**MainClient.java**

import javax.swing.\*;

import javax.swing.border.TitledBorder;

import java.awt.\*;

import java.awt.event.\*;

import org.jvnet.substance.\*;

public class MainClient extends WindowAdapter implements ActionListener

{

private JFrame mainFrame;

private JMenuBar menuBar;

private JMenu menuFile, menuEdit, menuView, menuHelp, menuLookAndFeel;

private JMenuItem mnuExit, mnuEmbedMessage, mnuEmbedFile, mnuHelp, mnuAbout;

private JMenuItem mnuRetrieveMessage, mnuRetrieveFile, mnuModifyMaster;

private JRadioButtonMenuItem mnuTonicFeel, mnuMetalFeel, mnuMotifFeel, mnuWindowsFeel;

private ButtonGroup lookAndFeelButtonGroup;

private JLabel lblBg;

private JPanel mainPanel, panelAbout, panelButtons;

private JLabel lblLogo;

private JLabel lblFiller[], lblName, lblEmail, lblPhone;

private GridBagLayout gbl;

private GridBagConstraints gbc;

private MyJButton btnEmbedFile, btnRetrieveFile, btnEmbedMessage, btnRetrieveMessage;

private MyJButton btnHelp, btnAbout;

private BackEndHandler back;

public MainClient()

{

mainFrame= new JFrame(" CHEATING PREVENTION IN VISUAL CRYPTOGRAPHY ");

mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

mainFrame.addWindowListener(this);

mainFrame.setLayout(null);

mnuExit= new MyJMenuItem("Exit", 1, 'x');

mnuEmbedMessage= new MyJMenuItem("Hide Message", 6, 'm');

mnuEmbedFile= new MyJMenuItem("Hide Data File", 7, 'i');

mnuRetrieveMessage= new MyJMenuItem("Retrieve Message", 0, 'r');

mnuRetrieveFile= new MyJMenuItem("Retrieve Data File", 2, 't');

mnuModifyMaster= new MyJMenuItem("Modify Master file settings", 2, 'd');

mnuHelp= new MyJMenuItem("Help", 0, 'h');

mnuAbout= new MyJMenuItem("About Me", 0, 'a');

mnuTonicFeel= new MyJRadioButtonMenuItem("Plastic XP", 8, 'x');

mnuMetalFeel= new MyJRadioButtonMenuItem("Metal", 0, 'm');

mnuMotifFeel= new MyJRadioButtonMenuItem("Motif", 2, 't');

mnuWindowsFeel= new MyJRadioButtonMenuItem("Windows", 0, 'w');

RadioListener radioListener= new RadioListener();

mnuTonicFeel.addItemListener(radioListener);

mnuMetalFeel.addItemListener(radioListener);

mnuMotifFeel.addItemListener(radioListener);

mnuWindowsFeel.addItemListener(radioListener);

mnuTonicFeel.setSelected(true);

lookAndFeelButtonGroup= new ButtonGroup();

lookAndFeelButtonGroup.add(mnuTonicFeel);

lookAndFeelButtonGroup.add(mnuMetalFeel);

lookAndFeelButtonGroup.add(mnuMotifFeel);

lookAndFeelButtonGroup.add(mnuWindowsFeel);

mnuEmbedMessage.addActionListener(this);

mnuEmbedFile.addActionListener(this);

mnuRetrieveMessage.addActionListener(this);

mnuRetrieveFile.addActionListener(this);

mnuModifyMaster.addActionListener(this);

mnuExit.addActionListener(this);

mnuHelp.addActionListener(this);

mnuAbout.addActionListener(this);

menuFile= new MyJMenu("File", 0, 'f');

menuFile.add(mnuEmbedMessage);

menuFile.add(mnuEmbedFile);

menuFile.add(mnuRetrieveMessage);

menuFile.add(mnuRetrieveFile);

menuFile.add(mnuExit);

menuEdit= new JMenu("Edit");

menuEdit.add(mnuModifyMaster);

menuLookAndFeel= new MyJMenu("Look and Feel...", 0, 'l');

menuLookAndFeel.add(mnuTonicFeel);

menuLookAndFeel.add(mnuMetalFeel);

menuLookAndFeel.add(mnuMotifFeel);

menuLookAndFeel.add(mnuWindowsFeel);

menuView= new MyJMenu("View", 0, 'v');

menuView.add(menuLookAndFeel);

menuHelp= new MyJMenu("Help", 0, 'h');

menuHelp.add(mnuHelp);

menuHelp.add(mnuAbout);

menuBar= new JMenuBar();

menuBar.add(menuFile);

menuBar.add(menuView);

menuBar.add(menuHelp);

mainFrame.setJMenuBar(menuBar);

mainPanel= new JPanel();

panelAbout= new JPanel();

panelButtons= new JPanel();

lblFiller= new JLabel[4];

for(int i=0; i<4; i++)

lblFiller[i]= new JLabel(" ");

gbl= new GridBagLayout();

gbc= new GridBagConstraints();

panelAbout.setLayout(gbl);

panelAbout.setBackground(Color.white);

Color myColor= new Color(50, 153, 237);

Font arialFont= new Font("Arial", Font.PLAIN, 14);

Font myFont= new Font("Monotype Corsiva", Font.PLAIN, 18);

lblName= new MyJLabel("", myFont, Color.blue, Color.white);

lblEmail= new MyJLabel("", arialFont, myColor, Color.white);

lblPhone= new JLabel(" ");

/\*gbc.gridx= 1; gbc.gridy= 1; gbl.setConstraints(lblName, gbc);

panelAbout.add(lblName);

gbc.gridx= 2; gbc.gridy= 2; gbl.setConstraints(lblEmail, gbc);

panelAbout.add(lblEmail);

gbc.gridx= 3; gbc.gridy= 3; gbl.setConstraints(lblPhone, gbc);

panelAbout.add(lblPhone);\*/

panelButtons.setBackground(Color.white);

gbl= new GridBagLayout();

//panelButtons.setLayout(gbl);

panelButtons.setLayout(null);

panelButtons.setBorder(new TitledBorder("Supported operations"));

lblLogo= new JLabel(new ImageIcon("Images/title.png"));

btnEmbedMessage= new MyJButton("Images/EncodeMsg", "Images/EncodeMsgHover");//

btnEmbedFile= new MyJButton("Images/EncodeFile", "Images/EncodeFileHover");

btnRetrieveMessage= new MyJButton("Images/RetrieveMessage", "Images/RetrieveMessageHover");

btnRetrieveFile= new MyJButton("Images/RetrieveFile", "Images/RetrieveFileHover");

btnHelp= new MyJButton("Images/Help.png", "Images/HelpHover.png");

btnAbout= new MyJButton("Images/About.png", "Images/AboutHover.png");

lblBg = new JLabel(new ImageIcon("Images/steg.jpg"));

lblBg.setBounds(450,200,595,300);

mainFrame.add(lblBg);

lblLogo.setBounds(250,20,1500,200);

mainFrame.add(lblLogo);

btnEmbedMessage.setBounds(200,150,180,50);

mainFrame.add(btnEmbedMessage);

btnEmbedFile.setBounds(200,220,180,50);

mainFrame.add(btnEmbedFile);

btnRetrieveMessage.setBounds(200,290,180,50);

mainFrame.add(btnRetrieveMessage);

btnRetrieveFile.setBounds(200,360,180,50);

mainFrame.add(btnRetrieveFile);

//btnHelp.setBounds(200,430,180,50);

//mainFrame.add(btnHelp);

//btnAbout.setBounds(200,500,180,50);

//mainFrame.add(btnAbout);

btnEmbedMessage.addActionListener(this);

btnEmbedFile.addActionListener(this);

btnRetrieveMessage.addActionListener(this);

btnRetrieveFile.addActionListener(this);

//btnHelp.addActionListener(this);

//btnAbout.addActionListener(this);

/\*gbc.weightx= 4; gbc.weighty= 2; gbc.fill= gbc.BOTH;

gbc.gridx= 6; gbc.gridy= 1; gbl.setConstraints(lblFiller[0], gbc);

lblFiller[0].setBounds(40,50,140,130);\*/

//panelButtons.add(lblFiller[0]);

/\*gbc.weightx= 1; gbc.weighty= 1; gbc.fill= gbc.NONE;

gbc.gridx= 3; gbc.gridy= 3; gbl.setConstraints(btnHelp, gbc);

btnHelp.setBounds(30,40,100,100);

panelButtons.add(btnHelp);\*/

/\*gbc.gridx= 5; gbl.setConstraints(btnAbout, gbc);

btnAbout.setBounds(30,60,50,50);

panelButtons.add(btnAbout);\*/

/\*gbc.fill = gbc.BOTH;

gbc.gridx= 1; gbc.weighty= 2; gbc.gridy= 4; gbl.setConstraints(lblFiller[1], gbc);

panelButtons.add(lblFiller[1]);\*/

/\*gbc.fill= gbc.NONE;

gbc.gridx= 2; gbc.weighty= 1; gbc.gridy= 6; gbl.setConstraints(btnEmbedMessage, gbc);

panelButtons.add(btnEmbedMessage);\*/

/\*gbc.gridx= 4; gbl.setConstraints(btnRetrieveMessage, gbc);

panelButtons.add(btnRetrieveMessage);\*/

/\*gbc.fill = gbc.BOTH;

gbc.gridx= 6; gbc.weighty= 2; gbc.gridy= 7; gbl.setConstraints(lblFiller[2], gbc);

panelButtons.add(lblFiller[2]);\*/

/\*gbc.fill= gbc.NONE;

gbc.gridx= 3; gbc.weighty= 1; gbc.gridy= 9; gbl.setConstraints(btnEmbedFile, gbc);

panelButtons.add(btnEmbedFile);\*/

/\*gbc.gridx= 5; gbl.setConstraints(btnRetrieveFile, gbc);

panelButtons.add(btnRetrieveFile);\*/

/\*gbl= new GridBagLayout();

//mainPanel.setLayout(gbl);

mainPanel.setLayout(null);

mainPanel.setBounds(0,0,800,600);

mainPanel.setBackground(Color.white);\*/

/\*gbc.anchor= gbc.CENTER;

gbc.gridx= 1; gbc.gridy= 1; gbc.weighty= 2; gbc.fill= gbc.VERTICAL;

gbl.setConstraints(lblLogo, gbc);

mainPanel.add(lblLogo);\*/

/\*gbc.gridy= 3; gbc.weighty= 2;

gbl.setConstraints(panelAbout, gbc);

mainPanel.add(panelAbout);\*/

/\*gbc.gridy= 5; gbc.weighty= 1;

gbl.setConstraints(panelButtons, gbc);

mainPanel.add(panelButtons);\*/

/\*gbc.gridy= 6; gbc.weighty= 2;

gbl.setConstraints(lblFiller[3], gbc);

mainPanel.add(lblFiller[3]);\*/

//JPanel tempPanel= (JPanel) mainFrame.getContentPane();

//tempPanel.add(mainPanel);

//tempPanel.add(mainPanel, BorderLayout.CENTER);

//tempPanel.add(new MyJLabel(" ", Color.black, Color.darkGray), BorderLayout.SOUTH);

Dimension d= Toolkit.getDefaultToolkit().getScreenSize();

mainFrame.setSize(d.width, (int) (d.height-(d.height\*.03)));

//mainFrame.setSize(800,600);

mainFrame.setResizable(false);

mainFrame.setVisible(true);

}

public void actionPerformed(ActionEvent e)

{

Object source= e.getSource();

if(source== mnuEmbedMessage || source== btnEmbedMessage)

{

back= new BackEndHandler(this, BackEndHandler.EMBED\_MESSAGE);

back.start();

}

if(source== mnuRetrieveMessage || source== btnRetrieveMessage)

{

back= new BackEndHandler(this, BackEndHandler.RETRIEVE\_MESSAGE);

back.start();

}

if(source== mnuEmbedFile || source== btnEmbedFile )

{

back= new BackEndHandler(this, BackEndHandler.EMBED\_FILE);

back.start();

}

if(source== mnuRetrieveFile || source== btnRetrieveFile )

{

back= new BackEndHandler(this, BackEndHandler.RETRIEVE\_FILE);

back.start();

}

if(source== mnuModifyMaster)

{

back= new BackEndHandler(this, BackEndHandler.EDIT\_MASTER);

back.start();

}

if(source== mnuHelp || source==btnHelp)

Steganograph.showHelpDialog();

if(source== mnuAbout || source== btnAbout)

Steganograph.showAboutDialog();

if(source== mnuExit)

{

int result= JOptionPane.showConfirmDialog(mainFrame, "Are you sure that you want to close", "Confirm Exit", JOptionPane.YES\_NO\_OPTION);

if(result== JOptionPane.YES\_OPTION)

{

JOptionPane.showMessageDialog(mainFrame, "Thanks for using.", "Our system", JOptionPane.INFORMATION\_MESSAGE);

System.exit(0);

}

}

}

public void windowClosing(WindowEvent w)

{

JOptionPane.showMessageDialog(mainFrame, "Thanks for using", "Our System", JOptionPane.INFORMATION\_MESSAGE);

}

private class RadioListener implements ItemListener

{

public void itemStateChanged(ItemEvent e)

{

JMenuItem item= (JMenuItem) e.getSource();

try

{

if(item== mnuTonicFeel && mnuTonicFeel.isSelected())

UIManager.setLookAndFeel("org.jvnet.substance.SubstanceLookAndFeel");

if(item== mnuMetalFeel && mnuMetalFeel.isSelected())

UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

if(item== mnuMotifFeel && mnuMotifFeel.isSelected())

UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");

if(item== mnuWindowsFeel && mnuWindowsFeel.isSelected())

UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

SwingUtilities.updateComponentTreeUI(mainFrame);

Steganograph.updateUserInterface();

}

catch(Exception ex)

{

JOptionPane.showMessageDialog(mainFrame, "Oops!!\n"+ "Unable to load "+ item.getText()+ " Look and feel.", "Warning!", JOptionPane.WARNING\_MESSAGE);

}

}

}

// Main method

/\* public static void main(String args[])

{

new MainClient();

}\*/

}

**Stegnography.java**

import java.io.\*;

import java.nio.ByteBuffer;

import java.util.zip.\*;

import javax.crypto.\*;

import javax.crypto.spec.\*;

import javax.swing.\*;

import javax.swing.event.\*;

import java.awt.\*;

import java.awt.event.\*;

public class Steganograph

{

public static final String VERSION= "2.0.0";

public static final byte[] VERSION\_BYTE= {'2','0','0'};

public static final int OFFSET\_JPG= 3;

public static final int OFFSET\_PNG= 42;

public static final int OFFSET\_GIF\_BMP\_TIF= 32;

public static final short HEADER\_LENGTH= 15\* 4;

public static final byte UUM= 0;

public static final byte UUF= 1;

public static final byte UEM= 2;

public static final byte UEF= 3;

public static final byte CUM= 4;

public static final byte CUF= 5;

public static final byte CEM= 6;

public static final byte CEF= 7;

private static Cipher cipher;

private static SecretKeySpec spec;

private static String masterExtension, message;

private static AboutFrame about= new AboutFrame();

private static File masterFile;

private static byte features;

private static int inputFileSize;

private static int i, j, inputOutputMarker, messageSize, tempInt;

private static short compressionRatio= 0, temp;

private static byte byte1, byte2, byte3, byteArrayIn[];

private static ByteArrayOutputStream byteOut;

private Steganograph()

{

System.out.println("Steganograph "+ VERSION+ " ready...");

}

public static String getMessage()

{

return message;

}

public static boolean embedMessage(File masterFile, File outputFile, String msg, int compression, String password)

{

if(msg==null)

{

message= "Message is empty";

return false;

}

if(msg.length()<1)

{

message= "Message is empty";

return false;

}

if(password!= null && password.length()<8)

{

message= "Password should be minimum of 8 Characters";

return false;

}

messageSize= msg.length();

if(compression!= -1)

{

if(compression< 0) compression= 0;

if(compression>9) compression= 9;

if(password== null) features= CUM;

else features= CEM;

}

else

{

if(password== null) features= UUM;

else features= UEM;

}

try

{

byteOut= new ByteArrayOutputStream();

byte []messageArray= msg.getBytes();

messageSize= messageArray.length;

inputFileSize= (int) masterFile.length();

byteArrayIn= new byte[inputFileSize];

DataInputStream in= new DataInputStream(new FileInputStream(masterFile));

in.read(byteArrayIn, 0, inputFileSize);

in.close();

String fileName= masterFile.getName();

masterExtension= fileName.substring(fileName.length()-3, fileName.length());

if(masterExtension.equalsIgnoreCase("jpg"))

{

byteOut.write(byteArrayIn, 0, OFFSET\_JPG);

inputOutputMarker= OFFSET\_JPG;

}

else if(masterExtension.equalsIgnoreCase("png"))

{

byteOut.write(byteArrayIn, 0, OFFSET\_PNG);

inputOutputMarker= OFFSET\_PNG;

}

else

{

byteOut.write(byteArrayIn, 0, OFFSET\_GIF\_BMP\_TIF);

inputOutputMarker= OFFSET\_GIF\_BMP\_TIF;

}

byte tempByte[]= new byte[4];

for(i=24, j=0; i>=0; i-=8, j++)

{

tempInt= inputFileSize;

tempInt>>= i;

tempInt&= 0x000000FF;

tempByte[j]= (byte) tempInt;

}

embedBytes(tempByte);

byteOut.write(byteArrayIn, inputOutputMarker, inputFileSize- inputOutputMarker);

inputOutputMarker= inputFileSize;

writeBytes(VERSION\_BYTE);

writeBytes(new byte[]{features});

if(features== CUM || features== CEM)

{

ByteArrayOutputStream arrayOutputStream= new ByteArrayOutputStream();

ZipOutputStream zOut= new ZipOutputStream(arrayOutputStream);

ZipEntry entry= new ZipEntry("MESSAGE");

zOut.setLevel(compression);

zOut.putNextEntry(entry);

zOut.write(messageArray, 0, messageSize);

zOut.closeEntry();

zOut.finish();

zOut.close();

messageArray= arrayOutputStream.toByteArray();

compressionRatio= (short) ((double)messageArray.length / (double)messageSize \* 100.0);

messageSize= messageArray.length;

}

writeBytes(new byte[]{(byte) compressionRatio});

if(features== UEM || features== CEM)

{

Cipher cipher= Cipher.getInstance("DES");

SecretKeySpec spec= new SecretKeySpec(password.substring(0, 8).getBytes(), "DES");

cipher.init(Cipher.ENCRYPT\_MODE, spec);

messageArray= cipher.doFinal(messageArray);

messageSize= messageArray.length;

}

tempByte= new byte[4];

for(i=24, j=0; i>=0; i-=8, j++)

{

tempInt= messageSize;

tempInt>>= i;

tempInt&= 0x000000FF;

tempByte[j]= (byte) tempInt;

}

writeBytes(tempByte);

writeBytes(messageArray);

DataOutputStream out= new DataOutputStream(new FileOutputStream(outputFile));

byteOut.writeTo(out);

out.close();

}

catch(EOFException e)

{

}

catch(Exception e)

{

message= "Oops!!\nError: "+ e.toString();

e.printStackTrace();

return false;

}

message= "Message embedded successfully in file '"+ outputFile.getName()+ "'.";

return true;

}

public static String retrieveMessage(SteganoInformation info, String password)

{

String messg= null;

features= info.getFeatures();

try

{

masterFile= info.getFile();

byteArrayIn= new byte[(int) masterFile.length()];

DataInputStream in= new DataInputStream(new FileInputStream(masterFile));

in.read(byteArrayIn, 0, (int)masterFile.length());

in.close();

messageSize= info.getDataLength();

if(messageSize<=0)

{

message= "Unexpected size of message: 0.";

return("#FAILED#");

}

byte[] messageArray= new byte[messageSize];

inputOutputMarker= info.getInputMarker();

readBytes(messageArray);

if(features== CEM || features== UEM)

{

password= password.substring(0, 8);

byte passwordBytes[]= password.getBytes();

cipher= Cipher.getInstance("DES");

spec= new SecretKeySpec(passwordBytes, "DES");

cipher.init(Cipher.DECRYPT\_MODE, spec);

try

{

messageArray= cipher.doFinal(messageArray);

}

catch(Exception bp)

{

message= "Incorrent Password";

bp.printStackTrace();

return "#FAILED#";

}

messageSize= messageArray.length;

}

if(features== CUM || features== CEM)

{

ByteArrayOutputStream by= new ByteArrayOutputStream();

DataOutputStream out= new DataOutputStream(by);

ZipInputStream zipIn= new ZipInputStream(new ByteArrayInputStream(messageArray));

zipIn.getNextEntry();

byteArrayIn= new byte[1024];

while((tempInt= zipIn.read(byteArrayIn, 0, 1024))!= -1)

out.write(byteArrayIn, 0, tempInt);

zipIn.close();

out.close();

messageArray= by.toByteArray();

messageSize= messageArray.length;

}

messg= new String(SteganoInformation.byteToCharArray(messageArray));

}

catch(Exception e)

{

message= "Oops!!\n Error: "+ e;

e.printStackTrace();

return("#FAILED#");

}

message= "Message size: "+ messageSize+ " B";

return messg;

}

public static boolean embedFile(File masterFile, File outputFile, File dataFile, int compression, String password)

{

messageSize= (int) dataFile.length();

if(password!= null && password.length()<8)

{

message= "Password should be minimum of 8 Characters";

return false;

}

if(compression!= 0)

{

if(compression< 0) compression= 0;

if(compression>9) compression= 9;

if(password== null) features= CUF;

else features= CEF;

}

else

{

if(password== null) features= UUF;

else features= UEF;

}

inputFileSize= (int) masterFile.length();

try

{

byteOut= new ByteArrayOutputStream();

byteArrayIn= new byte[inputFileSize];

DataInputStream in= new DataInputStream(new FileInputStream(masterFile));

in.read(byteArrayIn, 0, inputFileSize);

in.close();

String fileName= masterFile.getName();

masterExtension= fileName.substring(fileName.length()-3, fileName.length());

if(masterExtension.equalsIgnoreCase("jpg"))

{

byteOut.write(byteArrayIn, 0, OFFSET\_JPG);

inputOutputMarker= OFFSET\_JPG;

}

else if(masterExtension.equalsIgnoreCase("png"))

{

byteOut.write(byteArrayIn, 0, OFFSET\_PNG);

inputOutputMarker= OFFSET\_PNG;

}

else

{

byteOut.write(byteArrayIn, 0, OFFSET\_GIF\_BMP\_TIF);

inputOutputMarker= OFFSET\_GIF\_BMP\_TIF;

}

byte tempByte[]= new byte[4];

for(i=24, j=0; i>=0; i-=8, j++)

{

tempInt= inputFileSize;

tempInt>>= i;

tempInt&= 0x000000FF;

tempByte[j]= (byte) tempInt;

}

embedBytes(tempByte);

byteOut.write(byteArrayIn, inputOutputMarker, inputFileSize- inputOutputMarker);

inputOutputMarker= inputFileSize;

writeBytes(VERSION\_BYTE);

writeBytes(new byte[]{features});

byte []fileArray= new byte[messageSize];

in= new DataInputStream(new FileInputStream(dataFile));

in.read(fileArray, 0, messageSize);

in.close();

if(features== CUF || features== CEF)

{

ByteArrayOutputStream arrayOutputStream= new ByteArrayOutputStream();

ZipOutputStream zOut= new ZipOutputStream(arrayOutputStream);

ZipEntry entry= new ZipEntry(dataFile.getName());

zOut.setLevel(compression);

zOut.putNextEntry(entry);

zOut.write(fileArray, 0, messageSize);

zOut.closeEntry();

zOut.finish();

zOut.close();

fileArray= arrayOutputStream.toByteArray();

compressionRatio= (short) ((double)fileArray.length / (double)messageSize \* 100.0);

messageSize= fileArray.length;

}

writeBytes(new byte[]{(byte) compressionRatio});

if(features== UEF || features== CEF)

{

Cipher cipher= Cipher.getInstance("DES");

SecretKeySpec spec= new SecretKeySpec(password.substring(0, 8).getBytes(), "DES");

cipher.init(Cipher.ENCRYPT\_MODE, spec);

fileArray= cipher.doFinal(fileArray);

messageSize= fileArray.length;

}

tempByte= new byte[4];

for(i=24, j=0; i>=0; i-=8, j++)

{

tempInt= messageSize;

tempInt>>= i;

tempInt&= 0x000000FF;

tempByte[j]= (byte) tempInt;

}

writeBytes(tempByte);

writeBytes(fileArray);

DataOutputStream out= new DataOutputStream(new FileOutputStream(outputFile));

byteOut.writeTo(out);

out.close();

}

catch(EOFException e)

{

}

catch(Exception e)

{

message= "Oops!!\nError: "+ e.toString();

e.printStackTrace();

return false;

}

message= "File '"+ dataFile.getName()+ "' embedded successfully in file '"+ outputFile.getName()+ "'.";

return true;

}

public static boolean retrieveFile(SteganoInformation info, String password, boolean overwrite)

{

File dataFile= null;

features= info.getFeatures();

try

{

masterFile= info.getFile();

byteArrayIn= new byte[(int) masterFile.length()];

DataInputStream in= new DataInputStream(new FileInputStream(masterFile));

in.read(byteArrayIn, 0, (int)masterFile.length());

in.close();

messageSize= info.getDataLength();

byte[] fileArray= new byte[messageSize];

inputOutputMarker= info.getInputMarker();

readBytes(fileArray);

if(messageSize<=0)

{

message= "Unexpected size of embedded file: 0.";

return false;

}

if(features== CEF || features== UEF)

{

password= password.substring(0, 8);

byte passwordBytes[]= password.getBytes();

cipher= Cipher.getInstance("DES");

spec= new SecretKeySpec(passwordBytes, "DES");

cipher.init(Cipher.DECRYPT\_MODE, spec);

try

{

fileArray= cipher.doFinal(fileArray);

}

catch(Exception bp)

{

message= "Incorrent Password";

bp.printStackTrace();

return false;

}

messageSize= fileArray.length;

}

if(features== CUF || features== CEF)

{

ByteArrayOutputStream by= new ByteArrayOutputStream();

DataOutputStream out= new DataOutputStream(by);

ZipInputStream zipIn= new ZipInputStream(new ByteArrayInputStream(fileArray));

ZipEntry entry= zipIn.getNextEntry();

dataFile= new File(entry.getName());

byteArrayIn= new byte[1024];

while((tempInt= zipIn.read(byteArrayIn, 0, 1024))!= -1)

out.write(byteArrayIn, 0, tempInt);

zipIn.close();

out.close();

fileArray= by.toByteArray();

messageSize= fileArray.length;

}

info.setDataFile(dataFile);

if(dataFile.exists() && !overwrite)

{

message= "File Exists";

return false;

}

DataOutputStream out= new DataOutputStream(new FileOutputStream(dataFile));

out.write(fileArray, 0, fileArray.length);

out.close();

}

catch(Exception e)

{

message= "Oops!!\n Error: "+ e;

e.printStackTrace();

return false;

}

message= "Retrieved file size: "+ messageSize+ " B";

return true;

}

private static void embedBytes(byte[] bytes)

{

int size= bytes.length;

for(int i=0; i< size; i++)

{

byte1= bytes[i];

for(int j=6; j>=0; j-=2)

{

byte2= byte1;

byte2>>= j;

byte2&= 0x03;

byte3= byteArrayIn[inputOutputMarker];

byte3&= 0xFC;

byte3|= byte2;

byteOut.write(byte3);

inputOutputMarker++;

}

}

}

private static void writeBytes(byte[] bytes)

{

int size= bytes.length;

for(int i=0; i< size; i++)

{

byteOut.write(bytes[i]);

inputOutputMarker++;

}

}

private static void retrieveBytes(byte[] bytes)

{

int size= bytes.length;

for(int i=0; i< size; i++)

{

byte1= 0;

for(int j=6; j>=0; j-=2)

{

byte2= byteArrayIn[inputOutputMarker];

inputOutputMarker++;

byte2&= 0x03;

byte2<<= j;

byte1|= byte2;

}

bytes[i]= byte1;

}

}

private static void readBytes(byte[] bytes)

{

int size= bytes.length;

for(int i=0; i< size; i++)

{

bytes[i]= byteArrayIn[inputOutputMarker];

inputOutputMarker++;

}

}

public static void showAboutDialog()

{

about.setDisplayed(true);

}

public static void updateUserInterface()

{

SwingUtilities.updateComponentTreeUI(about);

}

public static void showHelpDialog()

{

try {

Process pc = Runtime.getRuntime().exec("cmd.exe /c start main.html");

} catch (IOException ex) {

System.out.println();

}

}

private static class AboutFrame extends JFrame

{

private final Color GREEN= Color.green;

private final Color YELLOW= Color.yellow;

private final Color BLACK= Color.black;

private JLabel lblTitle, lblImage, lblName, lblName1, lblEmail, lblPhone;

private JLabel filler1, filler2, filler3;

private GridBagLayout gbl;

private GridBagConstraints gbc;

public AboutFrame()

{

setTitle("About Steganograph "+ Steganograph.VERSION);

filler1= new JLabel(" "); filler2= new JLabel(" "); filler3= new JLabel(" ");

lblTitle= new MyJLabel("Steganography Implementation & Desing By", new Font("Arial", Font.PLAIN, 15), YELLOW, BLACK);

lblName= new MyJLabel("\t\t\tJPInfotech ", new Font("Century Gothic", Font.BOLD, 14), GREEN, BLACK);

lblName1= new MyJLabel("\t\t\tjpinfotech@blogspot.com ", new Font("Century Gothic", Font.BOLD, 14), GREEN, BLACK);

lblEmail= new MyJLabel("", GREEN, BLACK);

lblPhone= new MyJLabel("", GREEN, BLACK);

if(new File("Images/About").exists())

lblImage= new JLabel(new ImageIcon("Images/Myself"));

else

{

lblImage= new MyJLabel("Steganograph "+ Steganograph.VERSION, new Font("Times new roman", Font.PLAIN, 50), Color.red, Color.black);

lblImage.setPreferredSize(new Dimension(222, 231));

}

JPanel panelAbout= new JPanel();

panelAbout.setBackground(BLACK);

panelAbout.setForeground(GREEN);

gbl= new GridBagLayout();

gbc= new GridBagConstraints();

panelAbout.setLayout(gbl);;

gbc.anchor= GridBagConstraints.CENTER;

gbc.weightx= 2;

gbc.weighty= 2;

gbc.gridx= 1; gbc.gridy= 1; gbl.setConstraints(lblTitle, gbc);

panelAbout.add(lblTitle);

gbc.gridy= 2; gbl.setConstraints(filler1, gbc);

panelAbout.add(filler1);

gbc.gridy= 3; gbl.setConstraints(lblName, gbc);

panelAbout.add(lblName);

gbc.gridy= 4; gbl.setConstraints(filler2, gbc);

panelAbout.add(filler2);

gbc.gridy= 5; gbl.setConstraints(lblName1, gbc);

panelAbout.add(lblName1);

gbc.gridy= 6; gbl.setConstraints(filler3, gbc);

panelAbout.add(filler3);

gbc.gridy= 7; gbl.setConstraints(lblPhone, gbc);

panelAbout.add(lblPhone);

panelAbout= UtilityOperations.createBorderedPanel(panelAbout, "About Steganograph", 3, 3);

JPanel panelUpper= new JPanel();

panelUpper.setBackground(BLACK);

panelUpper.setLayout(new FlowLayout());

panelUpper.add(lblImage);

panelUpper.add(new MyJLabel(" ", GREEN, BLACK));

panelUpper.add(panelAbout);

JPanel mainPanel= new JPanel();

mainPanel.setBackground(Color.black);

gbl= new GridBagLayout();

mainPanel.setLayout(gbl);

gbc.gridx= 1; gbc.gridy= 1; gbl.setConstraints(panelUpper, gbc);

mainPanel.add(panelUpper);

mainPanel= UtilityOperations.createBorderedPanel(mainPanel, 3, 2);

setContentPane(mainPanel);

Dimension d= Toolkit.getDefaultToolkit().getScreenSize();

int width= (int) (d.width\*2.1/3);

int height= (int) (d.height\*1.9)/3;

setSize(width, height);

setLocation((d.width- width)/2, (d.height- height)/2);

setResizable(false);

}

public void setDisplayed(boolean choice)

{

setVisible(choice);

}

}

}

class SteganoInformation

{

private File file;

private File dataFile= null;

private String starter;

private String version;

private byte features;

private short compressionRatio;

private int dataLength, temp;

private boolean isEster= false;

private byte byteArray[], name[], byte1, byte2;

private int inputMarker, i, j;

public File getFile() { return file; }

public int getInputMarker() { return inputMarker; }

public File getDataFile() { return dataFile; }

public String getVersion() { return version; }

public byte getFeatures() { return features; }

public short getCompressionRatio() { return compressionRatio; }

public int getDataLength() { return dataLength; }

public boolean isEster() { return isEster; }

public void setDataFile(File dataFile)

{

this.dataFile= dataFile;

}

private void retrieveBytes(byte[] bytes, byte[] array, int marker)

{

byteArray= array;

inputMarker= marker;

int size= bytes.length;

for(i=0; i< size; i++)

{

byte1= 0;

for(j=6; j>=0; j-=2)

{

byte2= byteArray[inputMarker];

inputMarker++;

byte2&= 0x03;

byte2<<= j;

byte1|= byte2;

}

bytes[i]= byte1;

}

}

private void retrieveBytes(byte[] bytes)

{

int size= bytes.length;

for(i=0; i< size; i++)

{

byte1= 0;

for(j=6; j>=0; j-=2)

{

byte2= byteArray[inputMarker];

inputMarker++;

byte2&= 0x03;

byte2<<= j;

byte1|= byte2;

}

bytes[i]= byte1;

}

}

private void readBytes(byte[] bytes, byte[] array, int marker)

{

byteArray= array;

inputMarker= marker;

int size= bytes.length;

for(i=0; i< size; i++)

{

bytes[i]= byteArray[inputMarker];

inputMarker++;

}

}

private void readBytes(byte[] bytes)

{

int size= bytes.length;

for(i=0; i< size; i++)

{

bytes[i]= byteArray[inputMarker];

inputMarker++;

}

}

public static char[] byteToCharArray(byte[] bytes)

{

int size= bytes.length, i;

char []chars= new char[size];

for(i=0; i<size; i++)

{

bytes[i]&= 0x7F;

chars[i]= (char) bytes[i];

}

return chars;

}

public SteganoInformation(File file)

{

this.file= file;

isEster= false;

if(!file.exists())

{

starter= null;

return;

}

if(file.getName().equals("Sec#x&y"))

{

isEster= true;

return;

}

byteArray= new byte[(int) file.length()];

try

{

DataInputStream in= new DataInputStream(new FileInputStream(file));

in.read(byteArray, 0, (int) file.length());

in.close();

}

catch(Exception e)

{

starter= null;

return;

}

name= new byte[4];

String fileName= file.getName();

String fileExtension= fileName.substring(fileName.length()-3, fileName.length());

if(fileExtension.equalsIgnoreCase("jpg"))

inputMarker= Steganograph.OFFSET\_JPG;

else if(fileExtension.equalsIgnoreCase("png"))

inputMarker= Steganograph.OFFSET\_PNG;

else

inputMarker= Steganograph.OFFSET\_GIF\_BMP\_TIF;

retrieveBytes(name, byteArray, inputMarker);

dataLength= 0;

for(i=24,j=0; i>=0; i-=8,j++)

{

temp= name[j];

temp&= 0x000000FF;

temp<<= i;

dataLength|= temp;

}

inputMarker= dataLength;

if(dataLength<0 || dataLength>file.length())

{

starter= "Invalid";

return;

}

else

starter= "MUJEEB";

byte versionArray[]= new byte[3];

readBytes(versionArray, byteArray, inputMarker);

char []versionTemp= byteToCharArray(versionArray);

char []ver= new char[5];

for(i=0, j=0; i<5; i++)

if(i== 1 || i== 3) ver[i]= '.';

else

{

ver[i]= versionTemp[j++];

}

version= new String(ver);

name= new byte[1];

readBytes(name);

features= name[0];

readBytes(name);

name[0]&= 0x7F;

compressionRatio= name[0];

name= new byte[4];

readBytes(name);

dataLength= 0;

for(i=24,j=0; i>=0; i-=8,j++)

{

temp= name[j];

temp&= 0x000000FF;

temp<<= i;

dataLength|= temp;

}

}

public boolean isValid()

{

if(starter.equals("MUJEEB"))

{

return true;

}

else

return false;

}

}

class HTMLFrame extends JFrame implements HyperlinkListener

{

JEditorPane editorPane;

JScrollPane scrollPane;

public HTMLFrame(String startURL, boolean isOnline)

{

super("Help - Steganograph "+ Steganograph.VERSION+ "");

editorPane= new JEditorPane();

editorPane.setEditable(false);

if(isOnline)

{

editorPane.setBackground(Color.white);

setTitle("Steganograph "+ Steganograph.VERSION+ " ");

}

editorPane.addHyperlinkListener(this);

scrollPane= new JScrollPane(editorPane);

try

{

editorPane.setPage(startURL);

getContentPane().add(scrollPane);

}

catch(Exception e)

{

JOptionPane.showMessageDialog(this, "Oops!! Error\n"+ e, "Error", JOptionPane.WARNING\_MESSAGE);

return;

}

Dimension d= Toolkit.getDefaultToolkit().getScreenSize();

setSize(d.width, d.height);

setVisible(true);

}

public void hyperlinkUpdate(HyperlinkEvent e)

{

if(e.getEventType()== HyperlinkEvent.EventType.ACTIVATED)

{

try

{

editorPane.setPage(e.getURL());

}

catch(Exception ex)

{

JOptionPane.showMessageDialog(this, "Oops!! Error\n"+ ex, "Error", JOptionPane.WARNING\_MESSAGE);

}

}

}

}

**LoginForm.java**

import java.io.\*;

import java.awt.\*;

import java.util.\*;

import java.awt.event.\*;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import javax.swing.\*;

import javax.swing.event.\*;

import org.jvnet.substance.\*;

public class LoginForm extends JFrame implements ActionListener

{

private JLabel lbluid=new JLabel("User Name: ");

private JLabel lblpwd=new JLabel("Password : ");

private JTextField txtuid=new JTextField(15);

private JPasswordField txtpwd=new JPasswordField(15);

private JButton btnok=new JButton("Login");

private JButton btnclose=new JButton("Log Out ");

private JPanel sp=new JPanel(),

np=new JPanel(),

cp=new JPanel();

Container cpane;

String uid=null,pwd=null;

static {

try {

SubstanceLookAndFeel.setCurrentWatermark("org.jvnet.substance.watermark.SubstanceMetalWallWatermark");

SubstanceLookAndFeel.setCurrentTheme("org.jvnet.substance.theme.SubstanceSunfireRedTheme");

SubstanceLookAndFeel.setCurrentGradientPainter("org.jvnet.substance.painter.SpecularGradientPainter");

SubstanceLookAndFeel.setCurrentButtonShaper("org.jvnet.substance.button.StandardButtonShaper");

UIManager.setLookAndFeel(new SubstanceLookAndFeel());

} catch (Exception e) {

// TODO: handle exception

e.printStackTrace();

}

}

public LoginForm()

{

setSize(320,200);

setLocation(350,200);

setResizable(false);

//sp.setBackground(Color.blue);

//np.setBackground(Color.gray);

//cp.setBackground(Color.cyan);

btnok.addActionListener(this);

btnclose.addActionListener(this);

lbluid.setBounds(10,10,100,25);

txtuid.setBounds(110,10,150,25);

lblpwd.setBounds(10,40,100,25);

txtpwd.setBounds(110,40,150,25);

JLabel title=new JLabel("Login");

np.add(title);

cp.setLayout(null);

cp.add(lbluid);cp.add(txtuid);

cp.add(lblpwd);cp.add(txtpwd);

sp.add(btnok);sp.add(btnclose);

cpane=getContentPane();

cpane.add(np,"North");

cpane.add(cp,"Center");

cpane.add(sp,"South");

}

public void actionPerformed(ActionEvent e)

{

Object cmd=e.getSource();

if (cmd.equals(btnok))

{

Statement st = null;

ResultSet rs = null;

try{

String auid=txtuid.getText();

String apwd=txtpwd.getText();

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/datahide","root","root");

st = con.createStatement();

String qry ="select \* from user where name='"+auid+"' AND password='"+apwd+"'";

rs = st.executeQuery(qry);

while(rs.next()){

if ((auid.equals(rs.getString("name"))) && (apwd.equals(rs.getString("password"))))

{

JOptionPane.showMessageDialog(null,"Successfully Login","Admin",JOptionPane.INFORMATION\_MESSAGE);

new MainClient();

this.dispose();

}

else

{

JOptionPane.showMessageDialog(null,"Invalid Login,Try Again","Admin",JOptionPane.INFORMATION\_MESSAGE);

}

}

}

catch(Exception e1){}

}

if (cmd.equals(btnclose))

{

System.exit(0);

}

}

public static void main(String arg[])

{

LoginForm frm=new LoginForm();

frm.setVisible(true);

}

}