## ABSTRACT

## The system deals with security during transmission of data. Commonly used technologies are Cryptography. This system deals with implementing security using Steganography. In this end user identifies an image which is going to act as the carrier of data. The data file is also selected and then to achieve greater speed of transmission the data file and image file are compressed and sent. Prior to this the data is embedded into the image and then sent. The image if hacked or interpreted by a third party user will open up in any image previewed but not displaying the data. This protects the data from being invisible and hence be secure during transmission. The user in the receiving end uses another piece of code to retrieve the data from the image.

## STEGANOGRAPHY

## Steganography is not actually a method of encrypting messages, but hiding them within something else to enable them to pass undetected. Traditionally this was achieved with invisible ink, microfilm or taking the first letter from each word of a message. This is now achieved by hiding the message within a graphics. Form instance in a 256-greyscope image, if the least significant bit of each byte is replaced with a bit from the message then the result will be indistinguishable to the human eye. An eavesdropper will not even realize a message is being send. This is not Cryptography however, and although it would fool a human, a computer would be able to detect this very quickly and reproduce the original message.

## In the case of using this technique of hiding the data with an image file, the visibility of the image, resolution or clarity is not being affected. The hidden data can be length in size. To the Hacker, only the image is make going to be visible when previewed and not a trace of the hidden data.

## If the image file is opened across a text editor, then also the data is not going to be visible as the information is stored in an Encryption form, which is also binary. Hence making it difficult for the enclosure to differentiate the data to the image file.

1. **ABOUT PROJECT**

The system deals with security during transmission of data. Commonly used technologies are Cryptography. This system deals with implementing security using Steganography. In this the end user identifies an image, which is going to act as the carrier of data. The data file is also selected and then to achieve greater speed of transmission the data file and image file are compressed and sent. Prior to this the data is embedded into the image and then sent.

The image if hacked or interpreted by a third party user will open up in any image previewed but not displaying the data. This protects the data from being invisible and hence be secure during transmission.

The user in the receiving end uses another piece of code to retrieve the data from the image.

## RSA ALGORITHMS

In 1978, Ron Rivest, Adi Shamir, and Leonard Adleman introduced a cryptographic algorithm, which was essentially to replace the less secure National Bureau of Standards (NBS) algorithm. Most importantly, RSA implements a public-key cryptosystem, as well as digital signatures. RSA is motivated by the published works of Diffie and Hellman from several years before, who described the idea of such an algorithm, but never truly developed it.

**4.1. Introduction**

This algorithm is based on the difficulty of factorizing large numbers that have 2 and only 2 factors (Prime numbers). The system works on a public and private key system. The public key is made available to everyone. With this key a user can encrypt data but cannot decrypt it, the only person who can decrypt it is the one who possesses the private key. It is theoretically possible but extremely difficult to generate the private key from the public key, this makes the RSA algorithm a very popular choice in data encryption.

**4.2. Algorithm**

First of all, two large distinct prime numbers p and q must be generated. The product of these, we call n is a component of the public key. It must be large enough such that the numbers p and q cannot be extracted from it - 512 bits at least i.e. numbers greater than 10154. We then generate the encryption key e which must be co-prime to the number m = ϕ(n) = (p − 1)(q − 1). We then create the decryption key d such that de mod m = 1. We now have both the public and private keys.

**4.3. Encryption**

We let y = E(x) be the encryption function where x is an integer and y is the encrypted form of x

y = x e mod n.

**4.4. Decryption**

We let X = D(y) be the decryption function where y is an encrypted integer and X is the decrypted form of y

X = y d mod n.

**4.5. Simple Example**

1. We start by selecting primes p = 3 and q = 11.

2. n = pq = 33

m = (p − 1)(q − 1) = (2)(10) = 20.

3. Try e = 3

gcd(3, 20) = 1

⇒ e is co-prime to n

4. Find d such that 1 ≡ de mod m

⇒ 1 = Km + de

Using the extended Euclid Algorithm we see that 1 = −1(20) + 7(3)

⇒ d = 7

5. Now let’s say that we want to encrypt the number x = 9

We use the Encryption function y = x e mod n

y = 93 mod 33

y = 729 mod 33 ≡ 3

⇒ y = 3

6. To decrypt y we use the function X = y d mod n

X = 37 mod 33

X = 2187 mod 33 ≡ 9

⇒ X = 9 = x

⇒ It Works!

## APPLICATION

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

## EXISTING SYSTEM

In today’s dynamic and information rich environment, information systems have become vital for any organization to survive. With the increase in the dependence of the organization on the information system, there exists an opportunity for the competitive organizations and disruptive forces to gain access to other organizations information system. This hostile environment makes information systems security issues critical to an organization. Current information security literature either focuses on unreliable information by describing the information security attacks taking place in the world or it comprises of the technical literature describing the types of security threats and the possible security systems.

 In order to secure the transmission of data, Cryptography has to be implemented. Cryptography is the science of devising methods that allow information to be sent in a secure form in such a way that the only person able to retrieve this information is intended recipient.

The basic principle is this: A message being sent is known as plaintext. The message is then coded using a Cryptographic Algorithm. This process is called Encryption, an Encrypted message is known as Cipher Text, and is turned back into plaintext by the process of decryption. Encryption can be done using Symmetric or Asymmetric Algorithms. In Symmetric algorithms, only one key is used both to encrypt and decrypt the message. In Asymmetric Algorithms, a key used to encrypt a message is different from the key that is used to decrypt the message. There are several algorithms present in the market for Cryptography.  Some of the commonly used once are IDEA and RSA that involves Asymmetric or Symmetric methods and also involves private and public keys.

## PROPOSED SYSTEM

The algorithms present in the existing system was some what complicated.  In Cryptography, the meaning of data has been changed. So, it makes intention to the hacker to hack or destroy the data. In our proposed system, we implement a new technology called Steganography for Network security. It not only change the meaning of data but also hides the presence of data from the hackers.

## BENEFITS OR DRAWBACKS

## With LSB Substitution it is quite easy to tell if an image has been Steganographed with an Enhanced LSB Attack. A complex image yields a much better Steganographed image—in that is harder to visually detect Steganography. Chi-Square Analysis can detect Steganography much better than Enhanced LSB’s; however, one can still construct an image to account for statistical irregularities so that when applying Steganography to an image, they can make sure to preserve (as best as possible) the statistical frequencies so that a chi-square analysis fails to produce a qualified result.

## In my project I used a header to help with the embedding and extracting of a message from an image. This allows for quick updates to the encoding algorithm, as well as multiple user-specific modes of encoding. It also allows for better Stengagraphic hiding because the header format changes depending on the message type, mode, and length, making it harder to crack and detect my Steganography.

## Encryption of the message helps to improve the security of the message. Because of the encryption’s poly-alphabetic nature it makes it a lot harder to crack. However, if one can determine the length of the password, the following rule applies:

## charAt(i) = charAt(length+i)

## Once the length is found, the Caesar Cipher must be cracked by performing frequency analysis on the characters and one can (with time) determine the password. However, my implementation pre-encrypts the data so that even before a user applies a password, the message is already encrypted. This renders the frequency analysis useless because it implements the entire ASCII table, not just the alphabet, and it can have more than one character represent another. The complexity of frequency analysis increases quickly and largely on a poly-alphabetic message.

1. **SOFTWARE AND HARDWARE REQUIREMENTS** 
   1. **Hardware Requirements**

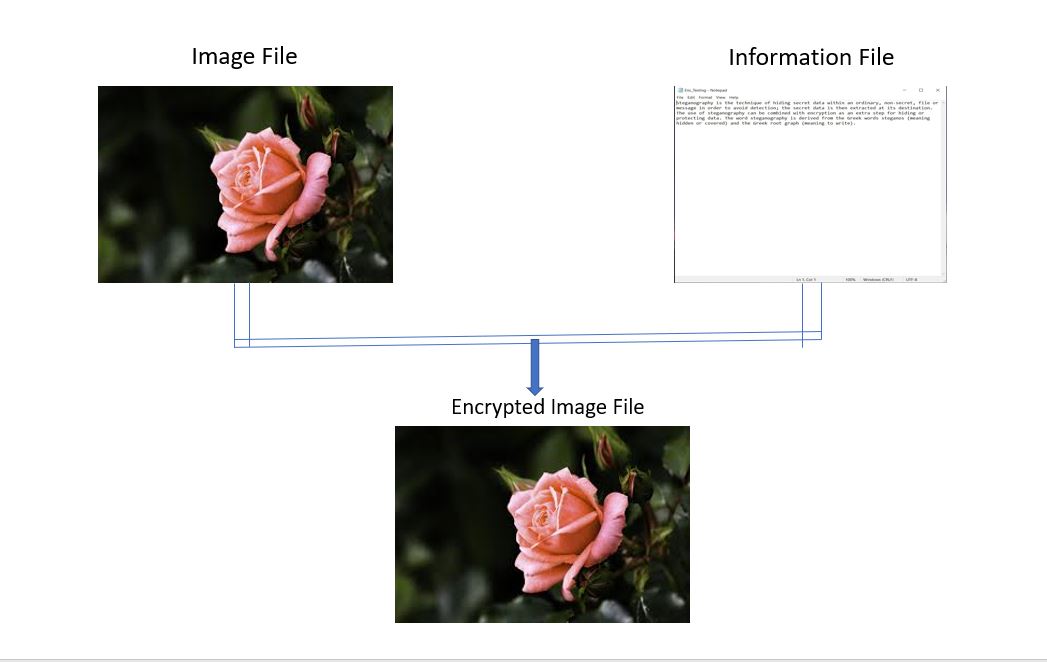
* **Processor :** intel core i3 or i5 with 2.3 GHz Clock Speed.
* **RAM :** 2GB or 4GB.
* **Hard Disk Capacity :** 20GB .
  1. **Software Requirements**
* **Operating System :** Windows 10 Home Single Language with 32 or 64 bits.
* **Front-End Design :** JDK 1.8 with Swing Components.

#### GRAPHICAL USER INTERFACE

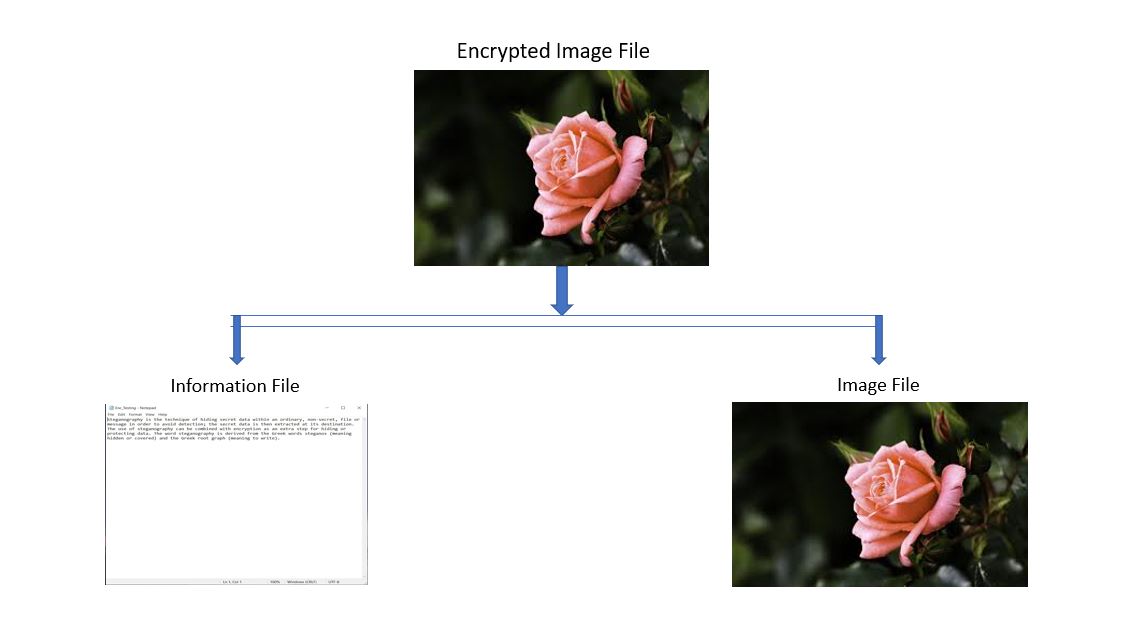
This project is developed using graphics in Java Swings. The options available are displayed in a menu format, like in an online editor. Clicking on any particular menu item through mouse or through keyboard a dropdown menu is displayed, listing all the options available under that menu item and the user can select the needed actions according to their wish.

#### ENCRYPTION AND DECRYPTION PROCESS

* 1. **Encryption Process**

****

* 1. **Decryption Process**

****

1. **CODE ANALYSIS**
   1. **RSA Algorithm Code**

import java.math.BigInteger;

import java.security.SecureRandom;

public class RSA{

private static final BigInteger one=new BigInteger("1");

private static final SecureRandom random=new SecureRandom();

private BigInteger publicKey;

private BigInteger privateKey;

private BigInteger modulus;

private BigInteger phi;

private static final int N=50;

public RSA(){

BigInteger p=BigInteger.probablePrime(N/2,random);

BigInteger q=BigInteger.probablePrime(N/2,random);

//System.out.println("p: "+p+"\nq: "+q);

modulus=p.multiply(q);

BigInteger x=p.subtract(one);

BigInteger y=q.subtract(one);

phi=x.multiply(y);

publicKey=new BigInteger("65537");

privateKey=publicKey.modInverse(phi);

}

public BigInteger encrypt(BigInteger original){

BigInteger encrypted=original.modPow(publicKey,modulus);

return encrypted;

}

public BigInteger decrypt(BigInteger encrypted){

BigInteger decrypted=encrypted.modPow(privateKey,modulus);

return decrypted;

}

public BigInteger getPrivateKey(){

return privateKey;

}

public BigInteger getModulus(){

return modulus;

}

}

* 1. **Data Encryption Code**

import java.awt.BorderLayout;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.filechooser.FileNameExtensionFilter;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

import javax.swing.JButton;

import javax.swing.JFileChooser;

import java.awt.event.ActionListener;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.math.BigInteger;

import java.awt.event.ActionEvent;

import javax.swing.JPasswordField;

public class Encrypt extends JFrame {

private JPanel contentPane;

JPasswordField pw;

JTextField t1,t2;

JButton b1,b2,b3,ok;

File textfile,imagefile;

static final int N=50;

RSA rsa=new RSA();

JLabel l3;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

Encrypt frame = new Encrypt();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public Encrypt() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 634, 387);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JLabel lblNewLabel = new JLabel("Select the text File");

lblNewLabel.setBounds(40, 57, 109, 24);

contentPane.add(lblNewLabel);

JLabel lblSelectAnImage = new JLabel("Select an Image");

lblSelectAnImage.setBounds(40, 132, 109, 24);

contentPane.add(lblSelectAnImage);

t1 = new JTextField();

t1.setBounds(202, 59, 204, 24);

contentPane.add(t1);

t1.setColumns(10);

t2 = new JTextField();

t2.setColumns(10);

t2.setBounds(202, 134, 204, 24);

contentPane.add(t2);

JButton b1 = new JButton("Choose");

b1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

JFileChooser file\_chooser=new JFileChooser();

int v=file\_chooser.showOpenDialog(Encrypt.this);

FileNameExtensionFilter filter=new FileNameExtensionFilter("Text File","txt");

file\_chooser.setFileFilter(filter);

if(v==JFileChooser.APPROVE\_OPTION){

textfile=file\_chooser.getSelectedFile();

String abspath=textfile.getAbsolutePath();

t1.setText(abspath);

}

}

});

b1.setBounds(444, 58, 89, 23);

contentPane.add(b1);

JButton b2 = new JButton("Choose");

b2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

JFileChooser file\_chooser=new JFileChooser();

int v=file\_chooser.showSaveDialog(Encrypt.this);

FileNameExtensionFilter filter=new FileNameExtensionFilter("JPG & GIF","jpg","gif","png");

file\_chooser.setFileFilter(filter);

if(v==JFileChooser.APPROVE\_OPTION){

imagefile=file\_chooser.getSelectedFile();

String abspath=imagefile.getAbsolutePath();

t2.setText(abspath);

}

}

});

b2.setBounds(444, 135, 89, 23);

contentPane.add(b2);

JButton b3 = new JButton("Encrypt");

b3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

l3.setVisible(true);

ok.setVisible(true);

pw.setVisible(true);

b3.setEnabled(false);

ok.setEnabled(true);

}

});

b3.setBounds(202, 257, 89, 23);

contentPane.add(b3);

l3 = new JLabel("Enter Password");

l3.setBounds(40, 211, 109, 23);

contentPane.add(l3);

l3.setVisible(false);

pw = new JPasswordField();

pw.setBounds(202, 212, 204, 20);

contentPane.add(pw);

pw.setVisible(false);

ok = new JButton("OK");

ok.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try{

String text="";

FileInputStream fis=new FileInputStream(textfile);

int i=0;

while((i=fis.read())!=-1){

text=text+(char)i;

}

System.out.println("Actual Message: "+text);

/\*--- dividing the text into N/8 i.e 6 bytes each and encrypting each chunk -----\*/

String enc\_msg="";

int max=N/8;

for(i=0;i<text.length();i=i+max){

if(text.length()-i>max){

int j=i+max;

String sub=text.substring(i,j);

byte b[]=sub.getBytes();

BigInteger bin=new BigInteger(b);

BigInteger enc=rsa.encrypt(bin);

String enc\_s=enc.toString();

enc\_msg=enc\_msg+enc\_s+"#";

System.out.println(sub+"->"+enc\_s);

}

else{

String sub=text.substring(i);

byte b[]=sub.getBytes();

BigInteger bin=new BigInteger(b);

BigInteger enc=rsa.encrypt(bin);

String enc\_s=enc.toString();

enc\_msg=enc\_msg+enc\_s+"#";

System.out.println(sub+"->"+enc\_s);

}

}

System.out.println("Encrypted Message: "+enc\_msg);

String pwd=pw.getText();

if(pwd.length()>=4 && pwd.length()<=6){

byte bb[]=pwd.getBytes();

BigInteger pass=new BigInteger(bb);

BigInteger encpwd=rsa.encrypt(pass);

enc\_msg=encpwd+"#"+enc\_msg+rsa.getModulus()+"#"+rsa.getPrivateKey();

System.out.println("Encrypted Message with password and key: "+enc\_msg);

enc\_msg=" "+enc\_msg;

FileOutputStream out=new FileOutputStream(imagefile,true);

byte ascii[]=enc\_msg.getBytes();

out.write(ascii);

JOptionPane.showMessageDialog(null,"Message has been encrypted to "+imagefile.getName());

out.close();

fis.close();

ok.setEnabled(false);

// b4.setEnabled(true);

}

else

JOptionPane.showMessageDialog(null,"Password must contain 4 to 6 chars");

}

catch(Exception ex){

ex.printStackTrace();

}

}

});

ok.setBounds(444, 211, 89, 23);

contentPane.add(ok);

ok.setVisible(false);

}

}

* 1. **Data Decryption Code**

import java.awt.BorderLayout;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.filechooser.FileNameExtensionFilter;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

import javax.swing.JPasswordField;

import javax.swing.JButton;

import javax.swing.JFileChooser;

import java.awt.event.ActionListener;

import java.io.File;

import java.io.FileOutputStream;

import java.io.RandomAccessFile;

import java.math.BigInteger;

import java.util.RandomAccess;

import java.awt.event.ActionEvent;

public class Decrypt extends JFrame {

private JPanel contentPane;

private JTextField t1;

private JTextField t2;

private JPasswordField pw;

private JButton b1,b2,b3;

File textfile,imagefile;

static final int N=50;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

Decrypt frame = new Decrypt();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public Decrypt() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 639, 374);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JLabel lblNewLabel = new JLabel("Select the image file");

lblNewLabel.setBounds(49, 56, 141, 14);

contentPane.add(lblNewLabel);

JLabel lblNewLabel\_1 = new JLabel("Select a text file");

lblNewLabel\_1.setBounds(49, 121, 147, 17);

contentPane.add(lblNewLabel\_1);

JLabel lblNewLabel\_2 = new JLabel("Password");

lblNewLabel\_2.setBounds(49, 184, 154, 14);

contentPane.add(lblNewLabel\_2);

t1 = new JTextField();

t1.setBounds(225, 54, 197, 23);

contentPane.add(t1);

t1.setColumns(10);

t2 = new JTextField();

t2.setColumns(10);

t2.setBounds(225, 118, 197, 20);

contentPane.add(t2);

pw = new JPasswordField();

pw.setBounds(228, 181, 194, 23);

contentPane.add(pw);

b3 = new JButton("Decrypt");

b3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Object ob=e.getSource();

if(ob==b3)

{

try

{

int index=0,cnt=0;

RandomAccessFile file=new RandomAccessFile(imagefile, "r");

int i=0;

while((i=file.read())!=-1)

{

cnt++;

if(i==32)

index=cnt;

}

file.seek(index);

String str="";

while((i=file.read())!=-1)

{

str=str+(char)i;

}

System.out.println(str);

str=str.trim();

String msg[]=str.split("#");

String en=msg[0];

String modulus=msg[msg.length-2];

String pk=msg[msg.length-1];

System.out.println("Enc Password"+en);

System.out.println("Modulus"+modulus);

System.out.println("Private Key"+pk);

BigInteger p=new BigInteger(en);

BigInteger m=new BigInteger(modulus);

BigInteger pk2=new BigInteger(pk);

BigInteger pwd=p.modPow(pk2, m);

byte b[]=pwd.toByteArray();

String pass=new String(b);

String passwd=pw.getText();

if(pass.equals(pass))

{

String original="";

for(i=1;i<=msg.length-3;i++)

{

BigInteger em=new BigInteger(msg[i]);

BigInteger dm=em.modPow(pk2, m);

byte b1[]=dm.toByteArray();

String om=new String(b1);

original=original+om;

}

//System.out.println("Message Decrypted");

FileOutputStream out=new FileOutputStream(textfile);

out.write(original.getBytes());

JOptionPane.showMessageDialog(null, "Message decrypted");

out.close();

}else

{

JOptionPane.showMessageDialog(null, "Invalid password");

file.close();

}

}catch (Exception e1) {

e1.printStackTrace();

}

}

}

});

b3.setBounds(203, 245, 89, 23);

contentPane.add(b3);

b1 = new JButton("Choose");

b1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Object ob=e.getSource();

if(ob==b1)

{

JFileChooser file\_chooser=new JFileChooser();

int v=file\_chooser.showOpenDialog(Decrypt.this);

FileNameExtensionFilter filter=new FileNameExtensionFilter("Image File","jpg","png");

file\_chooser.setFileFilter(filter);

if(v==JFileChooser.APPROVE\_OPTION)

{

imagefile=file\_chooser.getSelectedFile();

String abspath=imagefile.getAbsolutePath();

t1.setText(abspath);

}

}

}

});

b1.setBounds(457, 52, 89, 23);

contentPane.add(b1);

b2 = new JButton("Choose");

b2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Object ob=e.getSource();

if(ob==b2)

{

JFileChooser file\_chooser=new JFileChooser();

int v=file\_chooser.showOpenDialog(Decrypt.this);

FileNameExtensionFilter filter=new FileNameExtensionFilter("Image File","txt");

file\_chooser.setFileFilter(filter);

if(v==JFileChooser.APPROVE\_OPTION)

{

textfile=file\_chooser.getSelectedFile();

String abspath=textfile.getAbsolutePath();

t2.setText(abspath);

}

}

}

});

b2.setBounds(457, 118, 89, 23);

contentPane.add(b2);

}

}

1. **SYSTEM TESTING**
   1. **Introduction to Testing**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. It is the major quality measure employed during software development.

Testing is the exposure of the system to trial input to see whether it produces correct output. It is a process, which reveals errors in the program. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

* 1. **Testing Phases**

Software testing phases include the following:

1. Test activities are determined and test data selected.

2. The test is conducted and test results are compared with the expected results.

There are various types of Testing:

**13.2.1 Unit Testing**

Unit testing is essentially for the verification of the code produced during the coding phase and the goal is test the internal logic of the module/program.

**13.2.2 Integration Testing**

All the tested modules are combined into sub systems, which are then tested. The goal is to see if the modules are properly integrated, and the emphasis being on the testing interfaces between the modules.

**13.2.3 System Testing**

It is mainly used if the software meets its requirements. The reference document for this process is the requirement document.

**13.2.4 Acceptiance Testing**

It is performed with realistic data of the client to demonstrate that the software is working satisfactorily.

* 1. **Testing Methods**

Testing is a process of executing a program to find out errors. If testing is conducted successfully, it will uncover all the errors in the software. Any testing can be done basing on two ways:

**13.3.1 White Box Testing**

It is a test case design method that uses the control structures of the procedural design to derive test cases. In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases.

Using this testing a Software Engineer can derive the following test cases:

Exercise all the logical decisions on either true or false sides. Execute all loops at their boundaries and within their operational boundaries. Exercise the internal data structures to assure their validity.

White Box testing attempts to find errors in the following categories:

• Guarantee that all independent paths have been executed.

• Execute all logical decisions on their true and false Sides.

• Execute all loops at their boundaries and within their operational bounds

• Execute internal data structures to ensure their validity

**13.3.2 Black Box Testing**

It is a test case design method used on the functional requirements of the software. In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program It will help a software engineer to derive sets of input conditions that will exercise all the functional requirements of the program.

Black Box testing attempts to find errors in the following categories:

• Incorrect or missing functions

• Interface errors

• Errors in data structure or external database access

• Performance errors

• Initialization and termination errors

* 1. **Testing Approach**

Testing can be done in two ways:

• Bottom up approach

• Top down approach

**13.4.1 Bottom Up Approach**

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded with in the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

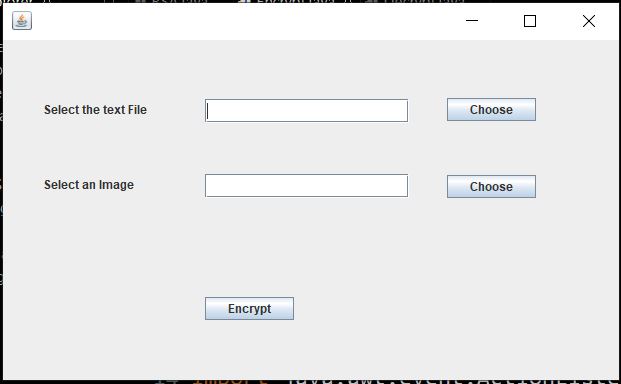
**13.4.2 Top Down Approach**

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

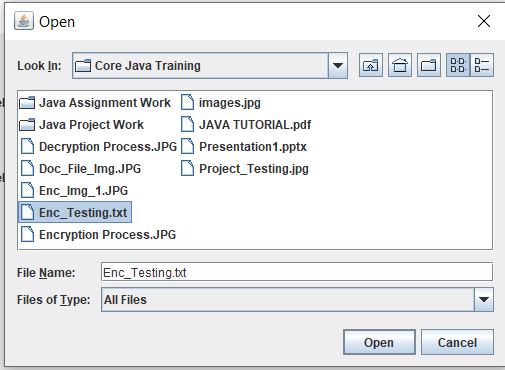
**13.4.3 Validation**

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

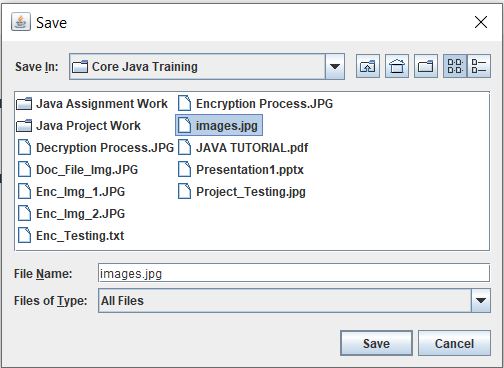
1. **INPUTS & OUTPUTS SCREENS**
   1. **Encryption**
2. This is an Image of Data Encryption which has two Button and two TextField for Data Encryption. The First Button is select a text file from your Desktop where your dada is loaded and the second Select Button is to select an Image file that where you want to encrypt the data.



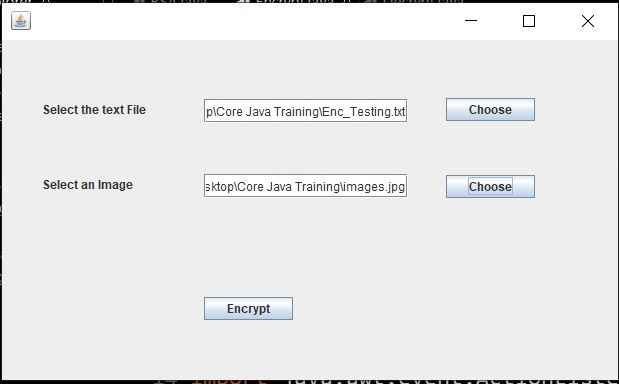
1. After Clicking the First Choose Button then we have to select a Text file where your data is stored. In my case this file name is “Enc\_Testing.txt”.

****

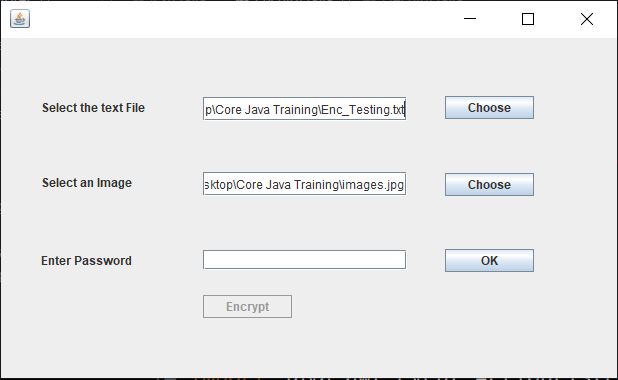
1. After selecting the text file then we have to select another image file. Then click second Choose Button and select a Image file. In my case the name of the Image file is “images.jpg”

****

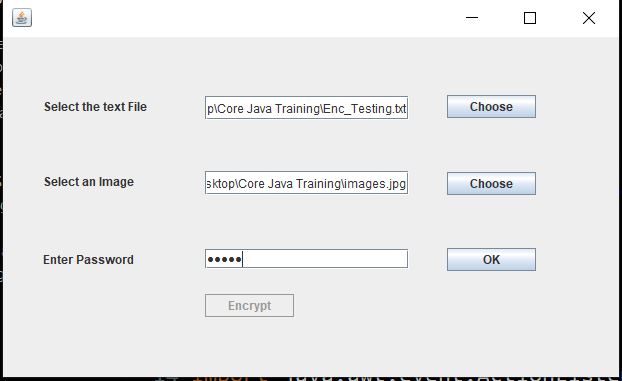
1. After Selecting one Text file and one Image file then click the “Encrypt” button.

****

1. After clicking Encrypt Button another TextField and a Button will come for Password.

****

1. After putting Password then click to “OK” Button for Encryption.

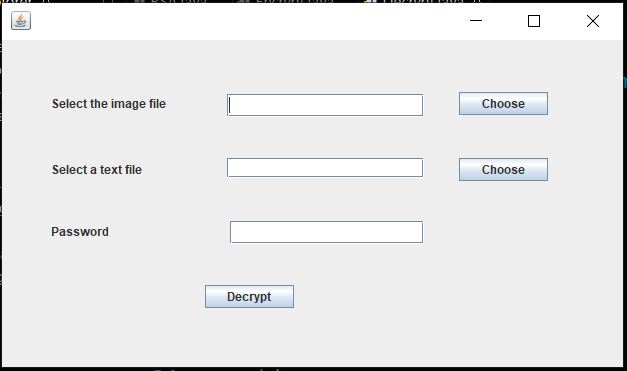
****

1. At the End If your data is successfully Encrypted then a Message will come for conformation.

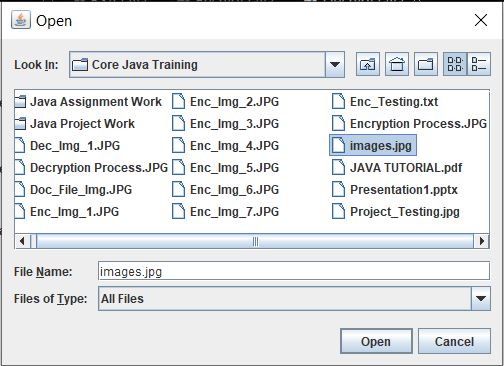
****

* 1. **Decryption**

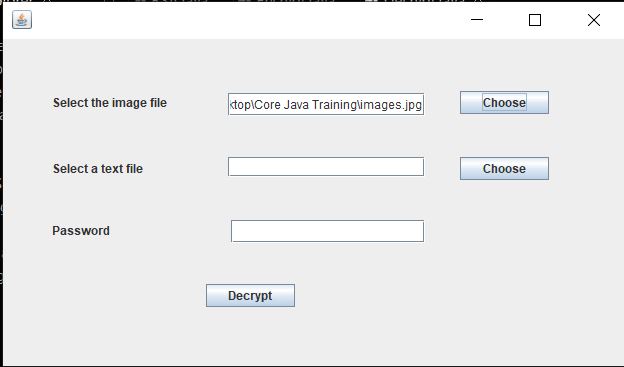
1. This is an Image of Data Decryption which has two Button and two TextField for Data Decryption which is totally blank where Encrypted data will stored. The First Button is select a Image file from your Desktop where your hiding dada is loaded and the second Select Button is to select an Text file that where you want to Decrypt the data.

****

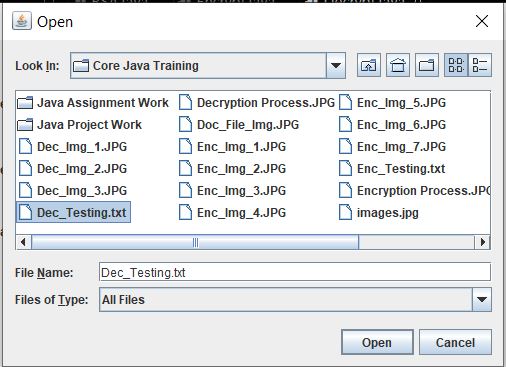
1. First We have to click first choose button and Select a Image file where the data is hided or Encrypted. In my case The File name is “images.jpg”.

****

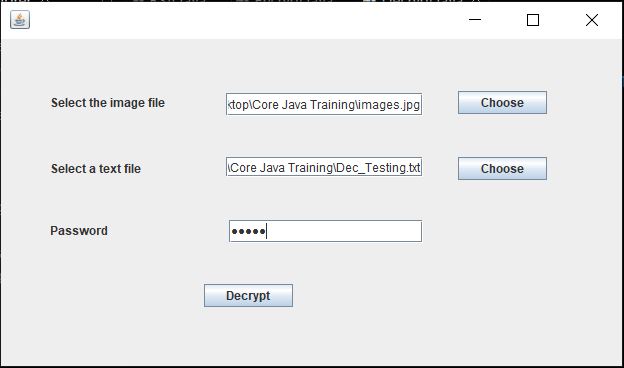
1. After Selecting the Image file then click next Choose button.

****

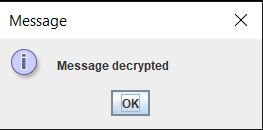
1. Then select a blank text file on your Desktop. In my case This blank text file name is “Dec\_Testing.txt”.

****

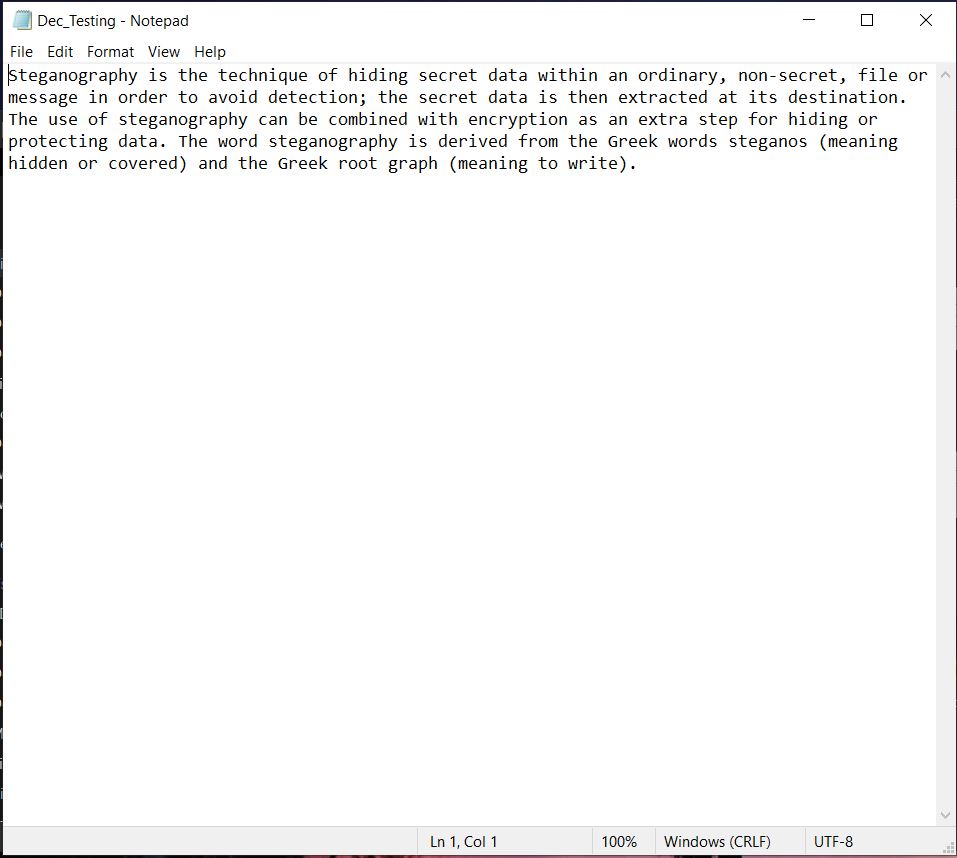
1. After choose one Blank Text file and one Image file, then put the Password Correctly.

****

1. Then click the Decrypt Button for Decryption and a message will come for the conformation.

****

1. After that a your Encrypted text file will be visible on the blank text file which you created for store the data of Decryption.

****

1. **CONCLUSION**

Steganography is a really interesting subject and outside of the mainstream cryptography and system administration that most of us deal with day after day.

Steganography can be used for hidden communication. We have explored the limits of steganography theory and practice. We printed out the enhancement of the image steganography system using LSB approach to provide a means of secure communication. A stego-key has been applied to the system during embedment of the message into the cover image.

This steganography application software provided for the purpose to how to use any type of image formats to hiding any type of files inside their. The master work of this application is in supporting any type of pictures without need to convert to bitmap, and lower limitation on file size to hide, because of using maximum memory space in pictures to hide the file.

Since ancient times, man has found a desire in the ability to communicate covertly. The recent explosion of research in watermarking to protect intellectual property is evidence that steganography is not just limited to military or espionage applications. Steganography, like cryptography, will play an increasing role in the future of secure communication in the “digital world”.

1. **BIBLIOGRAPHY**

**Websites**

Following websites are referring to create this project reports.

* [www.javatpoint.com](http://www.javatpoint.com)
* [www.geeksforgeeks.org](http://www.geeksforgeeks.org)
* [www.tutorialspoint.com](http://www.tutorialspoint.com)

**Books**

Following books and ebook are used to complete this project reports.

* Head First Java by Kathy Sierra & Bert Bates
* Object Oriented Programming Language by Vijay Bhaskar