A New Tool with Piracy Protection for Hiding Valuable Data in Digital Media

S.Bala Sudha [#1], K. V. N Rajesh [*2], G. Jyothi [*3]

[#1]M.TECH, Department of information technology, vignan's institute of information technology, JNTU-KAKINADA, Andhra Pradesh, India
[#1] Sudha.vitts@gmail.com

[*2]Assistant professor, Department of information technology, vignan's institute of information technology, Visakhapatnam, Andhra Pradesh, India
[*2] kvnrajesh@vignanvizag.com

[*3]Assistant professor, Department of information technology, vignan's institute of information technology, Visakhapatnam, Andhra Pradesh, India
[*3] jyothi506@gmail.com

Abstract

Stegnography is a new branch of security through which one form of data can be hidden in another form of data of either same type or of different form type. This new mechanism is mainly implemented in order to provide much more security for data which is transferring through the network. As the user regularly transfer a lot of files from one system to other system either within the range or far range by suing internet or intranet, he eventually looks for more security .As we know that ordinary file encryption and decryption concepts, which are readily available in java examples are easily captured by middle way (I.e. During transmission) itself. So we need more security combination for sending the digital form of data. This paper helps to analyze how to send a file from one place to another in a secured manner. Firstly the target file is encrypted using our new algorithm called as DES Bit Shifting and it is embedded into an audio or video or any media file. The resultant file will be protected by a password. This resultant media file is no change in its original format and it can be run in the player, we can't find any encrypted data inside it. This format will be sent through Internet or through any form of wired communication networks. In the destination point it will be retrieved only by the same Stegnography protection software and giving the relevant password. So it is highly secured.

Keywords

Embedding, De-Embedding, Hiding, Information Hiding, Stegnography, Watermarking,

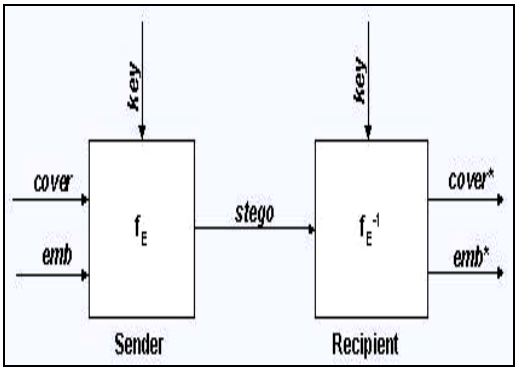
1. Introduction

Today's communication of valuable digital data (I.e. Image, Video, and Audio) through public or un-Secured channels have become a most critical problem in the society. This major problem is solved by using the new concept called steganography, which is the art and science of hiding valuable information into Master channels so as to conceal the information and prevent the detection of the hidden message. Steganography is also defined as hiding information within a noise; a way to supplement (not replace) encryption, to prevent the existence of encrypted data from being detected [1] by the un-authorized users.

According to greek literature ,steganography is also known as "covered writing method", a branch which deals mainly with only two methods like Embedding and De-Embedding of original valuable content within a Cover file like image, video, or audio. This new technique is mainly used by the Indian Government in the Military to establish relationship between more than two military commanders in a much secured manner without releasing or misusing any small part of embedded data [2], [3], [4], [5], [6].

1.1 Stegnographic System

We can clearly find the advantages of stegnography mechanism from the below figure .1, which clearly states the explanation of stegnography method used for embedding an image within an image, the same principle is used for embedding audio and video also within same type of formats or different type of formats like audio in image, image in audio, video in image, image in video, image in image, video in video, audio in audio and so on. So in this paper we clearly explain the advantage of mixed stegnography of how one type of digital media data is embedded with other type of digital data by giving password for the embedded data.



Where $\mathbf{f_E}$ **:** Which clearly denotes Stegnographic Function for embedding.

$\mathbf{f_E^{-1}}$: Which clearly denotes stegnographic function for extracting.

Cover: This is the main source in which the data will be hidden.

Emb: This is the function which indicates message to be hidden.

Key: Parameter of f_E

Stego: This is a function which denotes cover data with hidden data.

In this paper, we are not following the regular Cryptographic encryption and regular cryptographic decryption techniques. We are introducing a new algorithm called Bit Shift encryption in Random Cycle Order. I.e. totally 4 different types of Bit Shift algorithms are used randomly to encrypt the data like 4-Bit,6-Bit,12 Bit ,16 Bit Shift Encryption Algorithms. This Encryption is embedded into an Audio or Video File. Again it will be embedded into a media data. This double embedding is increase the level of security. Password Protection of this entire works gives an additional security for this total application, if there was no password facility the user may lost the valuable data in the terms of intruders between data transmission.

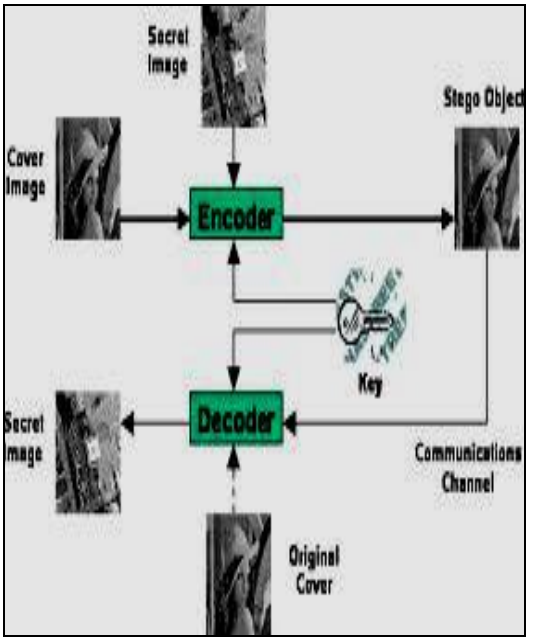


Figure. 1. Block diagram of Stegnography Mechanism

2. Literature Survey

In this section we will describe the assumptions that are used in the proposed paper. This section mainly surveys on the literature of our proposed method, we will try to find some of the related data regarding stegnography which was practically used in the real time applications.

2.1 Graphical Version of the Stegnographic System

The graphical version of the stegnography system is clearly represented in figure 2, where the stegnographic messages must be first encrypted and then it is hidden inside a master file, which results in forming the stegno text. Only those who know the exact technique which was used by the sender can recover the message and, if required, decrypt it.

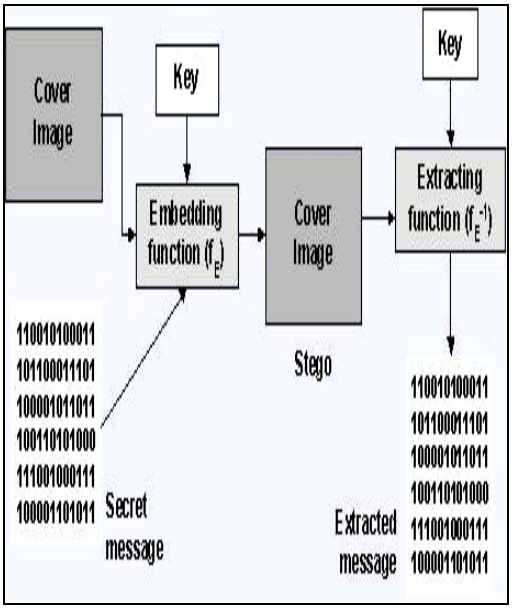


Figure. 2. Represents Graphical Version of the Stegnographic System

Digital images are commonly stored in either 24-bit or 8-bit files. For Example if an 8-bit image is viewed as a grid (I.e. Grid is nothing but representing of data in rows and columns as a tabular matrix), these cells are called as pixels. Each pixel consists of an 8-bit binary number (or a single byte), and each 8-bit binary number refers to the color palette (a set of colors defined within the image). All color variations for the pixels are derived from three primary colors: red, green, and blue. Each primary color is represented by 1 byte (= 8 bits).

3. Our Proposed Stegnography Model and its Methodology

The following are the proposed Stegnography model and some of the bit shift operations are explained in detail in this section. The methodologies that are used is explained in detail in this section.

3.1 Stegnography Techniques

Now a day's the data hiding techniques are receiving more and more attention. The main motivation for this new technique is largely due to fear of encryption services getting outlawed. There are several ways to hide information in digital images. We look at the following 3 important approaches:

A. Least Significant Bit Insertion
B. Masking And Filtering
C. Algorithms and Transformations

Each of these Techniques has Varying Degrees of Success in comparison.

Figure 3. Stegno Image for Data Hiding

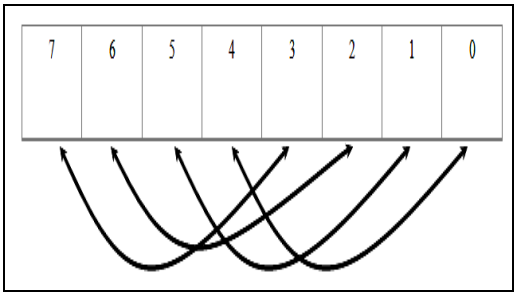
For Example

The above represented figure is a Embedded Image where we embedded show how the image is taken into matrix form with all Zero's and One's.

3.2 Shift Transformations Operations

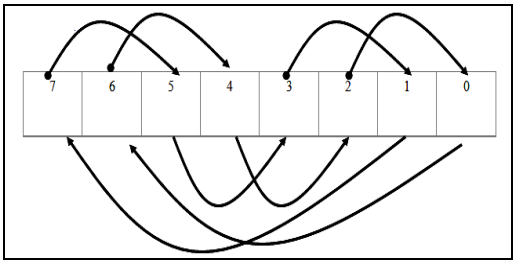
The following are the bit shift operations that are used in our present paper. They are as follows

Bit Position



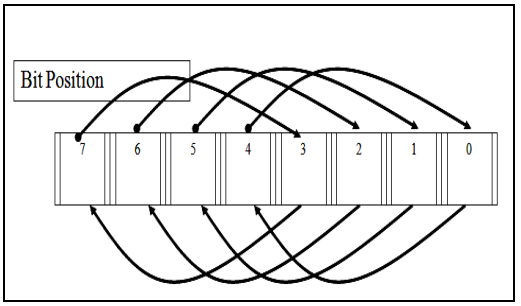
Shift Algorithm – 4 Shift

The above is the 4 Bit position which is used for shifting 4 bit positions from left to right, starts from the middle bit position.

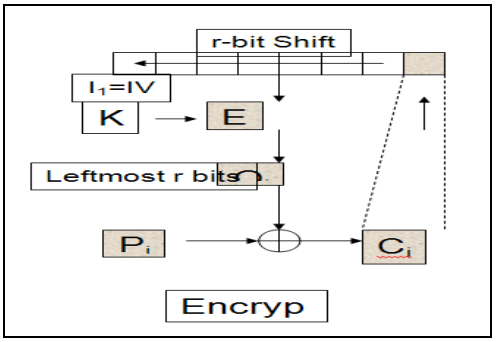


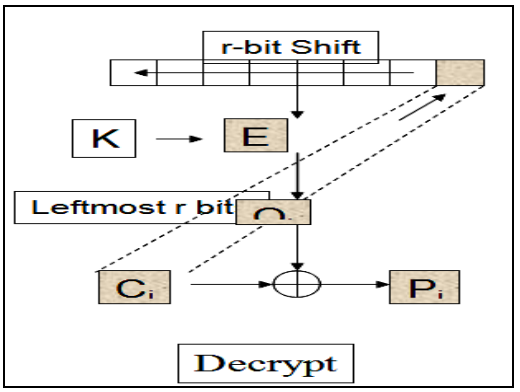
Shift Algorithm – 6 Shift

The above Bit Position clearly indicates it is a 8 bit string with change of 6 bit positions from left to right side. In the same way we can do continue with remaining other bit positions like 12 and 16 bit positions.



Bit Shifting – Encrypt & Decrypt





4. Methodology of Proposed DES Algorithm

In this paper we have implemented DES 64 bit key algorithm for encrypting the hidden data with a password. The password should be minimum 8 characters as the DES algorithm uses 64-bit key.

Data Encryption Algorithm

The Data Encryption Algorithm works as follows, which is clearly shown in figure 4.

DES Flow Chart

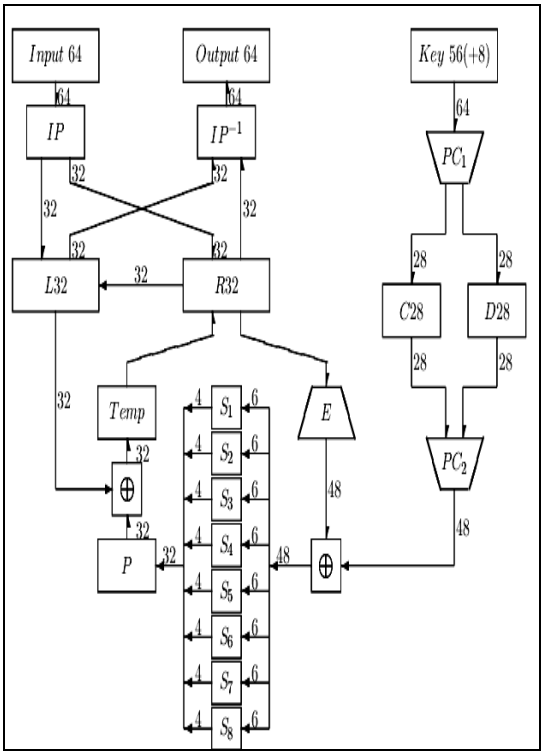


Figure 4. Flow chart representation of DES Encryption / Decryption

Substitution-Permutation Algorithm:

64-bit input and output blocks.56-bit key (with an additional 8 parity bits).Information data is cycled 16 times through a set of substitution and permutation transformations: highly non-linear input-output relationship. We can find the permutations of DES clearly in the Table 1.

1. Very high throughput rates achievable (up to 100 Mbits / s)
2. Availability of economical hardware to implement DES

3. Low to medium security applications (e.g. secure speech communications)

Box	Row	Column 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Table 1: S-boxes (substitution boxes) tables

5. Project Implementation Modules

As our application is implemented in Java as the chosen technology with Java Swings as front end user interfaces. In this application we are not using any data base as back end for storing the output file or any form of hidden data.

5.1 Main Modules

The following paper is divided into following four modules. With the help of these four modules we are able to provide security for the hidden data over transmission channel. They are as follows:

1. Embedding a Message in any Form of Digital Data:

Here in this module, we will embed a plain text message inside a .wmv format video file or any form of digital data file in order to hide valuable message like audio/image/video. We also give security for that hidden master file containing message with a password by encrypting the message with a key.

2. Embedding a data file within any form of digital data:

Here in this module, we will embed a data file containing valuable data inside a .wmv format video file or any form of digital data file in order to hide that sensible data like image/audio/video .We also give security for that hidden master file containing sensible with a password by encrypting that data file with a key.

3. Retrieving Message from hidden Master File:

Here in this module, retrieving message from a Master file: check whether compression and encryption have been used and the compression ratio if compression has been used. It also shows you the request you have made. If the message is encrypted and decrypted by the same password only.

4. Retrieving Data File from Hidden Master File.

Retrieving file from a Master file: check whether compression and encryption have been used and the compression ratio if compression has been used. It also shows you the request you have made. If the message is encrypted and decrypted by the same password only.

5.2 Main Source Code for the Proposed Mixed Stegnography Application

Here we will discuss the main source code for the proposed mixed stegnography application implementation. Here I will show the main logic part through which the current application is connecting its frond end user interface with the storage data. For this application we are using Java Swings as front end and there was no back end data base for the current application as the input for this application can be browse or chosen from entire system. The input may be of digital data comprising Audio / Video / Images.

Sample Code for MainFrame. Java

```
import javax.swing.UIManager.*;
public class MainFrame extends javax.swing.JFrame
 {
   public MainFrame()
   {
            super();
      initComponents();
   }   /**
    * This method is called from within the constructor to initialize the form.
    * WARNING: Do NOT modify this code. The content of this method is always    * regenerated by the Form Editor.    */
   @SuppressWarnings("unchecked")

   // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
   private void initComponents()
{

      jPanel1 = new javax.swing.JPanel();
      jLabel1 = new javax.swing.JLabel();
      jLabel2 = new javax.swing.JLabel();
      jScrollPane1 = new javax.swing.JScrollPane();
      jTextArea1 = new javax.swing.JTextArea();
      jButton1 = new javax.swing.JButton();

      setTitle("Copyright @ Sudha M.TECH 2014 Batch..VIGNAN COLLEGE");

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
      jPanel1.setBackground(new java.awt.Color(255, 255, 255));
      jLabel1.setFont(new java.awt.Font("Times New Roman", 3, 24)); // NOI18N
      jLabel1.setForeground(new java.awt.Color(204, 0, 204));
      jLabel1.setText("A New Tool with Piracy Protection for Hiding Valuable Data in Digital Media");
      jLabel2.setFont(new java.awt.Font("Times New Roman", 2, 18)); // NOI18N
      jLabel2.setForeground(new java.awt.Color(102, 102, 255));
      jLabel2.setText("About");

      jTextArea1.setColumns(20);

      jTextArea1.setRows(5);
            jTextArea1.setAlignmentX(5.0F);
      jTextArea1.setAlignmentY(5.0F);
      jScrollPane1.setViewportView(jTextArea1);
      jButton1.setFont(new java.awt.Font("Times New Roman", 3, 14)); // NOI18N
      jButton1.setForeground(new java.awt.Color(102, 102, 255));
      jButton1.setText("LOGIN TO ENTER INTO PIRACY PROTECTION SOFTWARE");
      jButton1.addActionListener(new java.awt.event.ActionListener() {
          public void actionPerformed(java.awt.event.ActionEvent evt) {
              jButton1ActionPerformed(evt);
          }
      });
```

Sample code for BackendHandler.java

```
import javax.swing.*;
import java.awt.*;
import java.awt. Event.*;
import java.io.*;
import javax.crypto.*;
import javax.crypto.spec.*;
public class BackEndHandler extends Thread
{
```



```
public static final short EMBED_MESSAGE= 0;
public static final short EMBED_FILE= 1;
public static final short RETRIEVE_MESSAGE= 2;
public static final short RETRIEVE_FILE= 3;
public static final short EDIT_MASTER= 4;
private short operation;
private WindowAdapter client;
private JFileChooser fileChooser;
private MyFileView fileView;
private File masterFile, dataFile, outputFile;
private int result, result2;
public BackEndHandler(WindowAdapter client, short operation)
{
this.client= client;
this.operation= operation;
fileChooser= new JFileChooser("./");
fileChooser.setFileSelectionMode(fileChooser.FILES_ONLY);
fileChooser.setDialogType(fileChooser.CUSTOM_DIALOG);
fileChooser.setAccessory(new FilePreviewer(fileChooser));
MyFileFilter filter3= new MyFileFilter(new String[]{"mpg","wmv","avi"}, "Video files");
fileChooser.addChoosableFileFilter(filter3);
        }
        public void run()
        {
if(!chooseMasterFile()) return;
        if(operation== EMBED_MESSAGE || operation== EMBED_FILE)
        if(!chooseOutputFile()) return;
if(operation== EMBED_FILE)
if(!chooseDataFile()) return;
SteganoInformation steg;
switch(operation){
case EMBED_MESSAGE: new EmbedMessageGUI(this); break;
case EMBED_FILE: new EmbedFileGUI(this); break;
case RETRIEVE_MESSAGE:
steg= new SteganoInformation (masterFile);
if(steg.isEster())
showEster(steg);
else
if(!steg.isValid())
JOptionPane.showMessageDialog(null, "File '"+ masterFile.getName()+ "' does not contain any message or file\nembedded using watermarking !", "Invalid watermarking file!", JOptionPane.WARNING_MESSAGE);
else new PreRetrieveGUI(steg, PreRetrieveGUI.RETRIEVE_MESSAGE);
break;
case RETRIEVE_FILE:
steg= new SteganoInformation (masterFile);
if(steg.isEster())
showEster(steg);
else
if(!steg.isValid())
JOptionPane.showMessageDialog(null, "File '"+ masterFile.getName()+ "' does not contain any message or file\nembedded using watermarking!", "Invalid watermarking file!", JOptionPane.WARNING_MESSAGE);
else
newPreRetrieveGUI(steg, PreRetrieveGUI.RETRIEVE_FILE);
        }
        }
        // Method for choosing input file
        public boolean chooseMasterFile()
        {
                int result;
                do
                {
result= fileChooser.showDialog(null, "Select Master file");
if(result== fileChooser.APPROVE_OPTION)
                        {
masterFile= fileChooser.getSelectedFile();
if(!checkFileExistency(masterFile))
continue;
else
break;
}} while(result!= fileChooser.CANCEL_OPTION);

if(result== fileChooser.CANCEL_OPTION)
return false;
else return true;
        }

        // Method for choosing output file
public boolean chooseOutputFile()
{
int result;
do{
File previousFile= fileChooser.getSelectedFile();
```



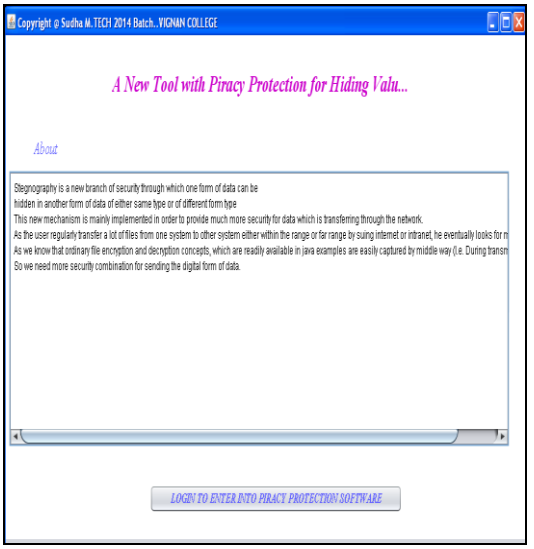
```
result= fileChooser.showDialog(null, "Select Output file");

if(result== fileChooser.APPROVE_OPTION)
{
outputFile= fileChooser.getSelectedFile();

if(outputFile.exists())
{
result2= JOptionPane.showConfirmDialog(null, "File "+ outputFile.getName()+ " already exists!\nWould you like to OVERWRITE it?", "File already exists!", JOptionPane.YES_NO_OPTION);

                                        if(result2== JOptionPane.NO_OPTION)

if(previousFile!= null)
fileChooser.setSelectedFile(previousFile);
continue;
}
}
break;
}
}
while(result!= fileChooser.CANCEL_OPTION);
if(result== fileChooser.CANCEL_OPTION)
        return false;
                else return true;
        }
// Method for choosing data file
        public boolean chooseDataFile()
        {
                do
{
result= fileChooser.showDialog(null, "Select Data file");
if(result== fileChooser.APPROVE_OPTION)

{dataFile= fileChooser.getSelectedFile();

if(!checkFileExistency(dataFile))

continue;else

break;
}
}
while(result!= fileChooser.CANCEL_OPTION);
if(result== fileChooser.CANCEL_OPTION)
        return false;else
        return true; }
```

6. Experimental Results

In this paper, we have mainly sender who want to embed the valuable hidden data onto a cover page and wants to transmit over communication channel, for this the window we created in java looks like below

The below window is the starting window or home window for our proposed project. In this window we will tell the project abstract in detail in the text area that is present in that main window. If the user who wishes to participate in stegnography process, he should click on submit button so that he can be enter into the home page, if not he will not be directed to stegnography page.

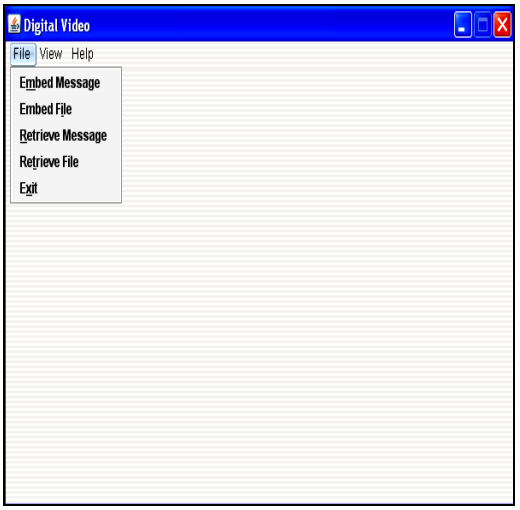
Main Frame



Once after we click on login button the following Stegnography window will displays in which we have facility of embedding message, embedding a file.

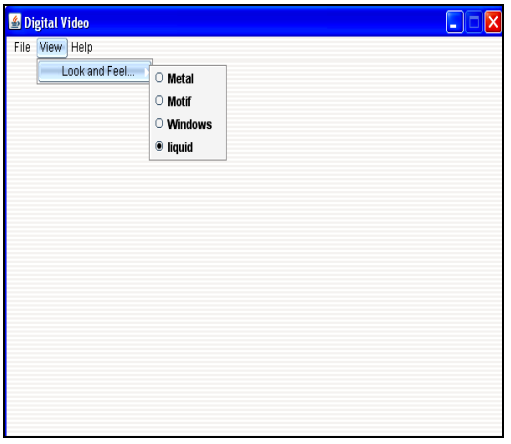
Stegnography Main Window

In this window we have a facility of embedding a message as well as embedding a data file for giving security as well as retrieving a Message/data file for getting the hidden data.



Look and Feel Design Window

In this application we have Look and Feel options for changing the default window appearance as the user wish .This is available in the view menu with various look and Feel options.



Exit Window

This window is mainly designed in order to ask confirmation whenever any user who wish to close the current process. If the user clicks on yes option then window gets closed otherwise it will be in same stegnography window.

7. Conclusion

In this paper, we mainly targeted on the problem of hiding sensitive valuable information into any digital form of data like audio, video, image. If one were able to hide the message in the video file in such a way, that there would be no perceivable changes in the audio file after the message insertion. At the same time, if the message that is to be hidden were encrypted, the level of security would be raised to quite a satisfactory level. Now that this ancient art and science has been applied to modern communications systems, it has become a very effective form of sending imperceptible messages.

8. Future Enhancement

In future we can extend the research of giving security for the multimedia digital data on OSN networks as the OSN users are communicating mostly with digital data and if they want to share the private digital data from one another, they can use the same stegnography concept for implementing security during data transmission.

9. References

[1] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding: A survey," *Proc. IEEE (Special Issue on Identification and Protection of Multimedia Information)*, vol. 87, pp. 1062-1078, July 1999.

[2] N. F. Johnson and S. Katzenbeisser, "A survey of steganographic techniques," in *Information Hiding*, S. Katzenbeisser and F. Petitcolas Eds. Norwood, MA: Artech House, 2000, pp. 43-78.

[3] S. Wang and H. Wang, "Cyber warfare: Steganography vs. steganalysis," *Communications of the ACM*, vol. 47, pp. 76-82, Oct. 2004.

[4] C. Cachin, "An information-theoretic model for steganography," in *Proc. 2nd Intern. Workshop on Information Hiding*, Portland, OR, Apr. 1998, pp. 306-318.

[5] G. J. Simmons, "The prisoner's problem and the subliminal channel," in *Advances in Cryptology: Proc. CRYPTO'83*. New York, NY: Plenum, 1984, pp. 51-67.

[6] J. Fridrich, *Steganography in Digital Media, Principles, Algorithms, and Applications*. Combridge, UK: Combridge Univeristy Press, 2010.

[7] Y. Wang and P. Moulin, "Perfectly secure steganography: Capacity, error exponents, and code constructions," *IEEE Trans. Inform. Theory*, vol. 54, pp. 2706-2722, June 2008.

