

import cv2
import tkinter as tk
import numpy as np
from tkinter.filedialog import askopenfilename, asksaveasfilename
from tkinter import messagebox
from PIL import Image, ImageTk
from lsb import LSB
from aes import AESCipher

# Activity handle all user interaction like:
# 1. preview image
# 2. handle button click interaction
# 3. program lifecycle from start and exit
# 4. how User Interface looks like
class Activity:
  # root window object
  master = tk.Tk()
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\studies\SEM 5\INFO SEC\J COMPONENT\info sec project python> python app.py
```

Python 3.7.9 64-bit ⊗ 5 ⚠ 0
Ln 169, Col 27 Spaces: 2 UTF-8 LF Python



AES + Steganography
Open
Encode
Decode
Save Image
Save Value

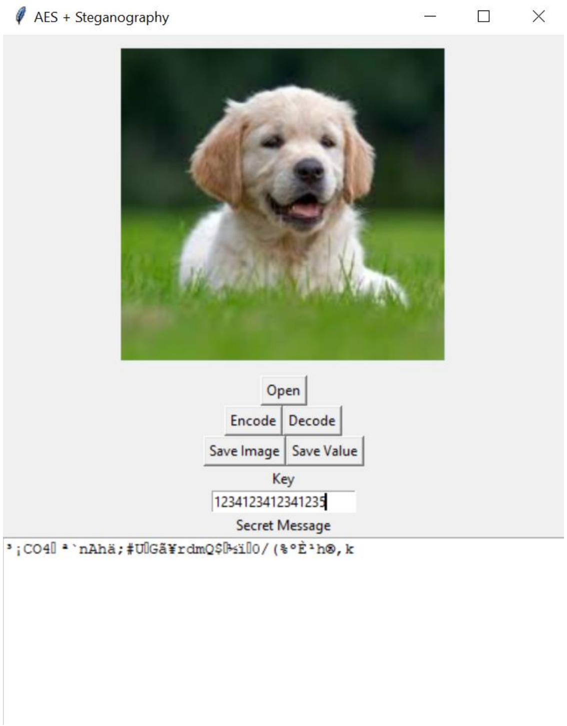
Key

1234123412341234

Secret Message

```
This is the secret message from laya
```

with the correct key



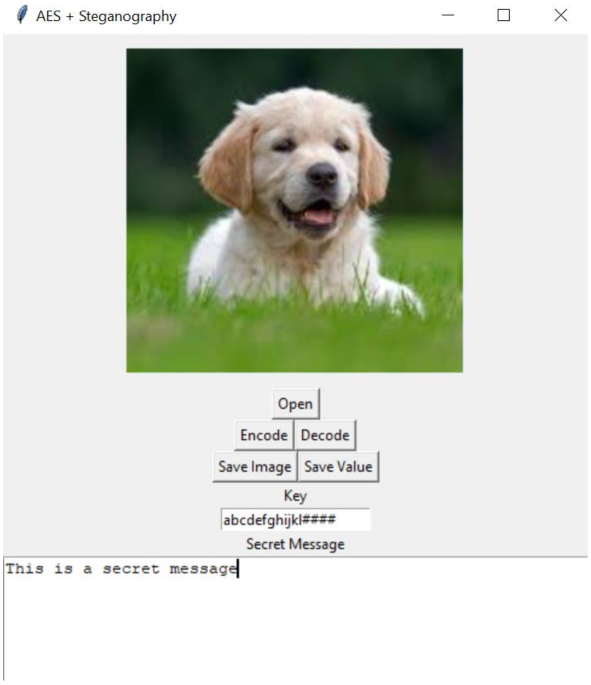
Key

1234123412341235

Secret Message

```
³¡CO4▯ª`nAhä;#U▯Gã¥rdmQ$▯½ï▯0/(%°È¹h®,k
```

with the incorrect key



Key

abcdefghijkl####

Secret Message

This is a secret message

The key can contain numbers and alphanumeric characters

Functionality of the application:

Open: To open a image file from a source path.

Encode: To encode the image file using AES encryption using the provided 16-character key, along with embedding the secret message within the image.

Decode: To decode the image file using AES decryption using the provided 16-character key, along with extracting the secret message from the image only if the key is correct.

Save image: The generated stego image can be saved on the desktop.

Save value: The extracted secret message can be stored on the desktop and a txt file.

5. CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The work accomplished throughout this project is summarized with the subsequent points. During this project we have implemented a secure system using the combination of cryptography, image processing, digital watermarking and steganography. Steganography, particularly combined with cryptography, may be a powerful tool that allows individuals to speak without the eavesdroppers even knowing that there is some kind of communication. The provided methodology provides acceptable image quality with little distortion within the image. The best advantage of this Crypto/Stego System is that the strategy used for encoding, AES, is extremely secure and also the LSB transformation Steganography techniques are very arduous to discover. It additionally shows varied strategy utilized for the smallest amount vital bit relying upon the potency also as encoding standards used. Non repudiation feature using DCT-DWT watermarking technique is a huge plus point compared to other works existing.

5.2 FUTURE WORK

We have implemented AES continued with LSB and watermarking techniques for the secure image transfer. The same can be done for the videos, audios and other type of files. The algorithms in this paper are used in the regular applications and some chatbots for security purpose. It can be further be continued as creating an application that can be used on smartphones and can be connected to cloud. Further watermarking techniques can also be integrated to improve the security levels of digital media.

6. REFERENCES

[1] Mare, Septimiu Fabian, Mircea Vladutiu, and Lucian Prodan. "Secret data communication system using Steganography, AES and RSA." In *2011 IEEE 17th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pp. 339-344. IEEE, 2011.

[2] Sajisha, K. S., and Sheena Mathew. "An encryption based on DNA cryptography and steganography." In *2017 International Conference of Electronics, Communication and Aerospace Technology (ICECA)*, vol. 2, pp. 162-167. IEEE, 2017.

[3] Sheth, Utsav, and Shiva Saxena. "Image steganography using AES encryption and least significant nibble." In *2016 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0876-0879. IEEE, 2016.

[4] Prashanti, G., B. V. Jyothirmai, and K. Sai Chandana. "Data confidentiality using steganography and cryptographic techniques." In *2017 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pp. 1-4. IEEE, 2017.

[5] Islam, Ammad Ul, Faiza Khalid, Mohsin Shah, Zakir Khan, Toqeer Mahmood, Adnan Khan, Usman Ali, and Muhammad Naeem. "An improved image steganography technique based on MSB using bit differencing." In *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*, pp. 265-269. IEEE, 2016.

[6] Mehboob, Beenish, and Rashid Aziz Faruqui. "A stegnography implementation." In *2008 International Symposium on Biometrics and Security Technologies*, pp. 1-5. IEEE, 2008.

[7] Conci, Aura, Andre Luiz Brazil, Simone Bacellar Leal Ferreira, and Trueman MacHenry. "AES cryptography in color image steganography by genetic algorithms." In *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, pp. 1-8. IEEE, 2015.

[8] Sarmah, Dipti Kapoor, and Neha Bajpai. "Proposed system for data hiding using cryptography and steganography." *International Journal of Computer Applications* 8, no. 9 (2010): 7-10.

[9] Reddy, V. Lokeswara, A. Subramanyam, and P. Chenna Reddy. "Implementation of least significant bit steganography and statistical steganalysis." In *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*, pp. 671-675. 2012.

[10] Rajput, G. G., and Ramesh Chavan. "A Novel Approach for Image Steganography based on LSB Technique." In *Proceedings of the International Conference on Compute and Data Analysis*, pp. 167-170. 2017.

****