**PROJECT REPORT**

Submitted for Network Information & Security in MCA ( ITA6007)

By

**GOURAV DUTTA - 18MCA1087**

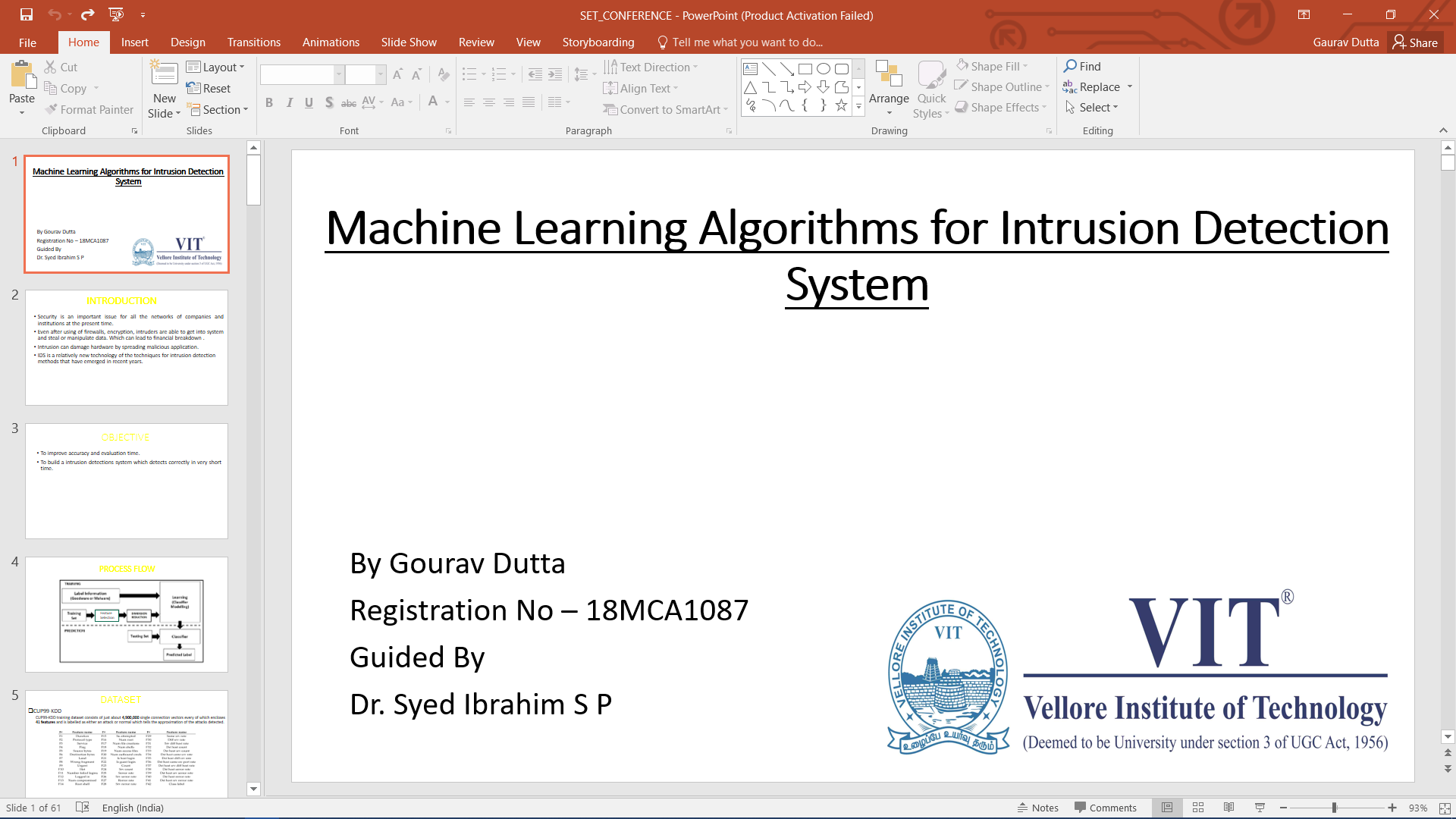
**SUBRATA SAHOO – 18MCA1072**

**DIPESH GYANCHANDANI – 18MCA1046**

**DIGITAL STEGNOGRAPHY**

**NAME OF FACULTY – DR. ASHA S**

**(SCHOOL OF COMPUTING SCIENCE AND ENGINEERING)**



May, 2019

ACKNOWLEDGEMENTS

In the completion of this project I have put in sincere efforts. First of all I would like to express our gratitude to Dr. ASHA S. for giving us constant guidance thought the project and showing is the way. I am thankful to our university for providing me with the education and recourses needed for the project. Finally, I am extremely thankful to my parents for making me capable of doing this project and providing me with the monetary support required. Gratitude to my fellow mates for helping me and having fruitful discussions regarding the project.

**GOURAV DUTTA– 18MCA1087**

**SUBRATA SAHOO – 18MCA1072**

**DIPESH GYANCHANDANI – 18MCA1046**

ABSTRACT

Maintaining secrecy is very important in a large corporation and because of the intelligent of the hackers it becomes tedious. Already we have crypto graphy for transmitting secret information. Even though crypto graphy successfully transmitting secret information, it will give a suspicion to the hackers and it affects unintended users.

Our project, **DIGITAL STEGANO GRAPHY** overcomes this factor and it gives a solution for transmitting secret formation without affecting unintended users. Steganography uses multimedia data as a covering medium (Covering secret information). By using steganography data (secret information) can hided with in data (multimedia data, here multimedia data is an image) and it can be sent anywhere to transfer the message easily without giving any suspicion to others.

METHODOLOGY

**2. SYSTEM ANALYSIS**

**2.1 EXISTING SYSTEM**

In the existing system, secret messages can be transferred but it gives irritation to the unintended people. And also maintaining secrecy is very tough because of the intelligent of the hackers. Already we have cryptography for transmitting secret information. Even though cryptography successfully transmitting secret information, it will give a suspicion to the hackers and it affects unintended users.

* 1. **PROPOSED SYSTEM**

Our project, **DIGITAL STEGANO GRAPHY** overcomes this factor and it gives a solution for transmitting secret formation without affecting unintended users. Steganography uses multimedia data as a covering medium (Covering secret information). By using steganography data (secret information) can hided with in data (multimedia data, here multimedia data is an image) and it can be sent anywhere to transfer the message easily without giving any suspicion to others.

**Features of proposed system:**

1. Provides a user friendlier interface.
2. Developed in java. So platform independent.
3. Highly flexible.

**3. SYSTEM IMPLEMENTATION**

**3.1 HARDWARE REQUIREMENT**

* Processor: Pentium IV
* 40 GB hard disk space.
* 256 MB RAM or more.
* 1.44 Floppy Disk Drive.
* 104 keys keyboard.
* Display capable of showing 65,000 colors or more.
* CD-ROM Drive for installing the package.
* Mouse with minimum two buttons.

**3.2 SOFTWARE REQUIREMENT**

* + - * Front End: Java (jdk1.4.1 and above)
      * OS : Windows / Linux / Solaris

**3.3 SOFTWARE DESCRIPTION**

**3.3.1 JAVA**

**Java Features Of Java**

The inventors of Java wanted to design a language which could offer solutions to some of the problems encountered in modern programming. They wanted the language to be not only reliable, portable and distributed but also simple, compact and interactive. Sun Microsystems officially describes java with the following attributes.

**Compiled and Interpreted**

Usually a computer language is either compiled or interpreted. Java combines both these approaches thus making java a two-stage system. First, java compiler translates source code into what is known as byte code instructions. Byte codes are not machine instructions and therefore, in the second stage, java interpreter generates machine code that can be directly executed by the machine that is running the java program. We can thus say that java is both a compiled and interpreted languages.

**Platform-Independent and Portable**

The most significant contribution of java over other languages is its portability. Java programs can be easily moved from one computer system to another, anywhere and anytime. Changes and upgrades in operating systems, processors and system resources will not force any changes in Java programs. This is the reason why Java has become a popular language for programming on Internet which interconnects different kinds of systems worldwide. We can download a Java applet from a remote computer onto out local system via Internet and execute it locally. This makes the Internet an extension of the user’s basic system providing practically unlimited number of accessible applets and applications.

Java ensures portability in two ways. First, Java compiler generates byte code instructions that can be implemented on any machine. Secondly, the sizes of the primitive’s data types are machine-independent.

**Object-Oriented**

Java is a true object-oriented language. Almost everything in Java is an object. All program code and data reside within objects and classes. Java comes with an extensive set of classes, arranged in packages that we can use in our programs by inheritance. The object model in Java is simple and easy to extend.

**Robust and Secure**

Java is a robust language. It provides many safeguards to ensure reliable code. It has strict compile time and run time checking for data types. It is designed as a garbage-collected language relieving the programmers virtually all memory management problems. Java also incorporates the concept of exception handling which captures series errors and eliminates any risk of crashing the system.

Security becomes an important issue for a language that is used for programming on Internet. Threat of viruses and abuse of resources is everywhere. Java systems not only verify all memory access but also ensure that no viruses are communicated with an applet. The absence of pointer in Java ensures that programs cannot gain access to memory locations without proper authorization.

**Distributed**

Java is designed as a distributed language for creating applications on networks. It has the ability to share both data and programs. Java applications can open and access remote objects on Internet as easily as they can do in a local system. This enables multiple programmers at multiple remote locations to collaborate and work together on a single project.

**Simple, Small and Familiar**

Java is a small and simple language. Many features of C and C++ that are either redundant or sources of unreliable code are not part of Java. For example, java does not use pointers, preprocessor header files, go to statement and many others. It also eliminates operators overloading and multiple inheritance.

Familiarity is another striking feature of Java. To make the language look familiar to the existing programmers, it was modeled on C and C++ languages. Java uses many constructs of C and C++ and therefore, Java code “looks like a C++” code.

**Multithreaded and Interactive**

Multithreaded means handling multiple tasks simultaneously. Java supports multithreaded programs. This means that we need not wait for the application to finish one task before beginning another. For example, we can listen to an audio clip while scrolling a page and at the same time download an applet from a distant computer. This feature greatly improves the interactive performance of graphical applications.

The Java runtimes comes with tools that support multiprocess synchronization and construct smoothly running interactive systems.

**High Performance**

Java performance is impressive for an interpreted language, mainly due to the use of intermediate byte code. According to Sun, Java speed is comparable to the native C/C++. Java architecture is also designed to reduce overheads during runtime. Further, the incorporation of multithreading enhances the overall execution speed of java programs.

**Dynamic and Extensible**

Java is a dynamic language. Java is capable of dynamically linking in new class libraries, methods and objects. Java can also determine the type of class through a query, making it possible to either dynamically link or abort the program, depending on the response.

Java programs support functions written in other languages such as C and C++. These functions are known as native methods. This facility enables the programmers to use the efficient functions available in these languages. Native methods are linked dynamically at runtime.

**3.3.2 SWING - OVERVIEW**

The original GUI components from the Abstract Windowing Toolkit package Java.awt (also called the AWT) are tied directly to the local platform’s graphical user interface capabilities. So, a java program executing on different platforms has a different appearance and sometimes even different user interacts with the program are known as that program’s look and feel. The Swing components allow the programmer to specify a different look and feel across all platforms, or even to change the look-and-feel while the program is running.

Swing components are often referred to as lightweight components they are written completely in java so they are not “weighed down” by the complex GUI capabilities of the platform on which they are used. AWT

Components (many of which parallel the Swing components) that are tied to the local platform are correspondingly called heavyweight components they are rely on the local platform’s windowing system to determine their functionality and their look feel. Each heavyweight component has a peer (from package java.awt.peer) that is responsible for the interactions between the component and the local platform to display and manipulate the component.

**FEATURES OF SWING OVER AWT:**

Even the simplest Swing components have capabilities far beyond what the AWT components offer.

1. Swing buttons and labels can display images instead of , or in addition to, text
2. You can easily add or change the borders drawn around most Swing components. For example, it’s easy to put a box around the outside of a container or label.
3. You can easily change the behavior or appearance of a Swing component by either invoking methods on it or creating a subclass of it.
4. Swing components don’t have to be rectangular. Buttons, for example, can be round.
5. Assistive technologies such as screen readers can easily get information from Swing components. For example, a tool can easily get the text that’s displayed on a button or label.

Swing lets you specify which look and feel your program’s GUI uses. By contrast, AWT components always have the look and feel of the native platform.

**Chapter 4**

**SYSTEM DESIGN**

**4.1 ARCHITECTURE OF THE SYSTEM**

**Architecture of the system**

**Receiver** (Extracting Secret Information)

Secret Code

Stegano Medium

Secret Information

**Sender** (Hiding Data)

(Secret code + Secret Information + Multimedia data)

Stegano Medium

Stegano Medium

**4.2 MODULE DESCRIPTION**

In this project there are two modules, namely

1. “Making stegano Medium”
2. “Getting secret information from stegano medium”

Each of the modules is described in detail as follows.

In making stegano Medium side, the secret information is hided with in an image file. Before hiding, for security, user has to enter a user code and secret information. A secret code will be generated using user code + secret information and this secret code will be used by the receiver to extract the secret information. After generating secret code stegano medium will be generated. This stegano medium is the final output and expected output from the sender side.

In getting secret information from stegano medium Side, Actually anyone may get this stegano medium that is picture with secret information, but only the person who knows secret code can read the message. Inputs for breaking the stegano medium are stegano mediam and secret code.

**4.3 DETAILED DESIGN**

**Algorithm:**

**Making Stegano Medium:**

Step 1: Start the process

Step 2: Enter the Secret Information

Step 3: Enter the User Code

Step 4: Load a multimedia data, here it is an Image

Step 5: Creation of Secret Code by using user code + secret information

Step 6: Hiding secret information with its security into the multimedia data

Step 7: A message box showing the secret key will appear

Step 8: Stop the process

**Extracting secret information from Steganography medium:**

Step 1: Start the process

Step 2: Enter the Secret Code

Step 3: Enter the Stegano Medium

Step 4: Extract secret information from stegano medium by using secret code.

Step 5: Stop the Process

**4.4 SOFTWARE MODELING**

**Sequence Diagram:**

Sender

Hide

1: Load

Picture

2: User

Code

3: Secret

Information

4. Stegano

Medium

5: Secret

Code

Receiver

Break

1: Stegano

Medium

2: Secret

Code

5: Secret

Information

**Use Case Diagram:**

**Sender:**

Creating Stegano Medium

Generating Secret Code

Entering User Code

Sender

Loading Picture

Entering Secret Information

**Receiver:**

Entering Secret Code

Receiver

Loading Stegano

Medium

Getting Secret Information

**Class Diagram:**

**Client:**





Hide()

Break()

Hide

User Code

Secret Information

Picture

Setagano Medium

Secret Code

Hide()

Break

Secret Code

Setagano Medium

Secret Information

Break()

**Activity Diagram:**

Start Stegano Application

Select Operation

Start

End

Hide

Break

Load Image

Load Image

Enter User Code

Enter Secret Code

Generate Secret Code

Extract Secret Information

Generate Stegano Medium

End

**Chapter 5**

**SYSTEM IMPLEMENTATION**

This project needs a java development kit (J2sdk1.4.1 and above). Project is implemented in java, so it can be run in any OS. For hiding data with in a picture we need to run the sender side program. For extracting the hidden secret information we need to run receiver side program.

Our stegano graphy application will contain both sender and receiver side programs. If a user wants to hide data he can use sender side program and if he wants to extract secret information he can use the receiver side program.

**Chapter 6**

**SYSTEM TESTING**

The testing of a conventional software system involves some of the following phases. They are

* Unit Testing
* Integrated Testing
* System Testing

Unit Testing:

A software module can be created by building up of many small parts into a single module. This small part is called as a unit. A unit is a piece of code that will perform a specific task. At the end of this testing all units will be tested so that we can get the correct result. By using unit testing we can easily identify the errors.

Integration Testing:

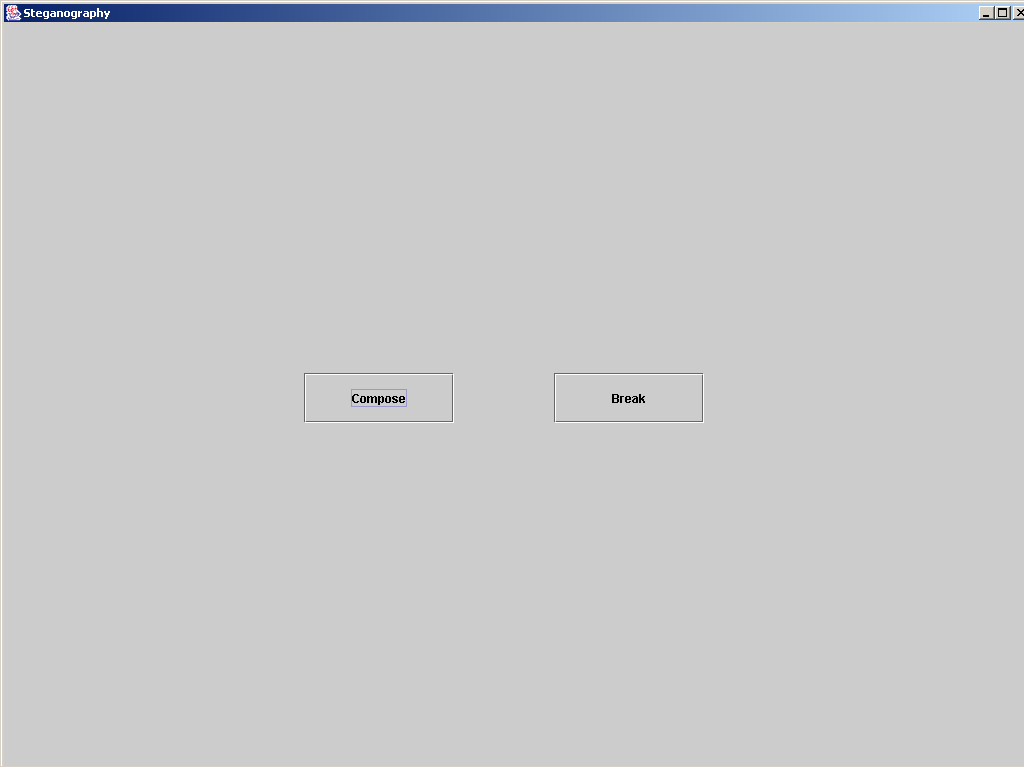
Combining all programs into a single application and testing its correct is called as Integration testing. Even all programs work correctly they may give a false result when they work together. Integration is very important to get the completed result.

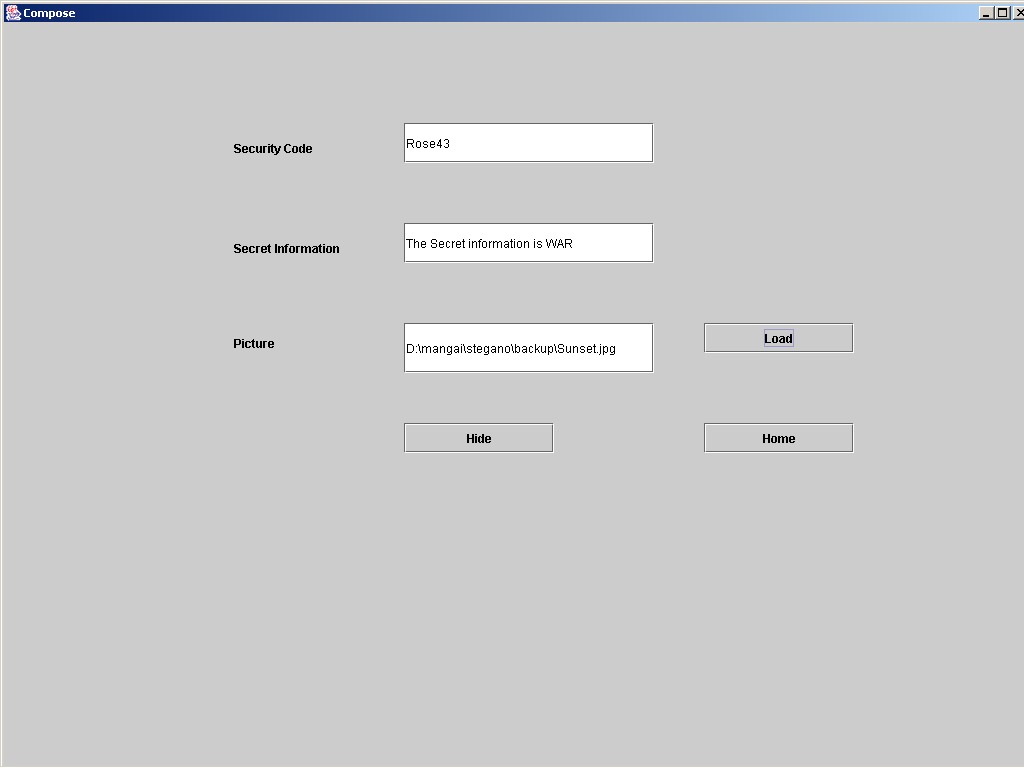
System Testing:

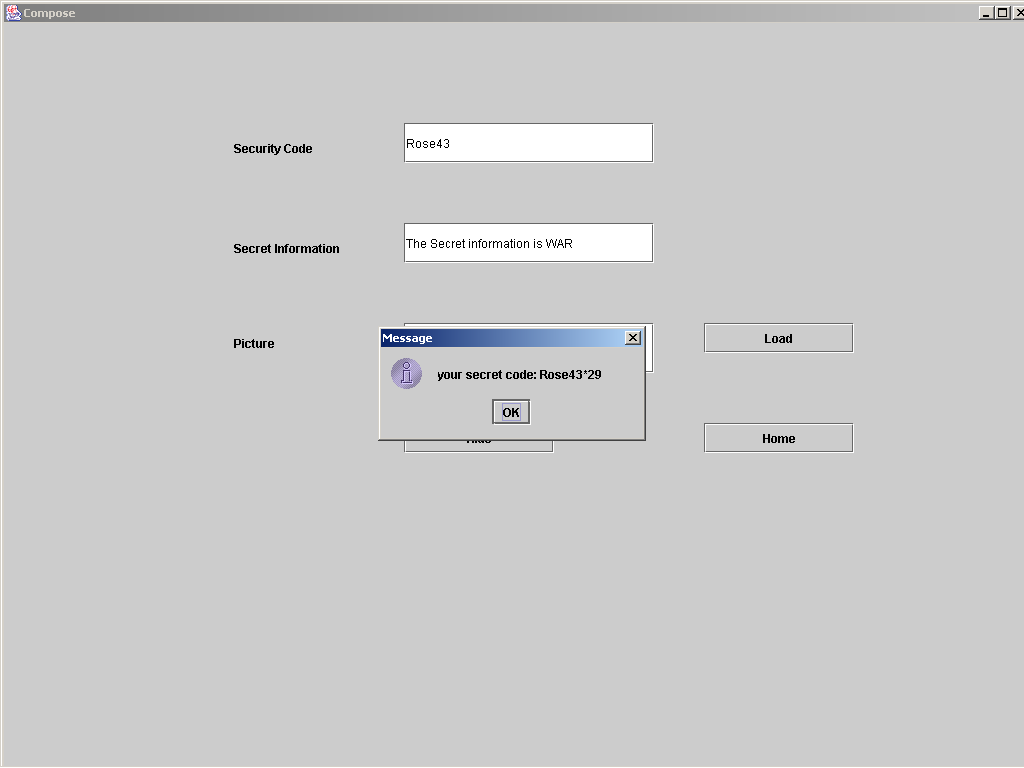
System testing means testing the whole system at once. By giving different inputs to the system we can check its correctness. For all inputs the system should produce correct result.

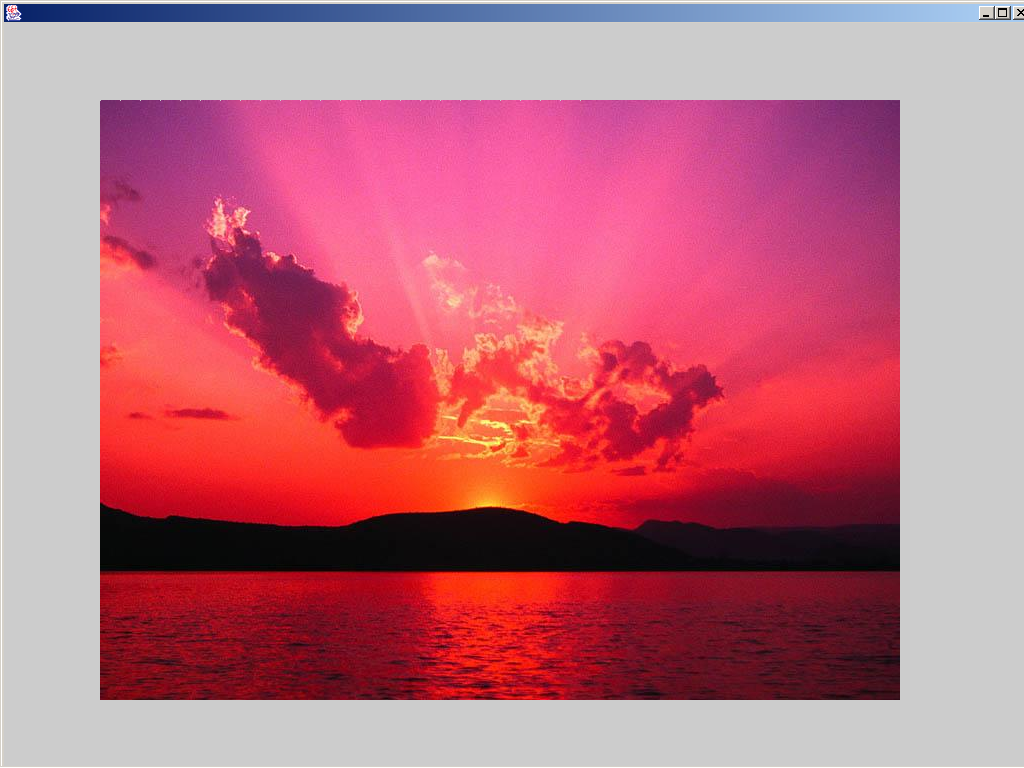
**Chapter 7**

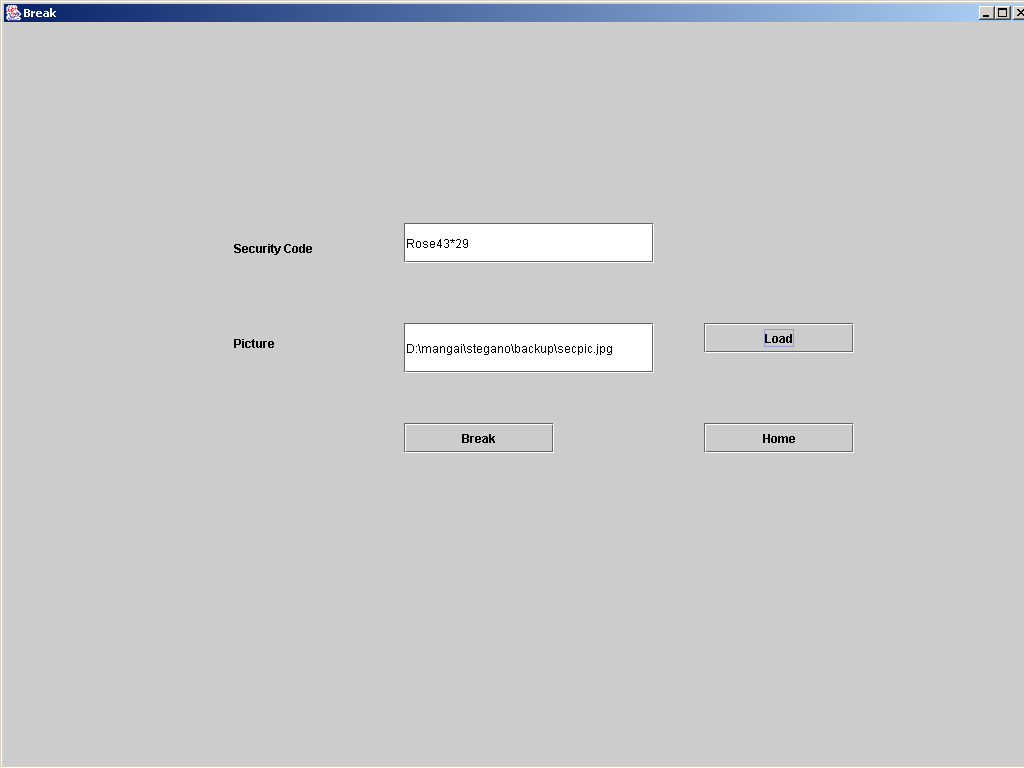
**RESULTS**

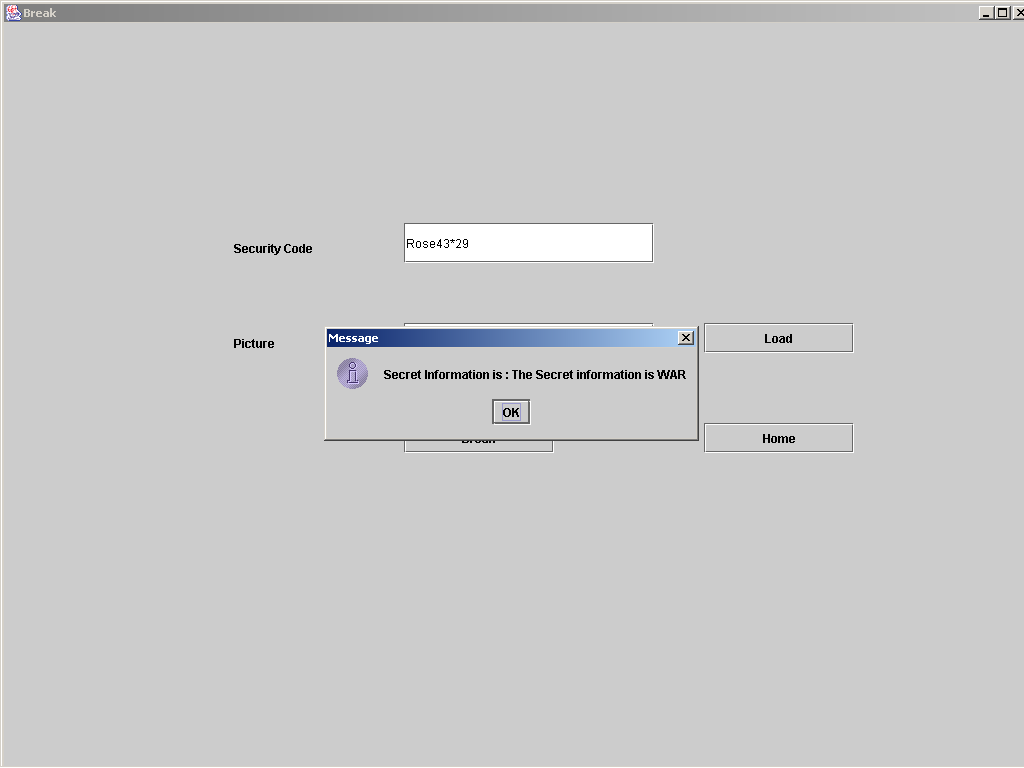












**Chapter 8**

**CONCLUSION**

This project provides a GUI, a user friendlier system, where secret information can easily be hided with in a picture file. It attains all java futures. It is platform independent so that it can be used in any OS. Thus secret information can be transferred to the intended user without giving any suspicion to the unintended user.

**Chapter 9**

**BIBLIOGRAPHY**

1. David & Deitel(1999), Java How to program Introducing Swing,Prentice Hall.
2. Roger S.Perssman,Software Engg A Practitioner’s Approach Fifth Edition-McGraww Hill International Edition,Software Engineering Series.
3. The Complete Reference JSP2.0,Tata McGraw-Hill publishing Company Limited, Phil Hanna

**Web Sites**

****** http://www.javaranch.com

**** http://forum.java.sun.com

**** http://java.sun.com

**** http://www.javaworld.com

**Chapter 10**

**APPENDIX**

**CODINGS:**

Home.java

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class Home extends JFrame implements ActionListener

{

private JButton compose,breakmsg;

Home()

{

super("Steganography");

Container con=getContentPane();

con.setLayout(null);

compose=new JButton("Compose");

compose.addActionListener(this);

compose.setBounds(300,350,150,50);

breakmsg=new JButton("Break");

breakmsg.addActionListener(this);

breakmsg.setBounds(550,350,150,50);

con.add(compose);

con.add(breakmsg);

}

public void actionPerformed(ActionEvent ae)

{

if(ae.getSource()==compose)

{

this.dispose();

ComposePage cp=new ComposePage();

cp.setSize(1035,790);

cp.setVisible(true);

}

if(ae.getSource()==breakmsg)

{

this.dispose();

BreakPage bp=new BreakPage();

bp.setSize(1035,790);

bp.setVisible(true);

}

}

public static void main(String args[])

{

Home h=new Home();

h.setSize(1035,790);

h.setVisible(true);

}

}

ComposePage.java

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

import java.io.\*;

import java.awt.image.\*;

import com.sun.image.codec.jpeg.\*;

public class ComposePage extends JFrame implements ActionListener

{

private JLabel code\_label,secret\_label,picture\_label;

private JTextField code\_text,secret\_text,picture\_text;

private JButton picture\_load\_button,hide\_button,home\_button;

String filepath="",secret\_code="",secret\_info="",user\_key="";

Container con=null;

JLabel jl;

byte img\_byte[]=new byte[6000];

FileDialog fd;

COMPOSE:

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

import java.io.\*;

import java.awt.image.\*;

import com.sun.image.codec.jpeg.\*;

public class ComposePage extends JFrame implements ActionListener

{

private JLabel code\_label,secret\_label,picture\_label;

private JTextField code\_text,secret\_text,picture\_text;

private JButton picture\_load\_button,hide\_button,home\_button;

String filepath="",secret\_code="",secret\_info="",user\_key="";

Container con=null;

JLabel jl;

byte img\_byte[]=new byte[6000];

FileDialog fd;

//////// Variables for creating an image from an integer array ///////////////////////////

Image img;

Dimension d;

int iw,ih;

int w=10,h=10;

int pix[];

int hist[]=new int[256];

int t[];

int max\_hist=0;

boolean ok;

static Image newimg;

int key,k;

ComposePage()

{

super("Compose");

con=getContentPane();

con.setLayout(null);

code\_label=new JLabel("Security Code");

code\_label.setBounds(230,100,150,50);

code\_text=new JTextField(200);

code\_text.setBounds(400,100,250,40);

secret\_label=new JLabel("Secret Information");

secret\_label.setBounds(230,200,150,50);

secret\_text=new JTextField(200);

secret\_text.setBounds(400,200,250,40);

picture\_label=new JLabel("Picture");

picture\_label.setBounds(230,300,250,40);

picture\_text=new JTextField(200);

picture\_text.setBounds(400,300,250,50);

picture\_load\_button=new JButton("Load");

picture\_load\_button.setBounds(700,300,150,30);

picture\_load\_button.addActionListener(this);

hide\_button=new JButton("Hide");

hide\_button.setBounds(400,400,150,30);

hide\_button.addActionListener(this);

home\_button=new JButton("Home");

home\_button.setBounds(700,400,150,30);

home\_button.addActionListener(this);

jl=new JLabel();

jl.setBounds(700,500,150,30);

fd=new FileDialog(new JFrame());

con.add(code\_label);

con.add(code\_text);

con.add(secret\_label);

con.add(secret\_text);

con.add(picture\_label);

con.add(picture\_text);

con.add(picture\_load\_button);

con.add(hide\_button);

con.add(home\_button);

//con.add(jl);

}

public void actionPerformed(ActionEvent ae)

{

if(ae.getSource()==picture\_load\_button)

{

fd.setVisible(true);

filepath=fd.getDirectory()+fd.getFile();

picture\_text.setText(filepath);

}else if(ae.getSource()==hide\_button)

{

int starflag=0;

secret\_code=code\_text.getText();

for(int i=0;i<secret\_code.length();i++)

{

if(secret\_code.charAt(i)=='\*')

{

starflag=1;

}

}

if(starflag==0)

{

secret\_info=secret\_text.getText();

user\_key=secret\_code+"\*"+new String(""+secret\_info.length());

System.out.println("user key :"+user\_key);

String secret\_code\_info=user\_key+"\*"+secret\_info+"\*";

byte secret\_byte\_array[]=secret\_code\_info.getBytes();

int secret\_int\_array[]=new int[secret\_byte\_array.length];

try{

if(filepath.equals("") && (secret\_text.getText()).equals(""))

JOptionPane.showMessageDialog(null,"image and secret info are empty. enter them");

else if(secret\_info.length()==0 && filepath.length()>0)

JOptionPane.showMessageDialog(null,"enter secret info");

else if(filepath.length()==0 && (secret\_text.getText()).length()>0)

JOptionPane.showMessageDialog(null,"load an image");

else

{

ImageIcon ic=new ImageIcon(filepath);

img=ic.getImage();

iw=img.getWidth(null);

ih=img.getHeight(null);

pix=new int[iw\*ih];

t=new int[iw\*ih];

PixelGrabber pg=new PixelGrabber(img,0,0,iw,ih,pix,0,iw);

ColorModel cm=pg.getColorModel();

int ww=pg.getWidth();

int hh=pg.getHeight();

pg.grabPixels();

key=secret\_byte\_array.length;

int k=key;

int j=0;

for(int i=0;i<pix.length;i++)

{

if((i%20)==0 && k>0)

{

secret\_int\_array[j]=(int)secret\_byte\_array[j];

System.out.println("user key :"+secret\_int\_array[j]);

pix[i]=secret\_int\_array[j];

j++;

k--;

}

}

newimg =con.createImage(new MemoryImageSource(ww,hh,cm,pix, 0, ww));

jl.setIcon(new ImageIcon(newimg));

JOptionPane.showMessageDialog(null,"your secret code: "+user\_key+"");

MediaTracker mediaTracker = new MediaTracker(new Container());

mediaTracker.addImage(newimg, 0);

mediaTracker.waitForID(0);

int thumbWidth = 400;//Integer.parseInt(400);

int thumbHeight = 400;//Integer.parseInt(400);

double thumbRatio = (double)thumbWidth / (double)thumbHeight;

int imageWidth = newimg.getWidth(null);

int imageHeight = newimg.getHeight(null);

double imageRatio = (double)imageWidth / (double)imageHeight;

if (thumbRatio < imageRatio)

{

thumbHeight = (int)(thumbWidth / imageRatio);

}

else

{

thumbWidth = (int)(thumbHeight \* imageRatio);

}

// draw original image to thumbnail image object and

// scale it to the new size on-the-fly

BufferedImage thumbImage = new BufferedImage(newimg.getWidth(null), newimg.getHeight(null), BufferedImage.TYPE\_INT\_RGB);

Graphics2D graphics2D = thumbImage.createGraphics();

graphics2D.setRenderingHint(RenderingHints.KEY\_INTERPOLATION,

RenderingHints.VALUE\_INTERPOLATION\_BILINEAR);

graphics2D.drawImage(newimg, 0, 0, newimg.getWidth(null), newimg.getHeight(null), null);

// save thumbnail image to OUTFILE

File f=new File("secpic.jpg");

BufferedOutputStream out = new BufferedOutputStream(new

FileOutputStream(f));

JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(out);

JPEGEncodeParam param = encoder.

getDefaultJPEGEncodeParam(thumbImage);

int quality = 80;//Integer.parseInt(args[4]);

quality = Math.max(0, Math.min(quality, 100));

param.setQuality((float)quality / 100.0f, false);

encoder.setJPEGEncodeParam(param);

encoder.encode(thumbImage);

out.close();

System.out.println("Done.");

test t=new test(newimg);

t.setSize(1035,790);

t.setVisible(true);

}

}catch(Exception e)

{

System.out.println(e);

}

}else

JOptionPane.showMessageDialog(null,"Do not enter '\*' in secrect code");

}else

{

this.dispose();

Home h=new Home();

h.setSize(1035,790);

h.setVisible(true);

}

}

public static void main(String args[])

{

ComposePage cp=new ComposePage();

cp.setSize(1035,740);

cp.setVisible(true);

}

}