The Sound of Encryption

New Mexico Adventures in
Supercomputing Challenge
Final Report
April 6, 2005

Team 26
Farmington High School

Team Members:
Caitlin McCarthy
Michael Blount
Zachary Blackwood
Nathaniel Ayoub

Teachers:
Shirley Maurer
Mike DeField

Mentor:
Christopher Hoppe

Table of Contents:

Executive Summary

The purpose of the project is to use the Java programming language to encrypt messages into MIDI files. To do this, first we had to understand the MIDI file. Through use of the JMusic java package, a MIDI files is manipulable by its components. The MIDI file is a score, which is comprised of parts. Each part contains one or more phrases, and each phrase contains notes. We were able to manipulate the notes to hold the ASCII values of characters. These characters came from an encrypted string of text. We used password-based encryption and implemented the Twofish encryption algorithm. We then added the muted phrase, comprised of the encrypted characters, to an existing part, and saved the new score as a MIDI file. We were then able to successfully extract and hide encrypted messages within MIDI files. We were able to make the program dynamic enough to support any MIDI file and allow the user to select whichever part and phrase to place the message in.

Introduction

In this project a Java program was created to hide and encrypt messages in a MIDI file using steganography. Steganography is the process of hiding one thing inside of another. Steganography has been used since the beginning of writing. It is used with invisible ink or with code words inside of a sentence, but in this project, stegonography is used with a MIDI file. Encryption is the converting of a message into a form that cannot be read. In the Java program the message is encrypted using the Twofish algorithm and a password. The encrypted message is placed into notes that make up the score of the MIDI file. After the message is encrypted into notes, it was then muted to further the steganography and make it theoretically unnoticeable when listening to the music.

A MIDI file, or the Musical Instrument Digital Interface, was chosen as the medium in which to hide the message. A MIDI file is made up of a score. The score is made up of parts, the parts are made up of phrases and the phrases are made up of notes. MIDI files contain only musical instruments. By using MIDI files, we were able to easily customize notes from the encrypted message. For the message to be hidden within the MIDI file, we had to convert it to numeric values. To produce these numeric values, we used the standard ASCII values. ASCII is the American Standard Code for Information Interchange. It is basically a table, which stores a standard numeric value for each character. These standard ASCII values were used to set the pitch of new notes in the new phrase we created within an existing part of a MIDI file. The phrase that contained the message was then muted out and started at the same time as another phrase, so that it was not noticeable.

This project was programmed using the Java programming language. Java is an object oriented programming language that was created around 1995 by Sun Microsystems. The original version of Java was called Oak, and was designed in the 1990s. It was designed to be able to control different consumer electronic devices using the same software. The Java programming language is platform independent, or can run from any type of personal computer, mainframe, or workstation.

Description

The scope of our project was limited to MIDI files and use of the open source JMusic Java library. We also limited the project to dealing with text as opposed to data.

Our project required use of Java SDK 1.4 or newer, Borland JBuilder 2005 Foundation, the unlimited strength encryption files from Sun Microsystems, the JMusic Java library, the BouncyCastle encryption provider Java library file, and MIDI files.

Method

To hide the message:

1. First, we parse the MIDI file by calling on the MidiParser of the JMusic package.
2. Then the user selects the part of the MIDI in which to hide the message.
3. The user then selects which phrase of the MIDI file to replace with the hidden message.
4. The selected phrase is then moved.
5. The message is encrypted using password-based encryption with either MD5 and DES algorithm, or SHAA and Twofish algorithm.
6. New notes are created, one for every character of the encrypted message.
7. These notes are then made into a phrase.
8. The phrase is placed where the selected phrase was and the rest of the phrases are moved one over.
9. The part is rewritten, this time containing the new phrase.
10. The part is replaced in the score.
11. The score is written as a MIDI file.

To retrieve the message:

1. The midi is parsed using the MidiParser of the JMusic package.
2. The user then selects which part of the MIDI to decode.
3. Then the user selects which phrase of the part to decode.
4. The user enters the password.
5. The pitches of the notes are extracted from the chosen phrase.
6. The pitches are converted into characters.
7. The message is then decrypted using the Twofish algorithm and the given password.
8. The message is displayed.

Results

The result of the program was a newly created MIDI file, which contained the encrypted message. There was a significant change in the music. The file size was also noticeably larger. We tried the program with many different MIDI files, but always ended up with a larger MIDI file, on account of the additional data, and some distortion to the music.

Conclusion

Although there was a significant change in the MIDI file, we believe that the program was still a success because it was able to successfully hide and retrieve a text message, which had been encrypted. We found that the size difference, although noticeable, is not significant because any song in MIDI format is available in numerous sizes. The sound difference is only a concern if the MIDI file is suspected as being abnormal.

Recommendation

We believe that if we used the straight JSound API supplied with the Java SDK, it is possible that we could have avoided the distortion to the music. We cannot find a reason why this distortion happens unless it has something to do with the open source JMusic Java library.

Significant Achievement

The most significant achievement of the project is that a message is encrypted and hid inside of a MIDI file using a Java program. By doing this, we have made use of a new medium in which steganography can be applied.

Acknowledgements

We would like to thank Sergei Piletsky of MIIK Ltd. of the Ukraine for providing an educational license of the JNI Wrapper program.

Christopher Hoppe of the US Patent and Trademark Office for being our mentor and helping with research.

Kristian Sandburg of Colorado University for explaining the discrete Fourier transform.

References

Churchhouse, Robert. Codes and ciphers : Julius Caesar, the Enigma, and the internet. Cambridge: Cambridge University Press, 2002.

Garms, Jess, et. al. Professional Java Security. Birmingham: Wrox, 2001.

Glatt, Jeff. The Beginnings of MIDI. 25 Oct. 2004 <http://www.borg.com/~jglatt/tutr/history.htm>.

John, Corinna. Steganography-Hiding Messages in MIDI Songs. The Code Project. 25 Oct. 2004 <http://www.thecodeproject.com/csharp/steganodotnet5.asp>.

Sierra, Kathy. Head First Java. Sebastopol: O'Reilly, 2003.

Singh, Simon. The Code Book. New York: Doubleday, 1999.

Code – MidiFrame.java

```
import java.io.*;
import java.util.*;
import javax.crypto.*;
import javax.crypto.spec.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

import jm.music.data.*;
import jm.util.*;
import sun.misc.*;



public class MidiFrame extends JFrame {
  JPanel contentPane;
  JMenuBar menuBar1 = new JMenuBar();
  JMenu menuFile = new JMenu();
  JMenuItem menuFileExit = new JMenuItem();
  JMenu menuHelp = new JMenu();
  JMenuItem menuHelpAbout = new JMenuItem();
  JToolBar toolBar = new JToolBar();
  JButton jButton1 = new JButton();
  JButton jButton2 = new JButton();
  JButton jButton3 = new JButton();
  ImageIcon image1;
  ImageIcon image2;
  ImageIcon image3;
  JLabel statusBar = new JLabel();
```

```
JScrollPane jScrollPane1 = new JScrollPane();
JTextArea jTextArea1 = new JTextArea();
JMenuItem jMenuItem2 = new JMenuItem();
JMenuItem jMenuItem3 = new JMenuItem();
JMenuItem jMenuItem4 = new JMenuItem();
JFileChooser jFileChooser1 = new JFileChooser();
String currFileName = null;  // Full path with filename.
boolean dirty = false;
Score score;
JSlider jSlider1 = new JSlider();
JPasswordField jPasswordField1 = new JPasswordField();
JSlider jSlider2 = new JSlider();
JSlider jSlider3 = new JSlider();
JButton jButton4 = new JButton();
JButton jButton5 = new JButton();
JButton jButton6 = new JButton();
Play play = new Play();
boolean playing = false;
private static int ITERATIONS = 1000;

/**
 * Construct the frame
 */
public MidiFrame() {
  enableEvents(AWTEvent.WINDOW_EVENT_MASK);
  try {
    jbInit();
    updateCaption();
  }
  catch(Exception e) {
    e.printStackTrace();
```

```
  }
}

/**
 * Component initialization
 *
 * @throws Exception exception
 */
private void jbInit() throws Exception  {
  image1 = new ImageIcon(MidiFrame.class.getResource("openFile.gif"));
  image2 = new ImageIcon(MidiFrame.class.getResource("closeFile.gif"));
  image3 = new ImageIcon(MidiFrame.class.getResource("help.gif"));
  contentPane = (JPanel) this.getContentPane();
  contentPane.setLayout(null);
  this.setSize(new Dimension(400, 300));
  this.setTitle("The Sound of Encryption");
  statusBar.setText(" ");
  statusBar.setBounds(new Rectangle(0, 285, 399, 15));
  menuFile.setText("File");
  menuFileExit.setText("Exit");
  menuFileExit.addActionListener(
    new MidiFrame_menuFileExit_ActionAdapter(this));
  menuHelp.setText("Help");
  menuHelpAbout.setText("About");
  menuHelpAbout.addActionListener(
    new MidiFrame_menuHelpAbout_ActionAdapter(this));
  jButton1.setIcon(image1);
  jButton1.addActionListener(new MidiFrame_jButton1_actionAdapter(this));
  jButton1.setToolTipText("Open File");
  jButton2.setIcon(image2);
  jButton2.addActionListener(new MidiFrame_jButton2_actionAdapter(this));
```

```
jButton2.setToolTipText("Save File");
jButton3.setIcon(image3);
jButton3.addActionListener(new MidiFrame_jButton3_actionAdapter(this));
jButton3.setToolTipText("About");
jTextArea1.setLineWrap(true);
jTextArea1.setWrapStyleWord(true);
jTextArea1.setBackground(Color.white);
jMenuItem2.setText("Open");
jMenuItem2.addActionListener(
  new MidiFrame_jMenuItem2_actionAdapter(this));
jMenuItem3.setText("Save");
jMenuItem3.addActionListener(
  new MidiFrame_jMenuItem3_actionAdapter(this));
jMenuItem4.setText("Save As");
jMenuItem4.addActionListener(
  new MidiFrame_jMenuItem4_actionAdapter(this));
jScrollPane1.setBounds(new Rectangle(0, 183, 400, 102));
toolBar.setBounds(new Rectangle(0, 0, 400, 25));
jSlider1.setMajorTickSpacing(1);
jSlider1.setMaximum(5);
jSlider1.setPaintLabels(true);
jSlider1.setPaintTicks(true);
jSlider1.setBounds(new Rectangle(44, 30, 315, 45));
jSlider1.setSnapToTicks(true);
jSlider1.addMouseListener(new MidiFrame_jSlider1_mouseAdapter(this));
jSlider1.addInputMethodListener(new
        MidiFrame_jSlider1_inputMethodAdapter(this));
jSlider1.setEnabled(false);
jSlider1.setValue(0);
jPasswordField1.setToolTipText("");
jPasswordField1.setText("");
```

```
jPasswordField1.setBounds(new Rectangle(120, 125, 146, 20));
jSlider2.setBounds(new Rectangle(0, 0, 200, 24));
jSlider3.setMajorTickSpacing(1);
jSlider3.setMaximum(5);
jSlider3.setSnapToTicks(true);
jSlider3.setPaintLabels(true);
jSlider3.setPaintTicks(true);
jSlider3.setBounds(new Rectangle(42, 77, 315, 45));
jSlider3.setEnabled(false);
jSlider3.setValue(0);
jButton4.setBounds(new Rectangle(26, 154, 108, 22));
jButton4.setText("Get Message");
jButton4.addActionListener(new MidiFrame_jButton4_actionAdapter(this));
jButton5.setBounds(new Rectangle(155, 153, 102, 23));
jButton5.setText("Hide Message");
jButton5.addActionListener(new MidiFrame_jButton5_actionAdapter(this));
jButton6.setBounds(new Rectangle(292, 155, 83, 23));
jButton6.setText("Play Midi");
jButton6.addActionListener(new MidiFrame_jButton6_actionAdapter(this));
toolBar.add(jButton1);
toolBar.add(jButton2);
toolBar.add(jButton3);
menuFile.add(jMenuItem2);
menuFile.add(jMenuItem3);
menuFile.add(jMenuItem4);
menuFile.addSeparator();
menuFile.add(menuFileExit);
menuHelp.add(menuHelpAbout);
menuBar1.add(menuFile);
menuBar1.add(menuHelp);
this.setJMenuBar(menuBar1);
```

```
  jScrollPane1.getViewport().add(jTextArea1, null);
  contentPane.add(toolBar, null);
  contentPane.add(statusBar, null);
  contentPane.add(jScrollPane1, null);
  contentPane.add(jButton4);
  contentPane.add(jButton6);
  contentPane.add(jButton5);
  contentPane.add(jSlider3);
  contentPane.add(jSlider1);
  contentPane.add(jPasswordField1);
}

/**
 * Display the About box.
 */

    void helpAbout() {
  MidiFrame_AboutBox dlg = new MidiFrame_AboutBox(this);
  Dimension dlgSize = dlg.getPreferredSize();
  Dimension frmSize = getSize();
  Point loc = getLocation();
  dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x, (
    frmSize.height - dlgSize.height) / 2 + loc.y);
  dlg.setModal(true);
  dlg.show();
}

/**
 * Handle the File|Open menu or button, invoking okToAbandon and openFile
 * as needed.
 */
```

```
void fileOpen() {
  if (!okToAbandon()) {
    return;
  }
  // Use the OPEN version of the dialog, test return for Approve/Cancel
  if (JFileChooser.APPROVE_OPTION == jFileChooser1.showOpenDialog(this)) {
    // Call openFile to attempt to load the text from file into TextArea
    openFile(jFileChooser1.getSelectedFile().getPath());
    currFileName = jFileChooser1.getSelectedFile().getName();
    updateCaption();
  }
  this.repaint();
}

/**
 * Open named file; read text from file into jTextArea1; report to
 * statusBar.
 *
 * @param fileName String
 */
void openFile(String fileName)  {
  try {
    // Open a file of the given name.
    File file = new File(fileName);
    parse(file);
    jSlider1.setEnabled(true);
    jSlider3.setEnabled(true);
    jSlider3.setMaximum(score.getPart(jSlider1.getValue()).size());
    updateCaption();
  }
```

```
    catch (Exception e)
    {
      System.out.println(e);
    }
  }

  void parse (File file)
{
    updateCaption();
 score = Read.midiOrJmWithNoMessaging(file);
 jSlider1.setMinimum(0);
 jSlider1.setMaximum(score.size() - 1);
}

void saveMidi()
  {
    Score s = new Score();
    s.addPartList(score.getPartArray());
    Write.midi(s,currFileName);
    updateCaption();
  }

  /**
   * Save current file; handle not yet having a filename; report to
   * statusBar.
   *
   * @return boolean
   */
  boolean saveFile() {

    // Handle the case where we don't have a file name yet.
```

```
  if (currFileName == null) {
    return saveAsFile();
  }

   saveMidi();
  return false;
}

/**
 * Save current file, asking user for new destination name. Report to statuBar
 *
 * @return boolean
 */
boolean saveAsFile() {
  this.repaint();
  // Use the SAVE version of the dialog, test return for Approve/Cancel
  if (JFileChooser.APPROVE_OPTION == jFileChooser1.showSaveDialog(this)) {
    // Set the current file name to the user's selection,
    // then do a regular saveFile
    currFileName = jFileChooser1.getSelectedFile().getPath();

    //repaints menu after item is selected
    this.repaint();
    return saveFile();
  }
  else {
    this.repaint();
    return false;
  }
}
```

```
/**
 * Check if file is dirty. If so get user to make a "Save? yes/no/cancel"
 * decision.
 *
 * @return boolean
 */
boolean okToAbandon() {
  if (!dirty) {
    return true;
  }
  int value =  JOptionPane.showConfirmDialog(this, "Save changes?",
                    "Midi Edit", JOptionPane.YES_NO_CANCEL_OPTION) ;

  switch (value) {
     case JOptionPane.YES_OPTION:
       // yes, please save changes
       return saveFile();
     case JOptionPane.NO_OPTION:
       // no, abandon edits
       // i.e. return true without saving
       return true;
     case JOptionPane.CANCEL_OPTION:
     default:
       // cancel
       return false;
  }
}

/**
 * Update the caption of the application to show the filename
 *  and its dirty state.
```

```
 */

   void updateCaption() {
  String caption;

  if (currFileName == null) {
     // synthesize the "Untitled" name if no name yet.
     caption = "Untitled";
  }
  else {
    caption = currFileName;
  }

  // add a "*" in the caption if the file is dirty.
  if (dirty) {
    caption = "* " + caption;
  }
  caption = "The Sound of Music v. 1.0 - " + caption;
  this.setTitle(caption);
}

/**
 * File | Exit action performed
 *
 * @param e ActionEvent
 */
public void fileExit_actionPerformed(ActionEvent e) {
  if (okToAbandon()) {
    System.exit(0);
  }
}
```

```
/*Retrieves the text from a selected part and phrase of a MIDI file */
public void getText()
{
  //Prevents calling on a nonexistent part or phrase.
  if (score.getPart(jSlider1.getValue()).getPhrase(jSlider3.getValue()) != null)
  {

    //Creates a character array based on the length of the selected phrase
    char[] text = new char[score.getPart(
        jSlider1.getValue()).getPhrase(jSlider3.getValue()).size()];

    //Create String to hold the recovered text
    String str = "";
    String encStr = "";
    try {

      //Converts all notes pitches of selected phrase into characters and
      //stores them in the text character array.
      for (int i = 0;
          i < score.getPart(jSlider1.getValue()).getPhrase(
              jSlider3.getValue()).size(); i++) {
        text[i] = (char) score.getPart(
            jSlider1.getValue()).getPhrase(
                jSlider3.getValue()).getNote(i).getPitch();
      }

      //Constructs a String from the text character array.
      str = new String(text);

      //Decrypts the String using the password in the password box
      encStr = decrypt(jPasswordField1.getPassword(), str);
```

```
      //Sets the TextArea to the decrypted text
      jTextArea1.setText(encStr);
    }

    //Outputs any exceptions to the terminal screen.
    catch (Exception e) {
      System.out.println(e);
    }
  }
}

/*Hides whatever text is in the TextArea and put it into the midi file. */
public void hideText()
{
  try{
    //Get Text from TextArea
    String text = jTextArea1.getText();

    //Encrypt the text message using the encrypt method
    String encryptText = encrypt(jPasswordField1.getPassword(), text);

    //Get all parts from the score
    ArrayList partsList = new ArrayList();
    for (int i = 0; i < score.size(); i++) {
      partsList.add(score.getPart(i));
    }

    //Create a local phrase
    Phrase phrase = new Phrase();
```

```
//Create an ArrayList for all the phrases
ArrayList phrasesList = new ArrayList();

//Create an array for notes the same length as characters in the text
//message
Note[] newNotes = new Note[encryptText.length()];

//Turn each character of the text message into the pitch of a note.
for (int i = 0; i < encryptText.length(); i++) {
  newNotes[i] = new Note( (int) encryptText.charAt(i), 1);
}

//Create the phrase using the new notes.
phrase.addNoteList(newNotes);

//Mute the phrase
phrase.setMute(true);

//Set start time the same as the original selected phrase.
phrase.setStartTime(
    score.getPart(jSlider1.getValue()).getPhrase(
        jSlider3.getValue()).getStartTime());

//Get all phrases from the score
for (int i = 0; i < score.getPart(jSlider1.getValue()).length(); i++) {
  phrasesList.add(score.getPart(jSlider1.getValue()).getPhrase(i));
}

//Add the newly created phrase to the phrase ArrayList at the selected
//position.
phrasesList.add(jSlider3.getValue() + 1, phrase);
```

```
//Create a new part
Part part = new Part();

//Create a new Array to use in the addPhrasesArray method
Phrase[] phraseArray = new Phrase[phrasesList.size()];

//Get all phrases from the ArrayList and put them in an array
for (int i = 0; i < phrasesList.size(); i++) {
  phraseArray[i] = (Phrase) phrasesList.get(i);
}

//Add the newly created phrase Array to the newly constructed part
part.addPhraseList(phraseArray);

//Replace the current part with the newly constructed part
partsList.set(jSlider1.getValue(), part);

//Create a new part Array for use with the score constructor
Part[] partList = new Part[partsList.size()];

//Get all parts from the ArrayList and put them into an array
for (int i = 0; i < partsList.size(); i++) {
  partList[i] = (Part) partsList.get(i);
}

//Construct the new score
score = new Score(partList);

//Prompt to save the newly created score
saveAsFile();
```

```
  }

  //Catch any exceptions and print to terminal screen
  catch (Exception e)
  {
    System.out.println(e);
  }
  }



/**
 * Help | About action performed
 *
 * @param e ActionEvent
 */
public void helpAbout_actionPerformed(ActionEvent e) {
  helpAbout();
}

/**
 * Overridden so we can exit when window is closed
 *
 * @param e WindowEvent
 */
protected void processWindowEvent(WindowEvent e) {
  super.processWindowEvent(e);
  if (e.getID() == WindowEvent.WINDOW_CLOSING) {
    fileExit_actionPerformed(null);
  }
}
```

```
void jMenuItem1_actionPerformed(ActionEvent e) {
  // Handle the File|New menu item.
  if (okToAbandon()) {
    // clears the text of the TextArea
    jTextArea1.setText("");
    // clear the current filename and set the file as clean:
    currFileName = null;
    dirty = false;
    updateCaption();
  }
}

void jMenuItem2_actionPerformed(ActionEvent e) {
  //Handle the File|Open menu item.
  fileOpen();
}

void jMenuItem3_actionPerformed(ActionEvent e) {
  //Handle the File|Save menu item.
  saveFile();
}

void jMenuItem4_actionPerformed(ActionEvent e) {
  //Handle the File|Save As menu item.
  saveAsFile();
}

void jButton1_actionPerformed(ActionEvent e) {
  //Handle tool bar Open button
  fileOpen();
```

```
  }

  void jButton2_actionPerformed(ActionEvent e) {
    //Handle tool bar Save button
    saveFile();
  }

  void jButton3_actionPerformed(ActionEvent e) {
   //Handle tool bar About button
    helpAbout();
  }



  public void jSlider1_caretPositionChanged(InputMethodEvent event) {
    jSlider3.setMinimum(0);
    jSlider3.setMaximum(score.getPart(jSlider1.getValue()).size() - 1);
  }

public void jButton4_actionPerformed(ActionEvent e) {
  getText();
}


  public void jButton5_actionPerformed(ActionEvent e) {
    hideText();
  }

  public void jButton6_actionPerformed(ActionEvent e) {

    if (!playing)
```

```
    {
      play.midi(score);
      jButton6.setText("Stop Playing");
      playing = true;
      System.out.println("YEP, MADE IT THROGH THE IF.");
    }
    else
    {
      play.stopMidi();
      jButton6.setText("Play Midi");
    }
  }


  private static String encrypt(char[] password, String plaintext)
  throws Exception
  {
// Begin by creating a random salt of 64 bits (8 bytes)

  byte[] salt = new byte[8];
  Random random = new Random();
  random.nextBytes(salt);

// Create the PBEKeySpec with the given password

  PBEKeySpec keySpec = new PBEKeySpec(password);

// Get a SecretKeyFactory for PBEWithSHAAndTwofish

  SecretKeyFactory keyFactory =
  SecretKeyFactory.getInstance("PBEWithSHAAndTwofish-CBC");
```

```
// Create our key

  SecretKey key = keyFactory.generateSecret(keySpec);

// Now create a parameter spec for our salt and iterations

  PBEParameterSpec paramSpec = new PBEParameterSpec(salt, ITERATIONS);

// Create a cipher and initialize it for encrypting

  Cipher cipher = Cipher.getInstance("PBEWithSHAAndTwofish-CBC");
  cipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);

  byte[] ciphertext = cipher.doFinal(plaintext.getBytes());

  BASE64Encoder encoder = new BASE64Encoder();

  String saltString = encoder.encode(salt);
  String ciphertextString = encoder.encode(ciphertext);

  return saltString+ciphertextString;
  }
  private static String decrypt(char[] password, String text)
  throws Exception
  {
// Below we split the text into salt and text strings.

  String salt = text.substring(0,12);
  String ciphertext = text.substring(12,text.length());
```

```
// BASE64Decode the bytes for the salt and the ciphertext

  BASE64Decoder decoder = new BASE64Decoder();
  byte[] saltArray = decoder.decodeBuffer(salt);
  byte[] ciphertextArray = decoder.decodeBuffer(ciphertext);

// Create the PBEKeySpec with the given password

  PBEKeySpec keySpec = new PBEKeySpec(password);

// Get a SecretKeyFactory for PBEWithSHAAndTwofish

  SecretKeyFactory keyFactory =
  SecretKeyFactory.getInstance("PBEWithSHAAndTwofish-CBC");

// Create our key

  SecretKey key = keyFactory.generateSecret(keySpec);

// Now create a parameter spec for our salt and iterations
  PBEParameterSpec paramSpec =
  new PBEParameterSpec(saltArray, ITERATIONS);

// Create a cipher and initialize it for encrypting

  Cipher cipher = Cipher.getInstance("PBEWithSHAAndTwofish-CBC");
  cipher.init(Cipher.DECRYPT_MODE, key, paramSpec);

// Perform the actual decryption

  byte[] plaintextArray = cipher.doFinal(ciphertextArray);
```

```
  return new String(plaintextArray);
}

  public void jSlider1_mouseClicked(MouseEvent e) {
    jSlider3.setMinimum(0);
    System.out.println("That number is: " + score.getPart(
      jSlider1.getValue()).size());
    jSlider3.setMaximum(score.getPart(jSlider1.getValue()).size() - 1);

  }

}

class MidiFrame_jButton4_actionAdapter
    implements ActionListener {
  private MidiFrame adaptee;
  MidiFrame_jButton4_actionAdapter(MidiFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButton4_actionPerformed(e);
  }
}

class MidiFrame_jButton6_actionAdapter
    implements ActionListener {
  private MidiFrame adaptee;
  MidiFrame_jButton6_actionAdapter(MidiFrame adaptee) {
    this.adaptee = adaptee;
```

```
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButton6_actionPerformed(e);
  }
}

class MidiFrame_jButton5_actionAdapter
    implements ActionListener {
  private MidiFrame adaptee;
  MidiFrame_jButton5_actionAdapter(MidiFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButton5_actionPerformed(e);
  }
}

class MidiFrame_jSlider1_inputMethodAdapter
    implements InputMethodListener {
  private MidiFrame adaptee;
  MidiFrame_jSlider1_inputMethodAdapter(MidiFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void inputMethodTextChanged(InputMethodEvent event) {
  }

  public void caretPositionChanged(InputMethodEvent event) {
    adaptee.jSlider1_caretPositionChanged(event);
```

```
  }
}

class MidiFrame_jSlider1_mouseAdapter
    extends MouseAdapter {
  private MidiFrame adaptee;
  MidiFrame_jSlider1_mouseAdapter(MidiFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void mouseClicked(MouseEvent e) {
    adaptee.jSlider1_mouseClicked(e);
  }
}

class MidiFrame_menuFileExit_ActionAdapter implements ActionListener {
  MidiFrame adaptee;

  MidiFrame_menuFileExit_ActionAdapter(MidiFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.fileExit_actionPerformed(e);
  }
}

class MidiFrame_menuHelpAbout_ActionAdapter implements ActionListener {
  MidiFrame adaptee;

  MidiFrame_menuHelpAbout_ActionAdapter(MidiFrame adaptee) {
```

```
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.helpAbout_actionPerformed(e);
  }
}


class MidiFrame_jMenuItem1_actionAdapter implements
    java.awt.event.ActionListener {
  MidiFrame adaptee;

  MidiFrame_jMenuItem1_actionAdapter(MidiFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItem1_actionPerformed(e);
  }
}

class MidiFrame_jMenuItem2_actionAdapter implements
    java.awt.event.ActionListener {
  MidiFrame adaptee;

  MidiFrame_jMenuItem2_actionAdapter(MidiFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
```

```
    adaptee.jMenuItem2_actionPerformed(e);
  }
}

class MidiFrame_jMenuItem3_actionAdapter implements
    java.awt.event.ActionListener {
  MidiFrame adaptee;

  MidiFrame_jMenuItem3_actionAdapter(MidiFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItem3_actionPerformed(e);
  }
}

class MidiFrame_jMenuItem4_actionAdapter implements
    java.awt.event.ActionListener {
  MidiFrame adaptee;

  MidiFrame_jMenuItem4_actionAdapter(MidiFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItem4_actionPerformed(e);
  }
}

class MidiFrame_jButton1_actionAdapter implements
```

```
    java.awt.event.ActionListener {
  MidiFrame adaptee;

  MidiFrame_jButton1_actionAdapter(MidiFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButton1_actionPerformed(e);
  }
}

class MidiFrame_jButton2_actionAdapter implements
    java.awt.event.ActionListener {
  MidiFrame adaptee;

  MidiFrame_jButton2_actionAdapter(MidiFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButton2_actionPerformed(e);
  }
}

class MidiFrame_jButton3_actionAdapter implements
    java.awt.event.ActionListener {
  MidiFrame adaptee;

  MidiFrame_jButton3_actionAdapter(MidiFrame adaptee) {
    this.adaptee = adaptee;
```

```
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButton3_actionPerformed(e);
  }
}
```

Code - MidiClass.java

```
import javax.swing.UIManager;
import java.awt.*;

public class MidiClass {
  boolean packFrame = false;

  /**
   * Construct the application
   */

     public MidiClass() {
   MidiFrame frame = new MidiFrame();
   //Validate frames that have preset sizes
   //Pack frames that have useful preferred size info, e.g. from their layout
   if (packFrame) {
     frame.pack();
   }
   else {
     frame.validate();
   }
   //Center the window
   Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
   Dimension frameSize = frame.getSize();
   if (frameSize.height > screenSize.height) {
     frameSize.height = screenSize.height;
   }
   if (frameSize.width > screenSize.width) {
     frameSize.width = screenSize.width;
   }
   frame.setLocation((screenSize.width - frameSize.width) / 2, (
```

```
    screenSize.height - frameSize.height) / 2);
  frame.setVisible(true);
 }

 /**
  * Main method
  *
  * @param args String[]
  */
 public static void main(String[] args) {
  try {
   UIManager.setLookAndFeel(
       UIManager.getCrossPlatformLookAndFeelClassName());
  }
  catch(Exception e) {
   e.printStackTrace();
  }
  new MidiClass();
 }
}
```

Code – MidiFrame_AboutBox.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class MidiFrame_AboutBox extends JDialog implements ActionListener {

  JPanel panel1 = new JPanel();
  JPanel panel2 = new JPanel();
  JPanel insetsPanel1 = new JPanel();
  JPanel insetsPanel2 = new JPanel();
  JPanel insetsPanel3 = new JPanel();
  JButton button1 = new JButton();
  JLabel imageControl1 = new JLabel();
  ImageIcon imageIcon;
  JLabel label1 = new JLabel();
  JLabel label2 = new JLabel();
  JLabel label3 = new JLabel();
  JLabel label4 = new JLabel();
  BorderLayout borderLayout1 = new BorderLayout();
  BorderLayout borderLayout2 = new BorderLayout();
  FlowLayout flowLayout1 = new FlowLayout();
  FlowLayout flowLayout2 = new FlowLayout();
  GridLayout gridLayout1 = new GridLayout();
  String product = "The Sound of Encryption";
  String version = "Version 1.0";
  String copyright = "Copyright (c) 2005";
  String comments = "Hides a message within a MIDI file.";
  public MidiFrame_AboutBox(Frame parent) {
    super(parent);
```

```
  enableEvents(AWTEvent.WINDOW_EVENT_MASK);
  try {
    jbInit();
  }
  catch(Exception e) {
    e.printStackTrace();
  }
  pack();
}

private void jbInit() throws Exception  {
  //imageIcon = new ImageIcon(getClass().getResource("[Your Image]"));
  this.setTitle("About");
  setResizable(false);
  panel1.setLayout(borderLayout1);
  panel2.setLayout(borderLayout2);
  insetsPanel1.setLayout(flowLayout1);
  insetsPanel2.setLayout(flowLayout1);
  insetsPanel2.setBorder(new EmptyBorder(10, 10, 10, 10));
  gridLayout1.setRows(4);
  gridLayout1.setColumns(1);
  label1.setText(product);
  label2.setText(version);
  label3.setText(copyright);
  label4.setText(comments);
  insetsPanel3.setLayout(gridLayout1);
  insetsPanel3.setBorder(new EmptyBorder(10, 60, 10, 10));
  button1.setText("Ok");
  button1.addActionListener(this);
  insetsPanel2.add(imageControl1, null);
  panel2.add(insetsPanel2, BorderLayout.WEST);
```

```
    this.getContentPane().add(panel1, null);
    insetsPanel3.add(label1, null);
    insetsPanel3.add(label2, null);
    insetsPanel3.add(label3, null);
    insetsPanel3.add(label4, null);
    panel2.add(insetsPanel3, BorderLayout.CENTER);
    insetsPanel1.add(button1, null);
    panel1.add(insetsPanel1, BorderLayout.SOUTH);
    panel1.add(panel2, BorderLayout.NORTH);
  }

  protected void processWindowEvent(WindowEvent e) {
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
      cancel();
    }
    super.processWindowEvent(e);
  }

  void cancel() {
    dispose();
  }

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == button1) {
      cancel();
    }
  }
}
```