A PROJECT REPORT

ON

"DATA SECURITY FOR FILE TRANSMISSION"

SUBMITTED FOR PARTIAL FULFILMENT OF AWARD OF

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

BY

SHYAM BABU JAYSWAL (1902300100094)
MADHAV KAUSHIK (1902300100051)
SHIVAM KUMAR RAI (1902300100092)
UMA SHANKAR YADAV (1902300100106)

UNDER THE SUPERVISION OF -

Prof. Bhanu Bharadwaj

Assistant Professor
(Department of CSE)

DRONACHARYA

DRONACHARYA GROUP OF INSTITUTIONS, GREATER NOIDA (Affiliated to AKTU - Lucknow, Uttar Pradesh)

MAY , 2023

CERTIFICATE

This is to certify that the project entitled " **Data Security for File Transmission** " is being submitted by **Shyam Babu Jayswal (1902300100094**), **Madhav Kaushik(1902300100051), Shivam Kumar Rai(1902300100092), Uma Shankar Yadav(1902300100106)**in partial fulfilment for the degree of **Bachelor of Technology** in Computer science and Engineering of **Dronacharya Group of institutions**, Greater Noida (Affiliated to AKTU-Lucknow) is a record of their own work, carried out under my supervision.

Prof. Bhanu Bharadwaj
(Assistant Professor)
Department of CSE

Prof . Bipin Pandey
(Head of Department)
Department of CSE

ABSTRACT

The “Data Security for File Transmission” is a web application which deals with security during transmission of data across the network. Security for the data is required, as there is always a possibility for someone to read that secret data. The system deals with implementing security using steganography. Steganography is the art of hiding information in ways to prevent detection of hidden messages. Secure data transmission through network is software, which tries to alter the originality of the data files into some encrypted form by using “Tiny Encryption Algorithm”. Encryption of data plays a vital role in the real time environment to keep the data out of the reach of unauthorized people, such that it is not altered and tampered. After encryption, the files can be transferred securely by using steganography. The application should have a reversal process as of which should be able to decrypt the data to its original format upon the proper request by the user.

The goal of our project is to design a tool for providing security to the system during transmission of data through the network. The project is developed using graphics in java swings.

ACKNOWLEDGEMENT

It is always a difficult task to acknowledge all those who have been of tremendous help in an academic project of this nature and magnitude nevertheless, we have made a sincere attempt to express our gratitude to all those who have contributed to the successful completion of this project through this project report.

As we represent this report on " Data Security for File Transmission " we are aware of humanity and grateful to all the individuals who have so kindly offered us their time, skill, knowledge, advice, and facilities or guidance.

We are extremely grateful to Prof. Bhanu Bharadwaj, Assistant Professor DGI-GN for giving his valuable input and opportunity to develop this project and for making all the resources available to us to succeed this project.

We would also like to thank Prof. Bipin Pandey, H.O.D. CSE DGI-GN for his idea, advice, knowledge, and all his support throughout this project.
We also take this opportunity to express my deepest gratitude to our Project Guide and our faculty, Prof. Bhanu Bharadwaj, for his constant support, guidance, and encouragement and thus being a constant source of inspiration for us.

Finally, we would like to thank my parents, all the employees and all my friends who have always been a strong support during the entire course of my project and without their co-operation the completion of this project would not have been possible.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

The project titled “Data Security” is mainly designed for providing security during transmission of data across the network. In this the sender encrypts the data in to some form by using “Tiny Encryption Algorithm”. This algorithm has been used because it requires less memory. It uses only simple operations; therefore, it is easy to implement.

While encrypting the data into some form, the key file is entered by the sender. The purpose of the key file is to provide security to the system as it is known only to the sender and the receiver. The encrypted data will be embedded with a video file by using the concept of steganography. By using the Input/Output packages the steganography will read the video file and encrypted data and takes whole it as a video file. So, whenever the hacker tries to open the file, only the video file is visible to them. Then this video file is sent to the network.

The receiver will receive the video file from the network. Then the receiver will de-embed the encrypted data from the video file. The application of decryption is only done when the receiver of the data enters the proper key. Thus, the data is transferred from sender to receiver in a secure manner.

2. PROBLEM SPECIFICATION

2.1 Problem Definition

Present day transactions are "un-trusted" in terms of security, i.e., they are relatively easy to hack. And, we have to consider the large amount of data through the network will give errors while transferring. Nevertheless, sensitive data transfer is to be carried out even if there is lack of an alternative. Network security in the existing system is the motivation factor for a new system with higher-level security standards for the information exchange.

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 Introduction

3.1.1 Purpose

- Implementation of Tiny Encryption Algorithm with input plain text and key length of 128-bit.
- Implementation of steganography using Input/Output packages to provide security.
- Achieving SECRECY.

3.1.2 Scope

Security is a major aspect in present day computer networks. In this application, we are making it secure by implementing Tiny Encryption Algorithm which makes unauthorized decryption to very complex and nearly impossible. Moreover, the key also needed to be decrypted thus it makes it more robust to break the key which is large.

3.1.3 Objective

The main objective of this project is to develop a software solution to provide security under an Operating System like Windows which has very low level of security. Security is concerned with which makes sure that people cannot read or worse yet, secretly modify messages intended for other recipients. It is also concerned with people trying to access remote services that they are not authorized to use.

3.2 Requirements

3.2.1 User Requirements

To use the project, the user must be acknowledged with minimum requirements such as extension file to be used, specifying proper key and contain required inputs**.**

3.2.2 Software Requirements

Operating system	:	Linux 4.2.1.6/windows XP SP2
Platform	:	Java J2SDK 1.5.

3.2.3 Hardware Requirements

RAM	:	512MB
Hard Disk	:	40GB
Processor	:	Pentium D CPU 3.00GHZ
Monitor	:	LCD color Monitor 17

3.2.4 Functional Requirements

Functional Requirements will describe the system's services in detail. Functional Requirements of the project are:

- Encryption
- Embed
- Send File to network.
- De embed.
- Decryption

3.2.5 Performance Requirements

- The video file must look like the original, even after the process is completed.
- The video should not reveal any clues after embedding, with respect to un-authorized detection, unable to detect or unsuspicious.

4. ANALYSIS

4.1 Existing System

In the traditional architecture there existed only the server and the client. In most cases the server was only a database server that can only offer data. Therefore, most of the business logic had to be placed on the client's system. This makes maintenance expensive. This also means that every client must be trained as to how to use the application and even security in communication is also the factor to be considered.

Since the actual processing of the data takes place on the remote client the data must be transported over the network, which requires a secured format of the transfer method. Present day transactions are "un-trusted" in terms of security, i.e., they are relatively easy to hack. And also, we have to consider the transfer of the large amount of data through the network will give errors while transferring. Nevertheless, sensitive data transfer is to be carried out even if there is lack of an alternative. Network security in the existing system is the motivation factor for a new system with higher-level security standards for the information exchange.

4.2 Proposed System

The proposed system should have the following features. The transactions should take place in a secured format between various clients in the network. It provides flexibility to the user to transfer the data through the network very easily. It should also identify the user and provide the communication according to the prescribed level of security with transfer of the file requested and run the required process at the server if necessary. In this system the data will be

4.3 Feasibility Study

A feasibility study is a high-level capsule version of the entire System analysis and Design Process. The following feasibilities are considered for the project to ensure that the project is available, and it does not have any major obstructions.

4.3.1 Technical Feasibility

It centers on the existing computer system (hardware, software) and to what extent it can support the proposed system.

- Accuracy, ease with which the data can be handled, reliability, and security.
  are the primary concerns and ensured as far as possible.
- The environment required in the development of system is any windows platform.
- The observer pattern along with factory pattern will update the results eventually.
- The language used in the development is JAVA 1.5 and Windows Environment.

4.3.2 Operational Feasibility

The user who wants to avail the features of DATA SECURITY should be able to know the operations that are required to send files across the network with security i.e., to know how to select a file, encrypt a file, embed a file etc.,

This system can be implemented in the organization because there is adequate support from management and users being developed in Java so that the necessary operations are carried out automatically.

4.3.3 Economical Feasibility

If a certain estimated cost for the project is accepted, then we say the system is economically feasible. The system will be developed and operated in the existing hardware and software infrastructure. The system developed and installed will be good benefit to the organization hence the proposed system is economically feasible. So, there is no need of additional hardware and software for the system.

4.4 Technical Analysis

People for a long time have tried to sort out the problems faced in the general digital communication system but as these problems exist even now, secured and easy transfer.

system evolved and came to be known as the Encryption and Decryption of the data and converting the file to video format to be transferred using the cryptographic standards and Steganography. The advantages of Secure Data Transmission through Network are:

- High level Security
- Cost effective transfer.

In this fast-growing world where every individual is free to access the information on the network and even the people are technically sound enough in hacking the information from the network for various reasons. The organizations have the process of information transfer in and out of their network at various levels, which need the process to be in a secured format for the organizational benefits.

If the organizations have these Systems, then each employee can send the information to any other registered employee and thus can establish communication and perform the prescribed tasks in a secure fashion. The video file that the employee sends reaches its destinations within no time in a video file format where the end user needs to de embed the file, decrypt it and use for the purpose. The various branches of the organization can be connected to a single host server and then an employee of one branch can send files to the employee of another branch through the server but in a secured format.

4.5 Functional Model

4.5.1 Use Case Model

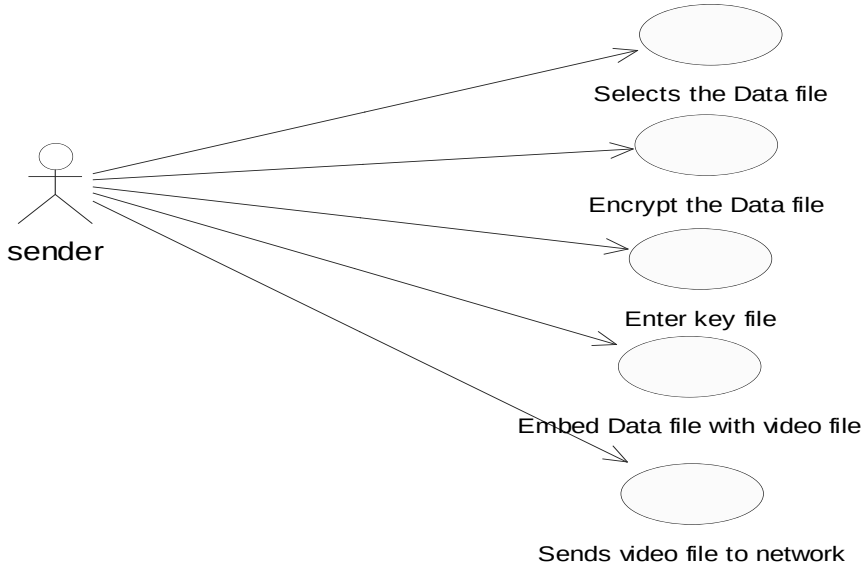


Fig: 4.5.1.1 Use case diagram for Sender

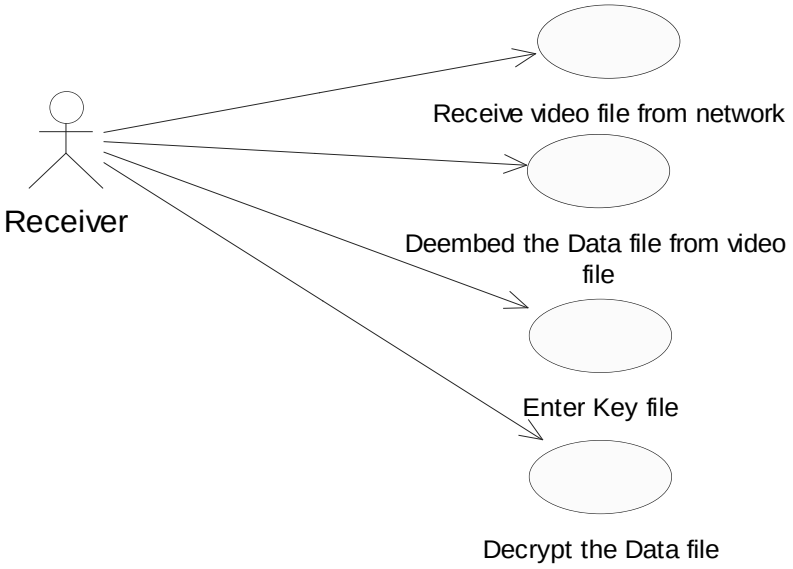


Fig: 4.5.1.2 Use case diagram for Receiver

4.6 Object Model

4.6.1 Class Diagram

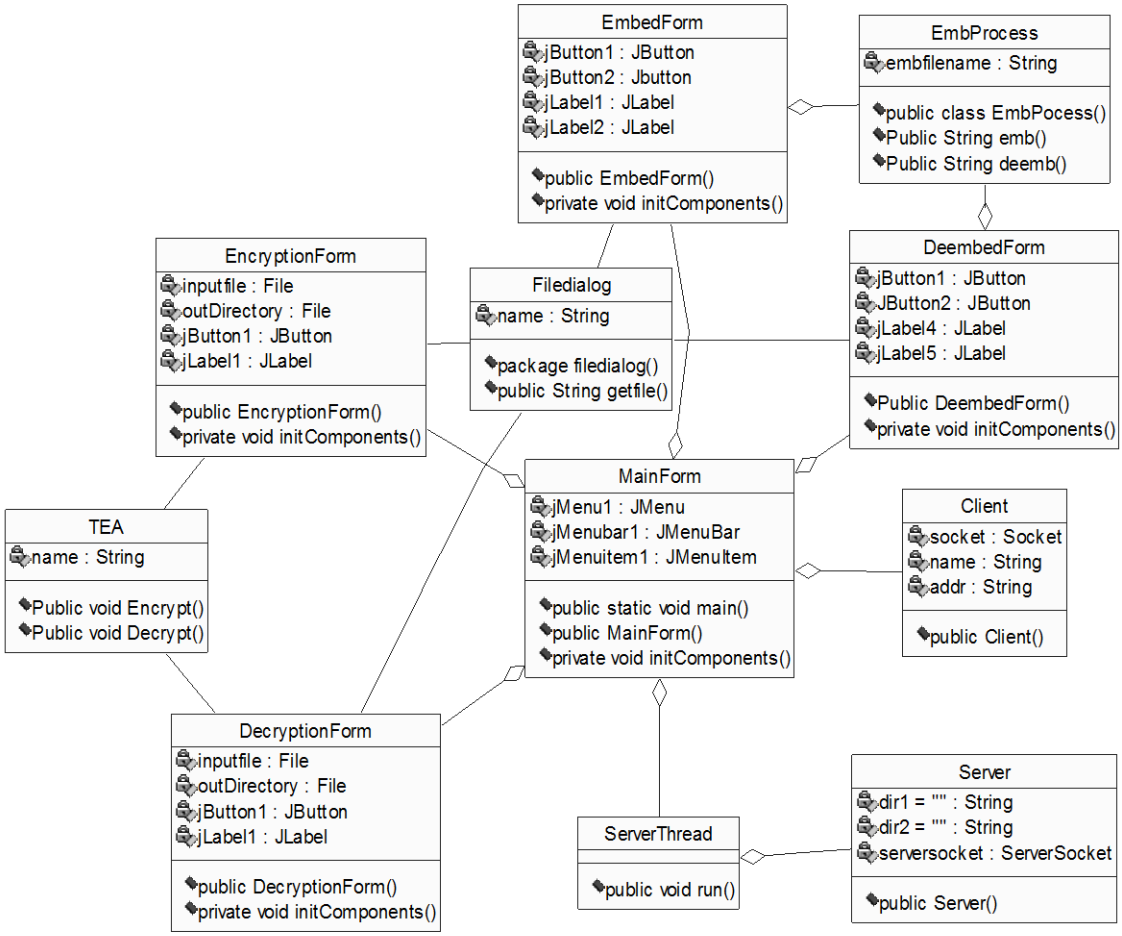


Fig: 4.6.1 Class diagram

4.7 Dynamic Model

4.7.1 Interaction Diagrams

Sequence Diagrams

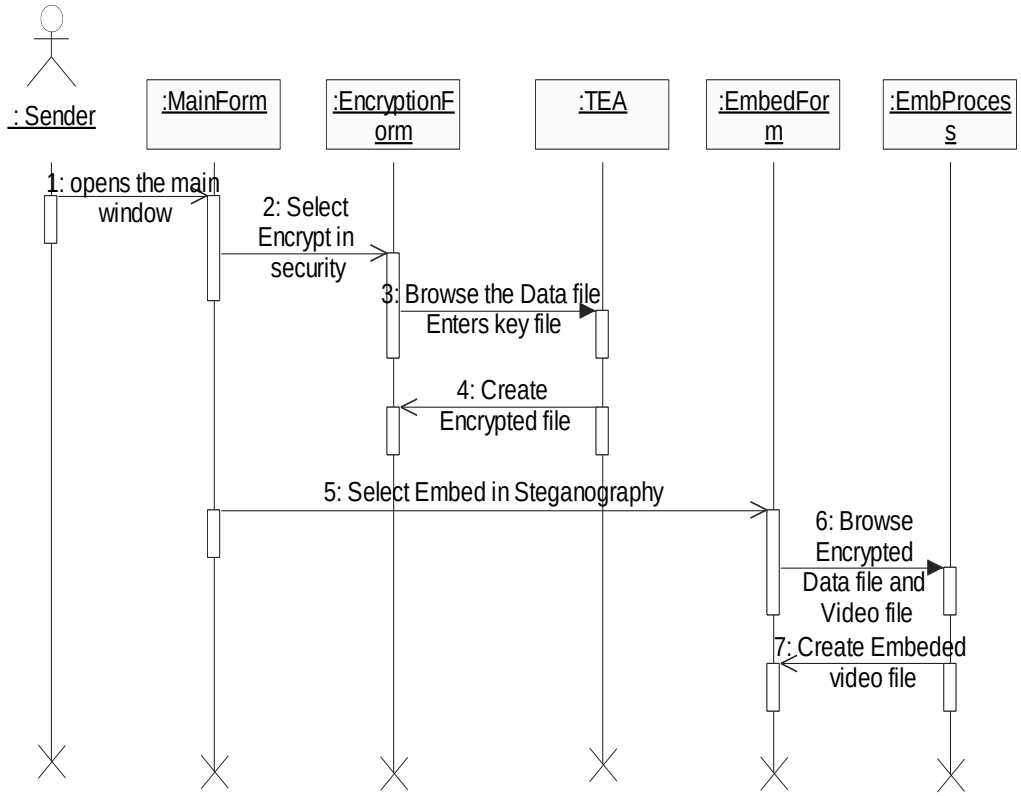


Fig: 4.7.1.1 Sequence diagram for Sender

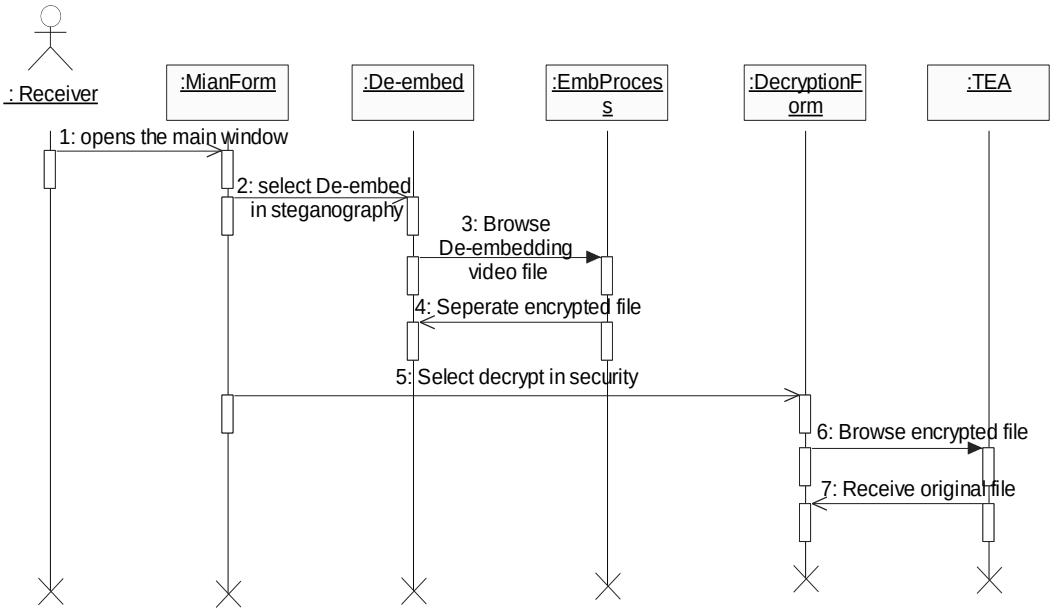


Fig 4.7.1.2 Sequence diagram for Receiver

Collaboration Diagrams

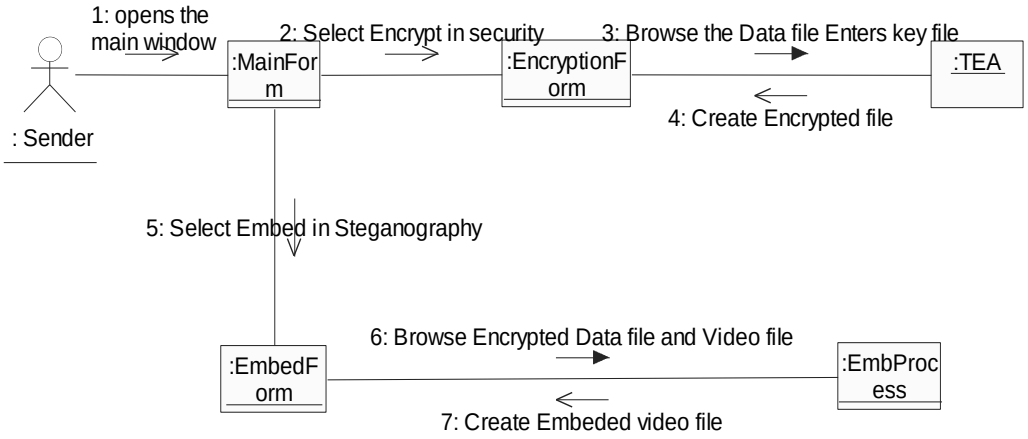


Fig: 4.7.1.3 Collaboration diagram for Sender

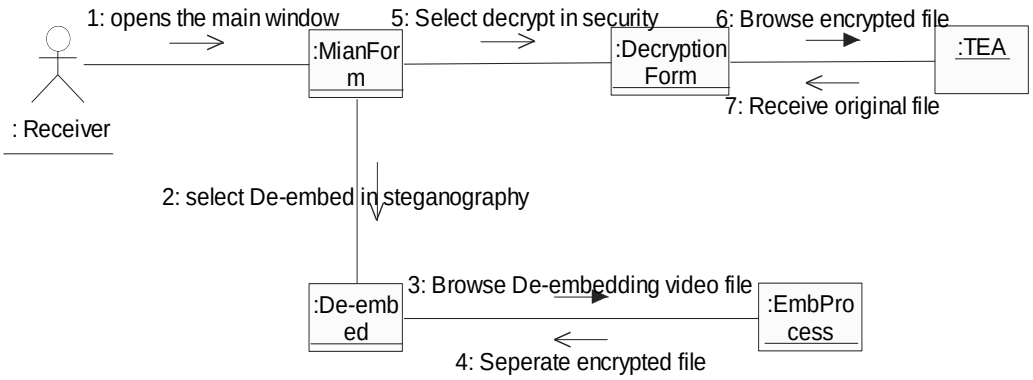


Fig: 4.7.1.4 Collaboration diagram for Receiver

4.7.2 Activity Diagram

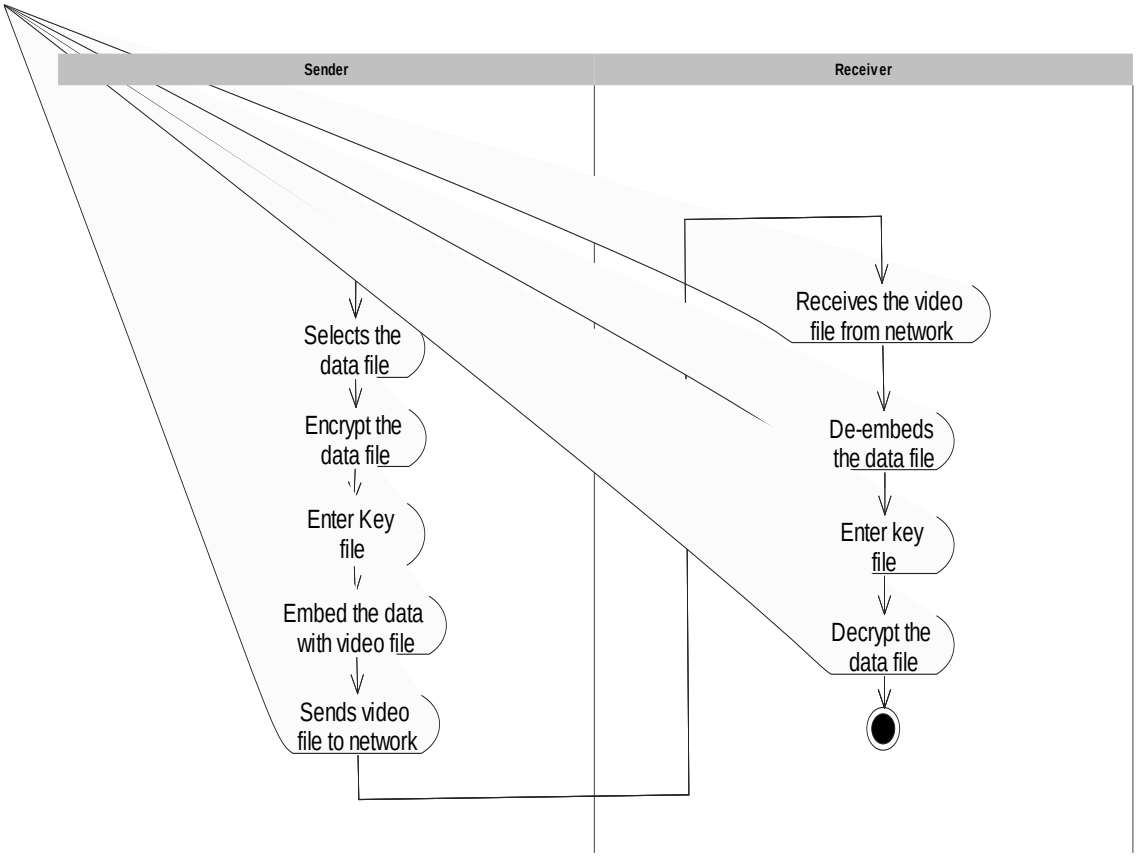


Fig: 4.7.2 Activity diagram

5. LITERATURE REVIEW

5.1 Tiny Encryption Algorithm

The Tiny Encryption Algorithm (TEA) is a cryptographic algorithm designed to minimize memory footprint and maximize speed. It is a Feistel type cipher that uses operations from mixed (orthogonal) algebraic groups. This research presents the cryptanalysis of the Tiny Encryption Algorithm. In this research we inspected the most common methods in the cryptanalysis of a block cipher algorithm. TEA seems to be highly resistant to differential cryptanalysis and achieves complete diffusion (where a one bit difference in the plaintext will cause approximately 32 bit differences in the cipher text) after only six rounds. Time performance on a modern desktop computer or workstation is very impressive.[7]

As computer systems become more pervasive and complex, security is increasingly important. Cryptographic algorithms and protocols constitute the central component of systems that protect network transmissions and store data. The security of such systems greatly depends on the methods used to manage, establish, and distribute the keys employed by the cryptographic techniques. Even if a cryptographic algorithm is ideal in both theory and implementation, the strength of the algorithm will be rendered useless if the relevant keys are poorly managed.

The following notation is necessary for our discussion.

- Hexadecimal numbers will be subscripted with "*h*," e.g., 10 = 16. *h*
- Bitwise Shifts: The logical left shift of *x* by *y* bits is denoted by $x << y$. The logical right shift of *x* by *y* bits is denoted by $x >> y$.
- Bitwise Rotations: A left rotation of *x* by *y* bits is denoted by $x <<< y$. The right rotation of *x* by *y* bits is denoted by $x >>> y$.
- Exclusive-OR: The operation of addition of n-tuples over the field (also known as 2F exclusive-or) is denoted by $x \oplus y$.

The Tiny Encryption Algorithm is a Feistel type cipher that uses operations from mixed (orthogonal) algebraic groups. A dual shift causes all bits of the data and key to be mixed repeatedly.

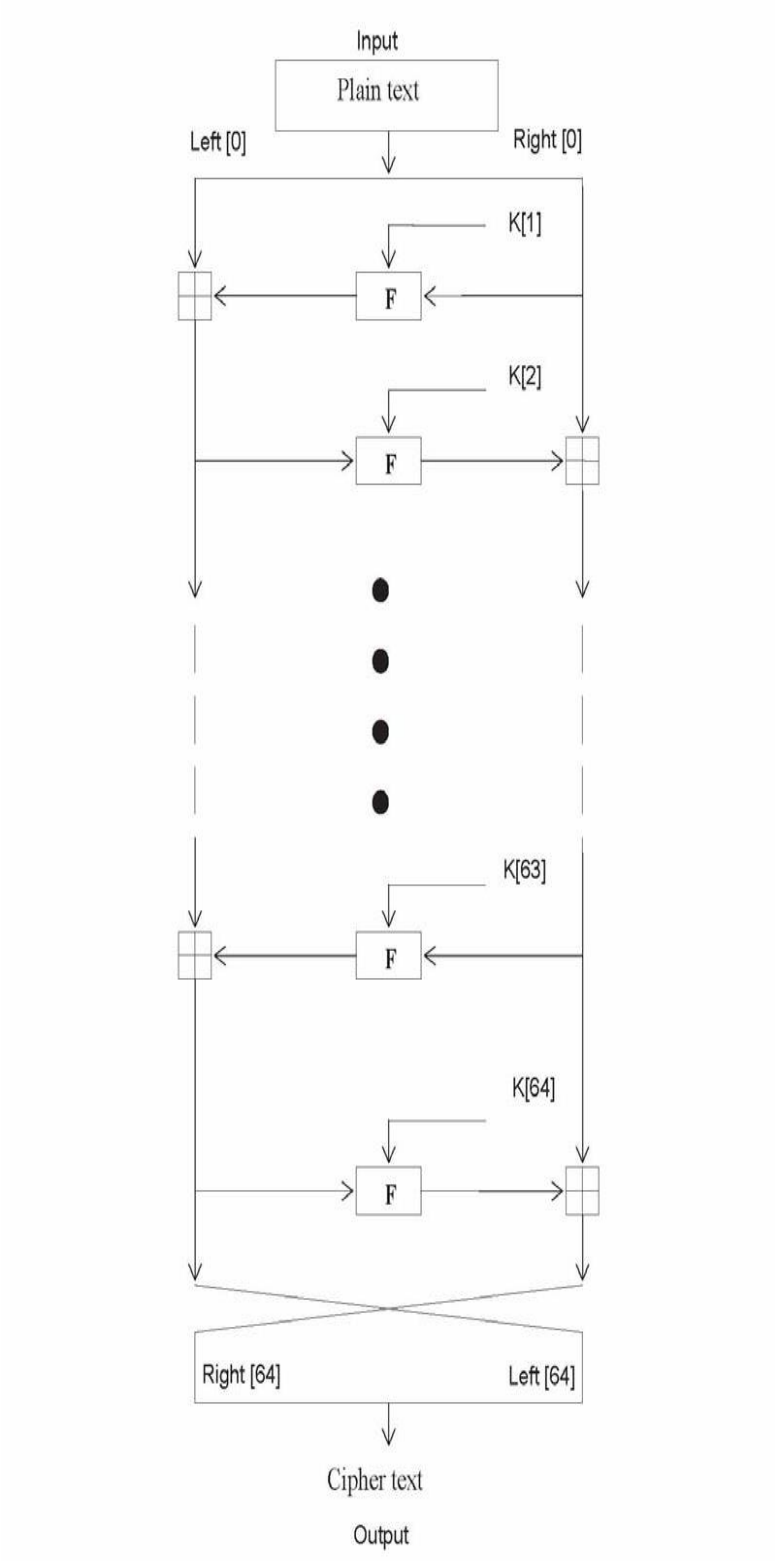


Fig: Diagram for Encryption— [5]

The inputs to the encryption algorithm are a plaintext block and a key K. The plaintext is P = (Left [0], Right [0]) and the cipher text is C = (Left [64], Right [64]). The plaintext block is split into two halves, Left [0] and Right [0]. Each half is used to encrypt the other half over 64 rounds of processing and then combine to produce the cipher text block.

- Each round *i* has inputs Left[*i*-1] and Right[*i*-1], derived from the previous round, as well as a sub key K[*i*] derived from the 128-bit overall K.
- The sub keys K[*i*] are different from K and from each other.
- The constant delta = $(5^{1/2}-1) * 2^{31}$ =9E3779B h , is derived from the golden number ratio to ensure that the sub keys are distinct and its precise value has no cryptographic significance.
- The round function differs slightly from a classical Feistel cipher structure in that integer addition modulo 2^{32} is used instead of exclusive-or as the combining operator.[6]

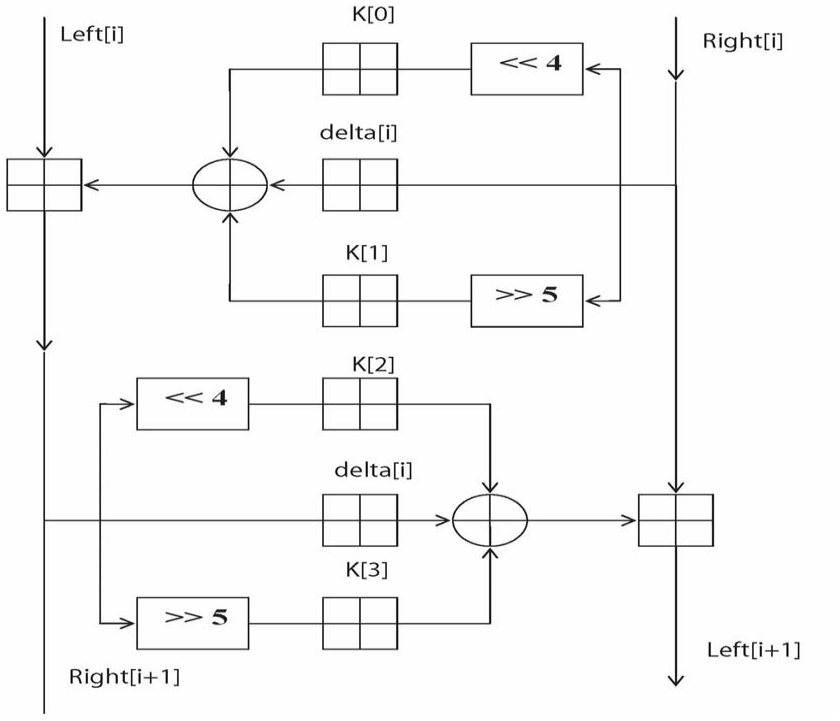


Fig: Diagram for Round Function

Above Figure presents the internal details of the *i*th cycle of TEA. The round function, F, consists of the key addition, bitwise XOR and left and right shift operation.

We can describe the output (Left [i +1], Right [i +1]) of the ith cycle of TEA with the input (Left[i] ,Right[i]) as follows

Left [i+1] = Left[i] F (Right[i], K [0, 1], delta[i]),

Right [i +1] = Right[i] F (Right [i +1], K [2, 3], delta[i]),

delta[i] = (i +1)/2 * delta,

The round function, F, is defined by.

F(M, K[j,k], delta[i]) = ((M << 4) K[j]) $\oplus$ (M delta[i]) $\oplus$ ((M >> 5) K[k]).

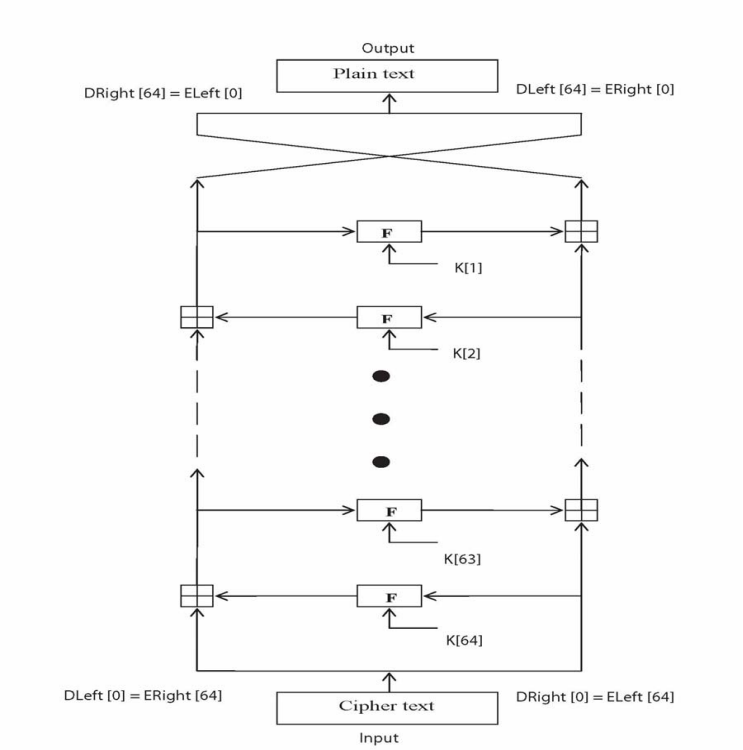
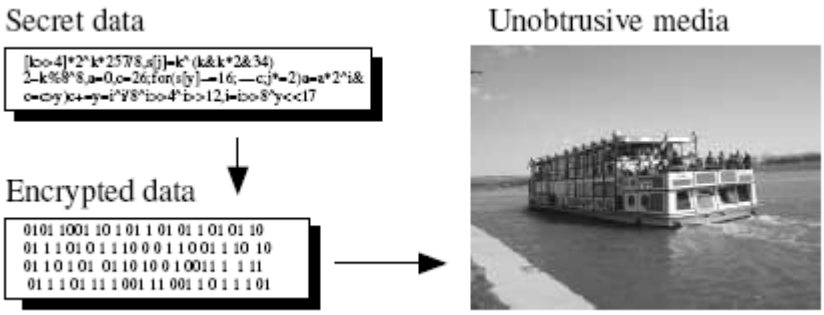


Fig: Diagram for Decryption— [5]

5.2 Steganography

Steganography is an art and science of hiding information within other information. The word itself comes\ from Greek and means. hidden writing. In recent years cryptography has become a very popular science. As steganography is very close to cryptography and its applications, we can with advantage highlight the main differences. Cryptography is about concealing the content of the message. At the same time encrypted data package is itself evidence of the existence of valuable information. Steganography goes a step further and makes the ciphertext invisible to unauthorized users. Hereby we can define steganography as cryptography with the additional property that its output looks unobtrusive. [10,11]

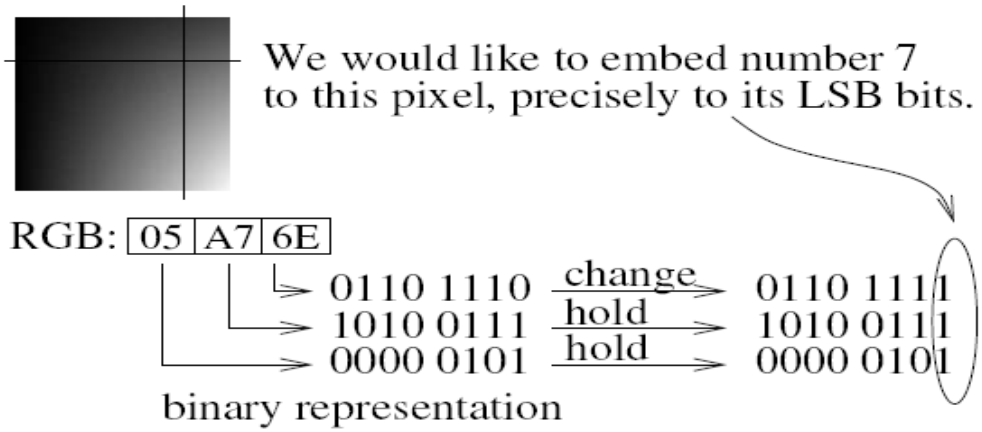


Steganography is the practice of hiding private or sensitive information within something that appears to be nothing out of the usual. Steganography is often confused with cryptology because the two are similar in the way that they both are used to protect important information. The difference between the two is that Steganography involves hiding information, so it appears that no information is hidden at all. If a person or persons views the object that the information is hidden inside of he or she will have no idea that there is any hidden information, therefore the person will not attempt to decrypt the information.
Steganography comes from the Greek words Stefanos (Covered) and Grates (Writing).
Steganography in the modern-day sense of the word usually refers to information or a file that has been concealed inside a digital Picture, Video or Audio file. What Steganography essentially does is exploit human perception, human senses are not trained to look for files that have information hidden inside of them, although there are programs available that can do what is called Steganalysis (Detecting use of Steganography.)

5.2.1 Embedding process

Data which holds effective information often has some redundancies. End users usually tend to think that redundancy is evil which cost extra money, as more disk space or network bandwidth is needed. Well, they are partially right, but optimal compression hardly ever exists. Moreover, common compress ratio is mostly a question of efficiency.

Now we know, there are almost always a few bytes one can play with, without destroying carried information. Least Significant Byte (LSB) substitution is a well-known and widely used method. Take for example a True-Color BMP image _le format. The color of pixel is coded in 3-byte array of indices to RGB palette. If you change only LSB bit in each color element, then the picture will seem still the same, but is not. It carries hidden information. A picture with size 120x100 pixels can hold approximately up to 4500B of hidden data if this method is used.



5.2.2 How Does It Work?

There are numerous methods used to hide information inside of Picture, Audio and Video files. The two most common methods are **LSB (Least Significant Byte)** and **Injection.**
I will discuss these two methods below.

5.2.2.1 Substitution - Altering/Replacing the LSB

When files are created there are usually some bytes in the file that aren't really needed, or at least aren't that important. These areas of the file can be replaced with the information that is to be hidden, without significantly altering the file or damaging it. This allows a person to hide information in the file and make sure that no human could detect the change in the file. The LSB method works best with Picture files that have a high resolution and use many different colors, and with Audio files that have many different sounds and that are of a high bit rate. The LSB method usually does not increase the file size, but depending on the size of the information that is to be hidden inside the file, the file can become noticeably distorted.

5.2.2.2 Injection

Injection is quite a simple method which simply involves directly injecting secret information into the carrier file. The main problem with this method is that it can significantly increase the size of the carrier file .

5.2.3 Steganography in Video

When information is hidden inside video the program or person hiding the information will usually use the DCT (Discrete Cosine Transform) method.
DCT works by slightly changing each of the images in the video, only so much though so it isn’t noticeable by the human eye. To be more precise about how DCT works, DCT alters values of certain parts of the images, it usually rounds them up.
For example, if part of an image has a value of 6.667 it will round it up to 7.
Steganography in Videos is like that of Steganography in Images, apart from information is hidden in each frame of video. When only a small amount of information is hidden inside of video it generally isn’t noticeable at all, however the more information that is hidden the more noticeable it will become.

6. DESIGN

6.1 System Design

6.1.1 Introduction

The System Design includes the maintenance of the secure file transfer service with a prescribed encryption format and split at the interested level of encryption and embed process and the receiving service at the other end with de-embed and decryption process. The design also includes the provision of facility to the user to manipulate the concerned information according to his personal use and communication process. The design of our system basically involves the interface architecture, Security services, and communication system.

In the interface design we involve with the design of the user interface with GUI standards and a proper navigation system where the user needs to enter the flow of transactions authorization services are check and further access is provided into the system. Then the user needs to select into the operations provided through the GUI where encryption, embedding, de-embedding, Decryption, and sending of the file, General Information are provided.

Here the Encryption and decryption and services are provided connecting to the security services module where the encryption and decryption are carried out using the cryptographic standards implementing the tiny algorithm. After the decryption process is completed, the user selects the file for encryption. After encryption, the file is completed, the user is to select the file for embedding it to the video file and sending through the network to the desired user by specifying the targeted user's system IP address in the panel designed. Then the system gets connected to the targeted user and delivers the file in video format after which the user working with the software should go for the option De-Embed Files and decrypt the file by selecting the file path by which the file gets decrypted the file and is viewed on the system.

6.1.1.1 Purpose

- Implementation of Tiny Encryption Algorithm with input plain text and key length of 128-bit.
- Implementation of steganography using Input/Output packages to provide security.
- Achieving SECRECY.

6.1.1.2 Design Goals

- Designing user interface and sending requests to sender and receive requests from receiver.
- To perform operations such as encryption to convert the data into unreadable form.
- To perform operations such as embedding to hide the data.
- To perform operations such as de-embed to take the data out of video file.
- To perform operations such as decrypt to convert the data into the original format.

6.1.1.3 Overview

The most important and criteria phase in the software life cycle is the design phase. The design process involves developing of the conceptual view of the system, establishing the system structures, identifying data stores, decomposition high level functions into sub functions, establishing connections between individual programs, interconnecting among components, and developing concrete data representation.

This is the step while developing any product after analysis. The goal of this is to produce a model of the entities involved in the project which later needs to be built. The representations of the entities that are to be used in the product being developed need to be designed. Software design is a process that gradually changes various new, better, and more complete methods with a broader understanding of the whole problem in general come into existence. There are various kinds of methods in software design.

6.1.2 Software System Overview

6.1.2.1 Component Diagram

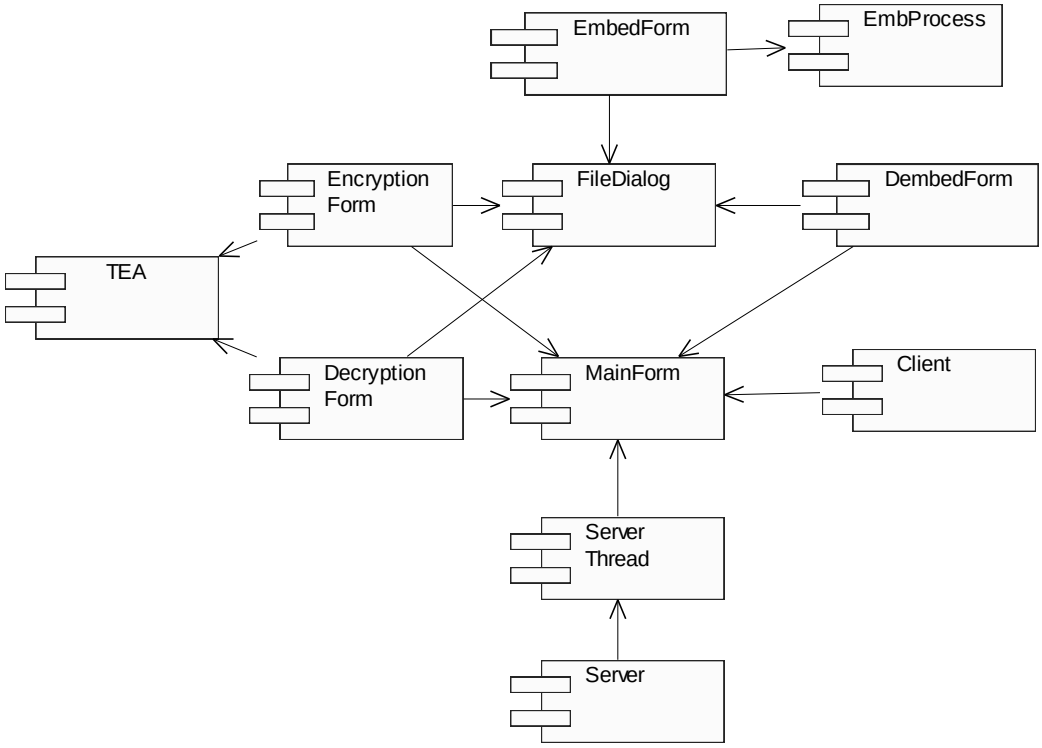


Fig: 7.1.2.1 Component diagram

6.1.2.2 Deployment Diagram

Fig: 7.1.2.2 Deployment diagram

7. IMPLEMENTATION

7.1 Software Overview

About Java

Initially the language was called "oak", but it was renamed as "Java" in 1995. The primary motivation of this language was the need for a platform-independent language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer's language.
- Java is cohesive and consistent.
- Except for those constraints imposed by the Internet environment, Java gives the programmer full control.

Finally, Java is to Internet programming where C was to system programming.

Applications and Applets

An application is a program that runs on our computer under the operating system of that computer. It is like one creating using C or C++. Java's ability to create Applets makes it important. An Applet is an application designed to be transmitted over the Internet and executed by a Java –compatible web browser. An applet is a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

About Swings

The java Foundation classes, or JFC, are a collection of APIs for developing graphical user interfaces.

The java foundation classes include the following APIs.

Abstract window toolkit.

2D API

Swing Components

Accessibility API

The Abstract window tool kit, or AWT, is java's original tool kit for developing user interfaces.

The 2D API provides additional graphical capabilities that are lacking in AWT.

Swing is a set of mostly lightweight components built on top of the AWT. Swing provides lightweight replacements for the AWT's heavy weight components, in addition to a multitude of additional components that the AWT lacks.

The Accessibility API consists of a set of classes that enable Swing components to interact with assertive technologies for uses with disabilities.

Lightweight Vs Heavyweight components

Originally, the AWT included only heavyweight components that were associated with a native peer component and were rendered in their own native, opaque window.

Lightweight components, on the other hand, do not have a native peer and are rendered in their heavyweight container's window.

Some of the Swing components are.

Button – is a push button that is a replacement for java.awt. button. Instances of Button, like swing labels, can display text, an icon, or both. Like AWT buttons, swing buttons fire action events and they are activated.

Constructors

public Button ()

public Button (String, Icon)

Checkbox – checkboxes fire action events when activated, property change events their bound properties are modified.

Constructors

public Checkbox ( )

public Checkbox (String, Icon, Boolean selected)

Menu – Swing Menus are essentially buttons that have a popup menu associated with them. When a menu is activated, it's popup menu is displayed beneath the menu.

Opiomania – Option Panes are components that are meant to be placed in a dialog box. Option panes can display an icon, a message, one or more selectable values, and a row of buttons.

Constructors

public Opiomania ()

public Opiomania (object message, int message Type, int Option Type, Icon icon, Object [] options, Object initial value)

Intraframe – Internal frames are frames because there are facsimiles of external frames; they are internal because they are contained within another Swing Container, usually a Desktop Pane.

Constructors

public Intraframe ().

public Estoppage ().

Spodumene – Popup menus can also be used outside the content of a menu; a popup menu can be displayed anywhere within a component or relative to the screen.

Constructors

public Spodumene ().

public Spodumene (String).

Passworded – Swings password field conceals its text by displaying an '*' for every character entered in the field.

Constructors

public Passworded (); public Passworded (Document, String, int).

8. TESTING

No system is said to be perfect until it compromises the real time environment. Hence it is our duty to check the individual system separately and must integrate to see whether the system is working perfectly up to the expectations of the user.

Testing is the major quality measure employed during software engineering development. Its basic function is to detect errors in the software. Testing is necessary for the proper functioning of the system. A good test case is one that has a high probability of finding an undiscovered error.

During the requirement analysis and design, the output is a document that is usually textual and non-executable. After the coding phase, computer programs are available that can be executed for testing purposes. This implies that testing not only has to uncover errors introduced during coding, but also errors introduced during the previous phases. Thus, the goal of testing is to uncover requirements, design or coding errors in the programs.

Testing cannot show the absence of defects, it can only show that software defects are present. It is important to keep this statement in mind as testing is being conducted. Any product can be tested in one of these ways knowing the specific function that a product has designed to perform; a test can be conducted that demonstrate each function is fully operational. This approach is called ‘black box testing’.

The starting point of testing is unit testing. In this a module is tested separately and is often performed by the coder himself simultaneously with the coding of the module. The purpose is to exercise the different parts of the module code to detect coding errors. After this the modules are gradually integrated into subsystems, which are then integrated themselves to eventually from the entire system. During integration of modules, ***integration testing*** is performed. The goal of this testing is to detect design errors, while focusing on testing the interconnection between modules.

After the system is put together, ***system testing*** is performed. Here the system is tested against the system requirements to see if all the requirements are met and the system performs as specified by the requirements.

8.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design module. All the modules in this system are tested under this strategy of unit test. In this each line of the code of all the classes were tested. We encountered several general syntax errors, which we corrected during the compilation of the classes.

ReceiverClient. java

```
package Server.
import java.awt.FileDialog;
import java.io. *;
import java.net.Socket;
import java.net.UnknownHostException;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
public class ReceiverClient extends Thread.
{
   Socket socket;
   public ReceiverClient()
   {
     addr = JOptionPane.showInputDialog("Enter IP address of Receiver: ");
        name = "";
        if(addr.equals(""))
      {
OptionPane.showMessageDialog(null, "Enter IP address of Receiver", "Message", 1);
       }
      else
       {
```

```
try
    {
    FileDialog filedialog = new FileDialog(new JFrame(), "select the File", 0);
        filedialog.setFile("*.*");
        filedialog.show();
        if(filedialog.getFile() != null)
            name=(new
StringBuilder()).append(filedialog.getDirectory()).append(filedialog.getFile()).toString();
 System.out.println((new StringBuilder ()). append("name"). append(name).toString());
        File file = new File(name);
          if(file.exists())
        {           if(JOptionPane.showConfirmDialog(null,"Confirm the host
"+addr)==0){
       dir=name.substring(name.lastIndexOf("\\")+1,name.lastIndexOf("."));
            System.out.println("dir==="+name);
            byte abyte1[] = new byte[name.getBytes().length];
            socket = new Socket(addr, 2222);
            System.out.println("dir123==="+name);
            abyte1=name.getBytes();
             java.io.OutputStream outputstream = socket.getOutputStream();
            outputstream.write(abyte1);
            FileInputStream fileinputstream = new FileInputStream(file);
            byte abyte0[] = new byte[fileinputstream.available()];
            outputstream = socket.getOutputStream();
            for(int i = 0; i < abyte0.length; i++)

    abyte0[i] = (byte)fileinputstream.read();
```

```
            ByteArrayOutputStream bytearrayoutputstream = new ByteArrayOutputStream();
                    bytearrayoutputstream.write(abyte0,0, abyte0.length);
                    bytearrayoutputstream.writeTo(outputstream);
 JOptionPane.showMessageDialog(null, "File Sending in process", "Message", 1);
socket.close();             }
                  } else
                  {
                   JOptionPane.showMessageDialog(null, "File not found", "Message", 1); }
}
                  catch(UnknownHostException unknownhostexception)
                  {
                 JOptionPane.showMessageDialog(null, "Host not found", "Message", 1);
                  }
                  catch(IOException ioexception)
                  {          ioexception.printStackTrace();
          OptionPane.showMessageDialog(null,"Unable to send the file","Message", 1);
                  }
            catch(Exception e)
                  {          e.printStackTrace();
                            JOptionPane.showMessageDialog(null, e, "Message", 1);       }
                } }
                  String addr;
                  String name;
          }
```

TEST REPORT

S.No	Test Case	Expected Output	Result
1.	Variables	All variables are declared before using them.	Pass
2.	Classes in Different Packages	Required Built-in classes are imported from the packages.	Pass
3.	Syntax Errors	Syntax Errors are Eliminated.	Pass

8.2 Black Box Testing

Test Cases

Test Case 1: Encryption

Input	1. Select the data file and enter the key. 2. Click on the Encrypt button when the data file is not selected or when the null value is entered. 3. Click on the Ok button when the key is not entered.
Result	1. Displays the "Your file has been encrypted and saved" message. 2. Error message "Select the file" is displayed. 3. Error message "File decryption failed" is displayed.
Condition	You should select the data file and the key must be entered.

Test Case 2: Embeding encrypted file.

Input	1. Select encrypted file and video file. 2. Click on embed button when the file is not selected or when the null value is entered.
Result	1. Displays the "Embed process completed" message. 2. Error message "Embed process failed" is displayed.
Condition	You should select the embeding encrypted file and video file.

Test Case 3: Send file

Input	1. Enter the correct IP address of receiver. 2. Click on Ok button when the address is not entered or when the null value is entered. 3. Enter the wrong IP address of receiver.
Result	1. Displays the “File successfully sent” message. 2. Error message “Enter the IP address of receiver” is displayed. 3. Displays the “Unable to send the file” message.
Condition	You should enter the correct IP address of receiver.

Test Case 4: De-embeding Encrypted file.

Input	1. Select the video file. 2. Click on De-embed button when the file is not selected or when the null value is entered.
Result	1. Displays the “De-embed process completed” message. 2. Error message “De-embed process failed” is displayed.
Condition	You should select the De-embeding Encrypted file.

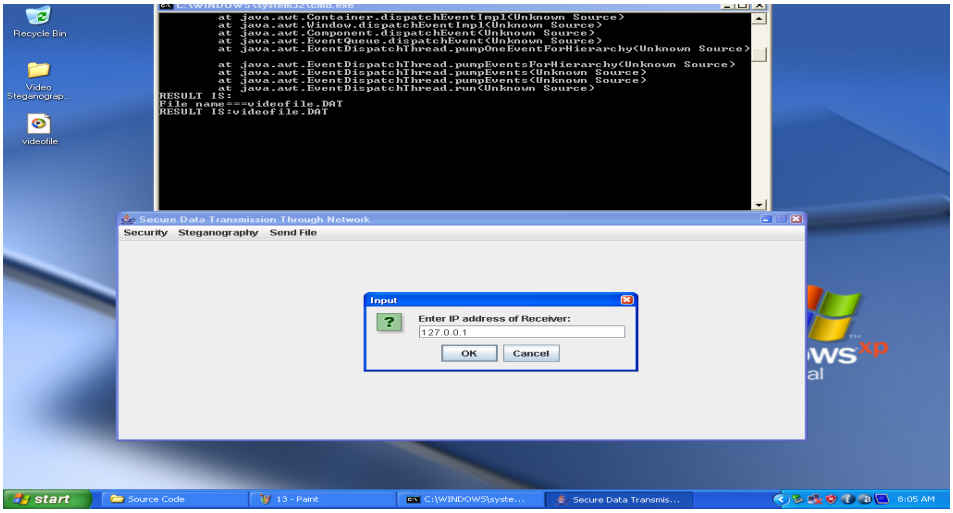
Test Case 5: Decryption

Input	1. Select the file and enter the key. 2. Click on the decrypt button when the file is not selected or when the null value is entered. 3. Click on Ok button when the correct key is not entered.
Result	1. Displays the “Your file has been decrypted and saved” message. 2. Error message “Select the file” is displayed. 3. Error message “File decryption failed” is displayed.
Condition	You should select the file and enters the correct key.

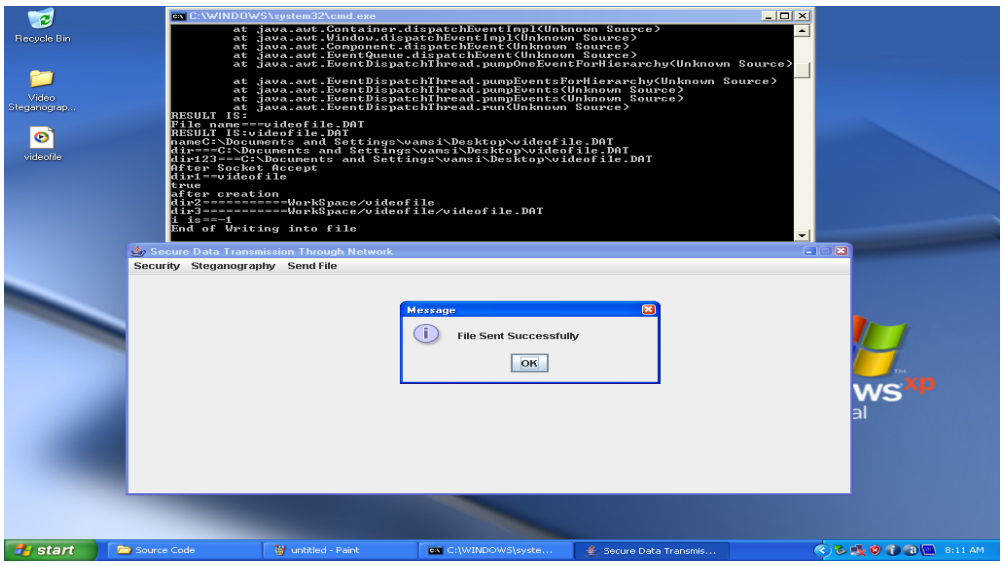
8.3 Functional Testing

Specifies the operating condition, input values and expected result. The function should be designed to take care of the situation. Performance tests should be designed to verify response time, throughput and secondary memory utilization.

- Send file is tested, whether it can send files across the network when we enter the address of the receiver.



- File Successfully Sent



8.4 Integration Testing

Test Cases

Test Case1: Security Module

Input	Expected Behavior	Observed Behavior	Status P=Passed F=Failed
Encryption	Encrypts the given data file and saved.	Encrypts the given data file and saved.	P
Decryption	Decrypts the data file and saved.	Decrypts the data file and saved.	P

Test Case 2: Steganography Module

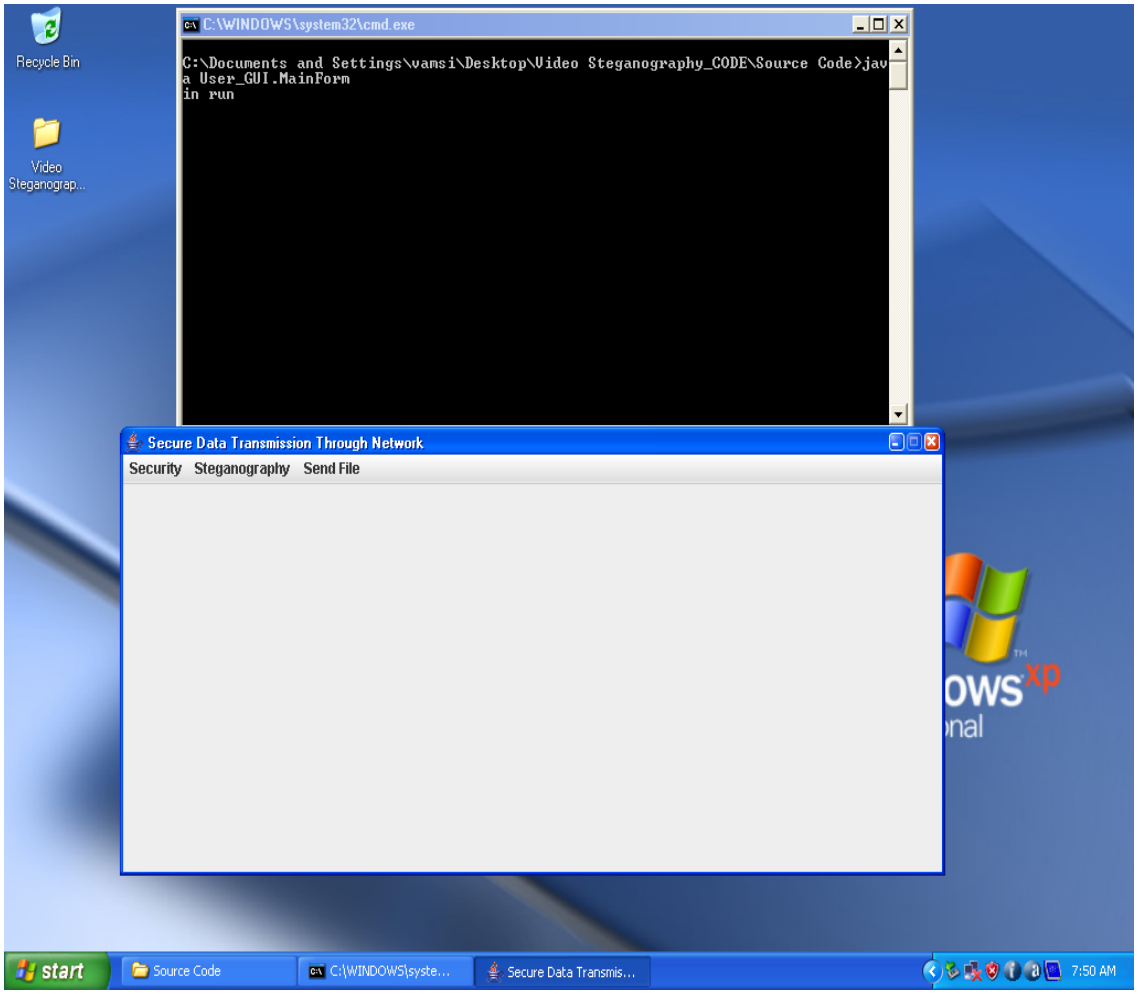
Input	Expected Behavior	Observed Behavior	Status P=Passed F=Failed
Embedding Encrypted file	Embed the encrypted file in a video file.	Embed the encrypted file in a video file.	P
De-embedding Encrypted file	De-embed the encrypted file from the video file	De-embed the encrypted file from the video file	P

Test Case 3: Send File

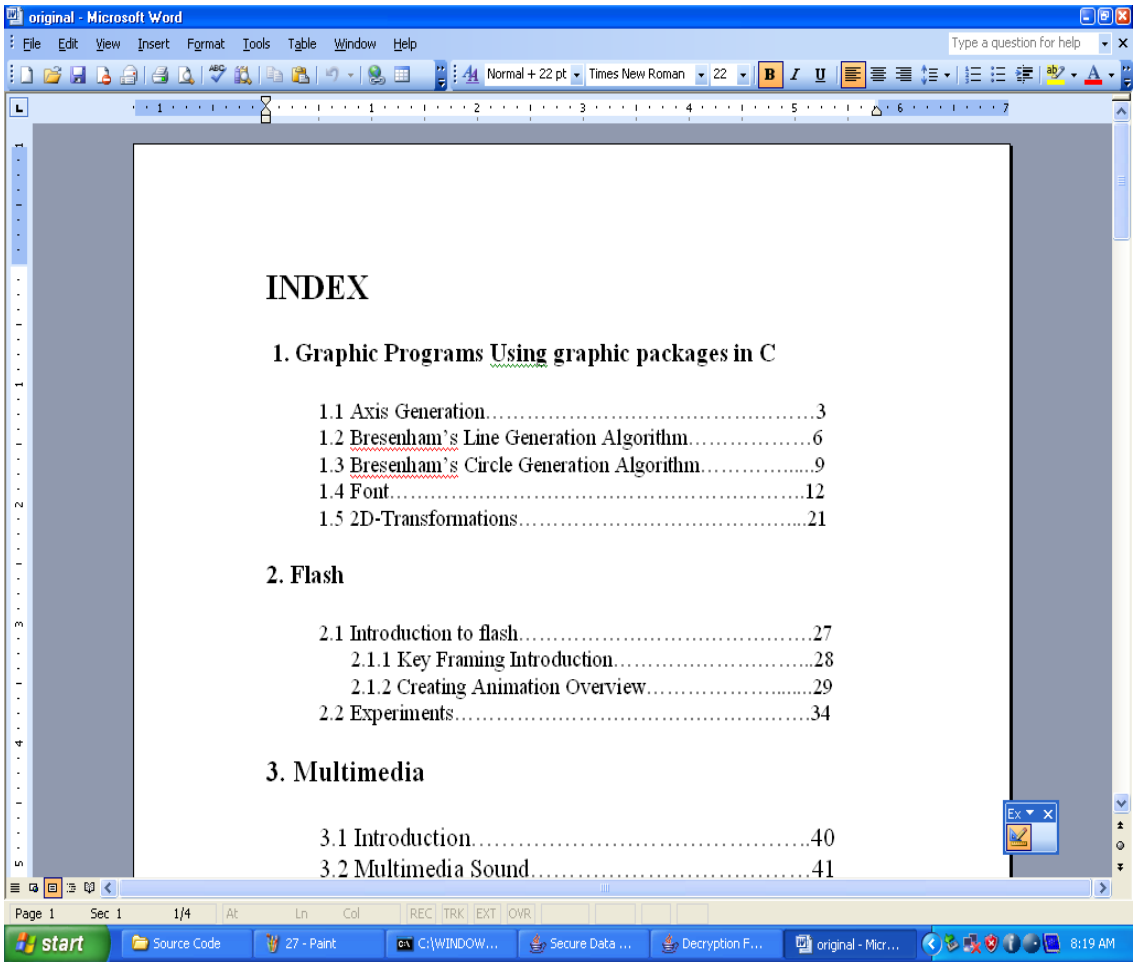
Input	Expected Behavior	Observed Behavior	Status P=Passed F=Failed
Send File	File will be successfully send across the network	File will be successfully send across the network	P

9. OUTPUT SCREENS

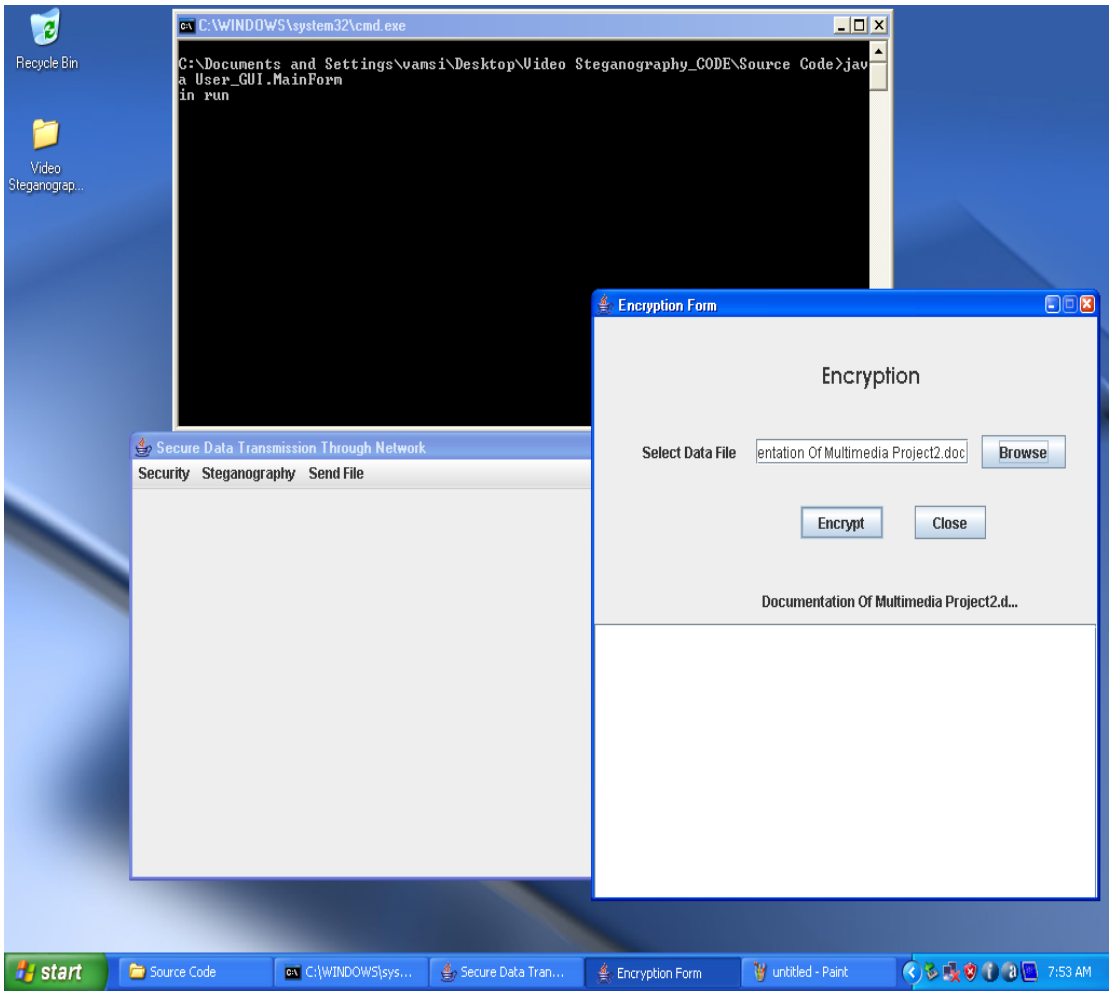
Main Form



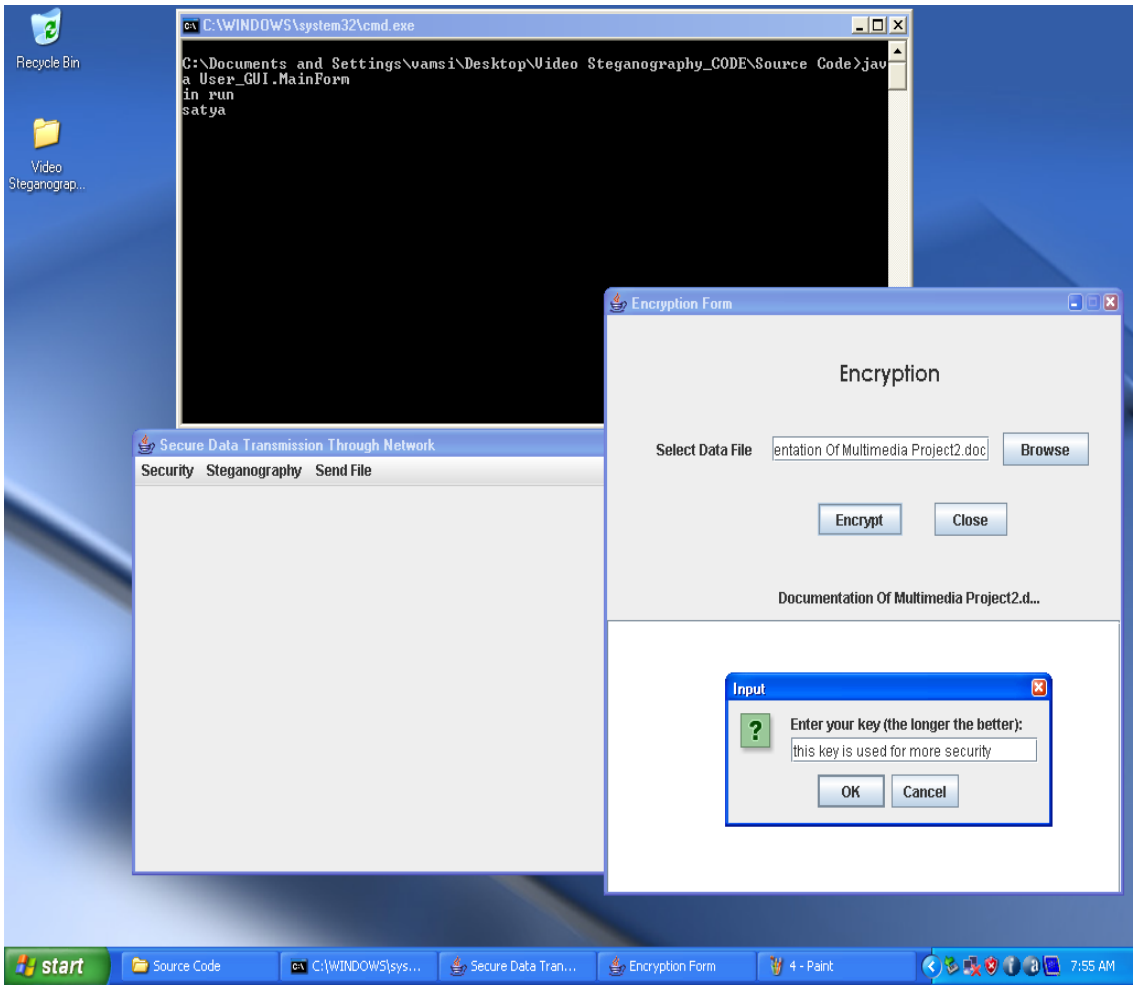
Original Data File



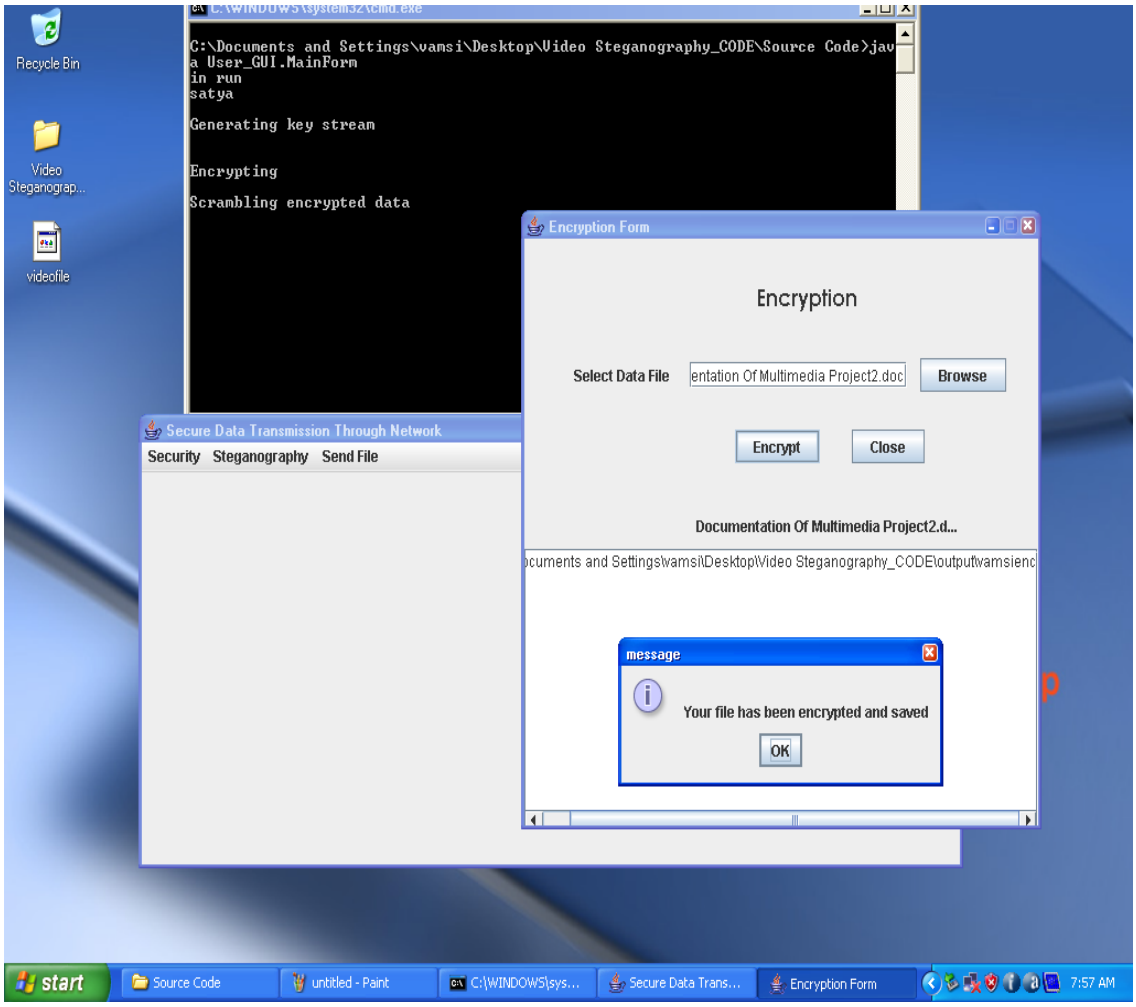
Selecting Data File



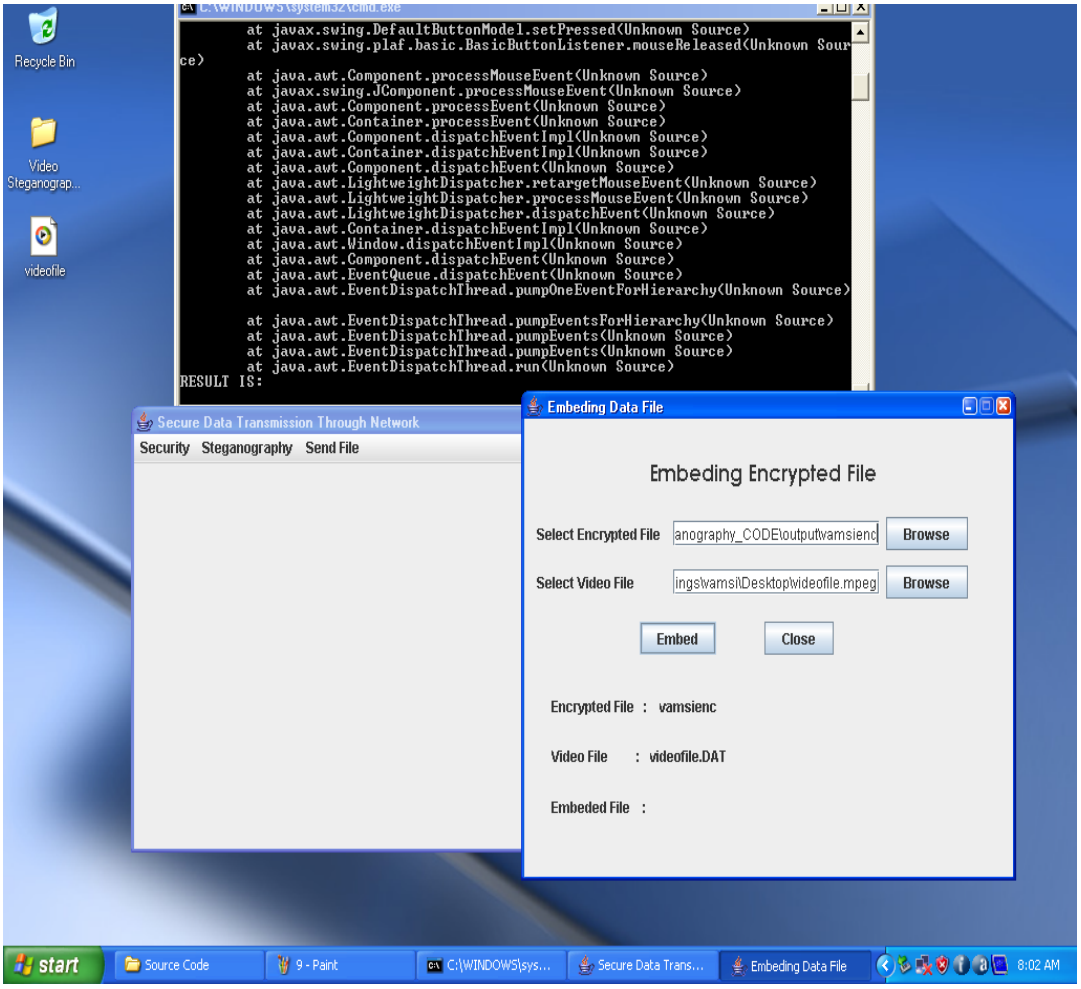
Entering the key during encryption process



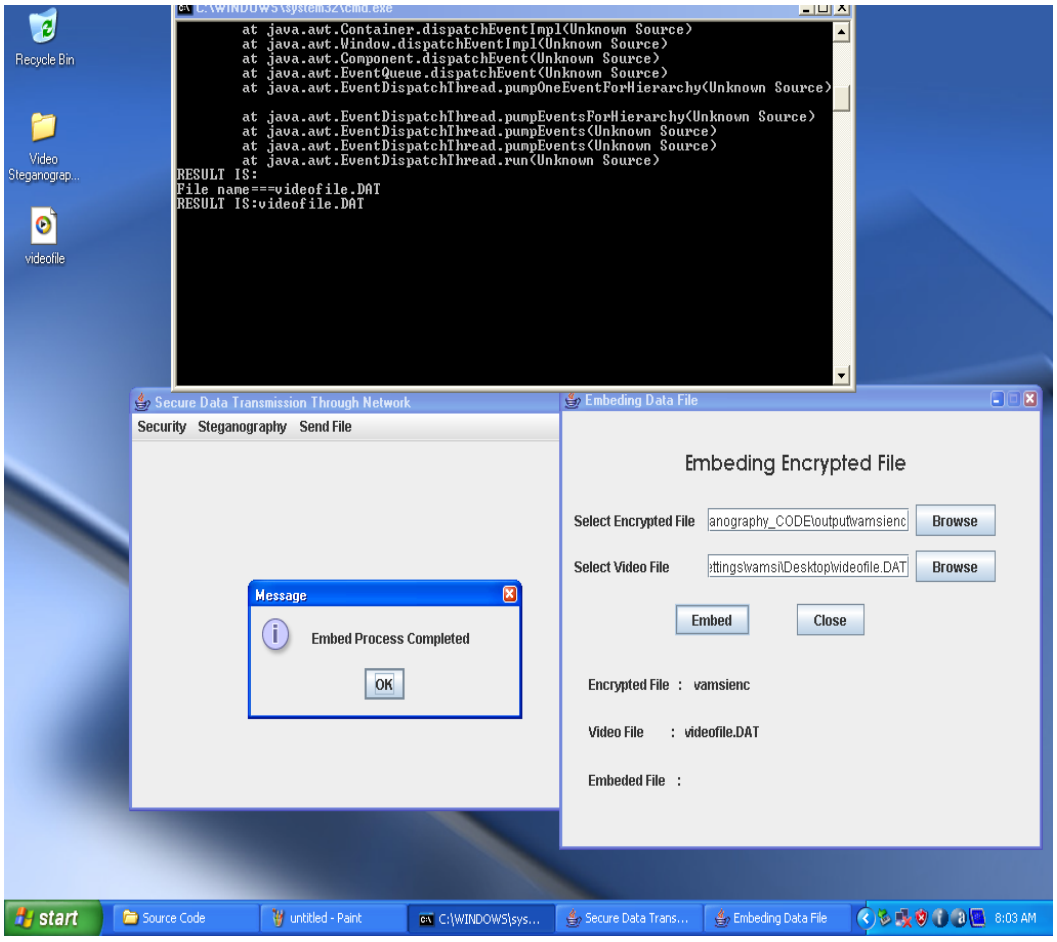
File Successfully Encrypted



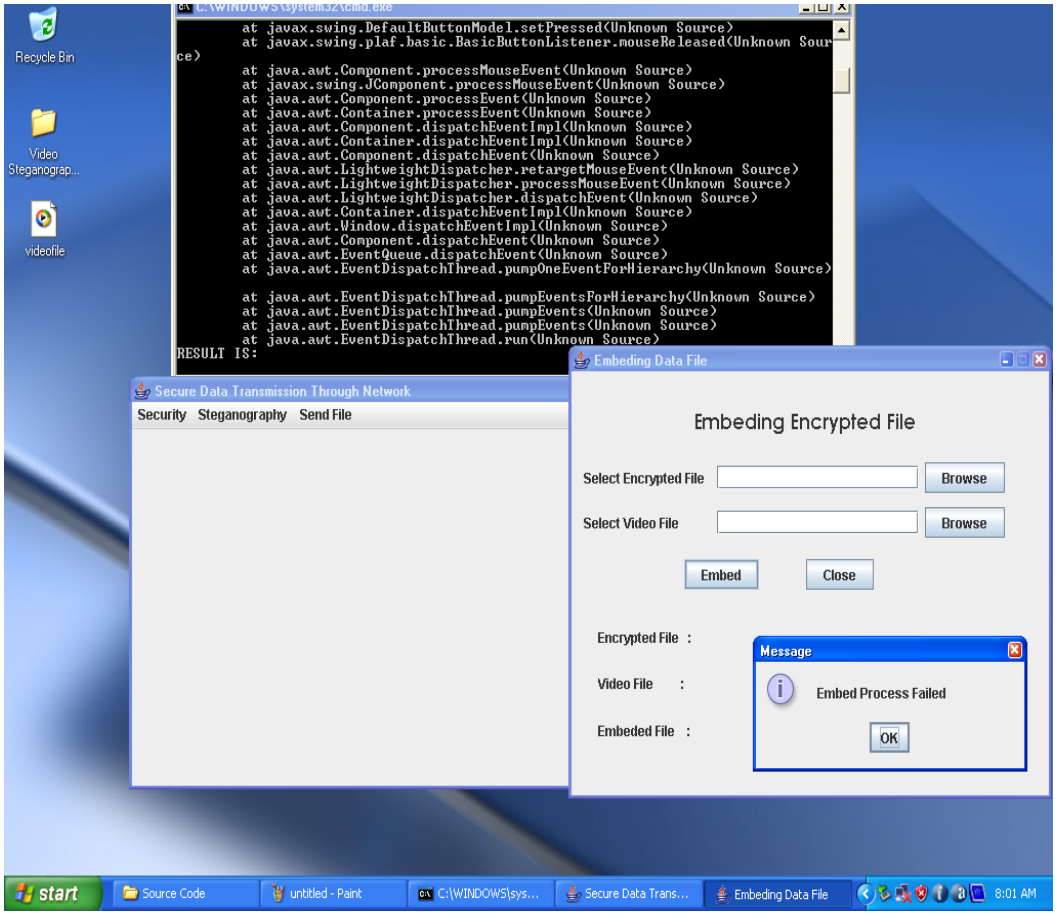
Embedding Encrypted File



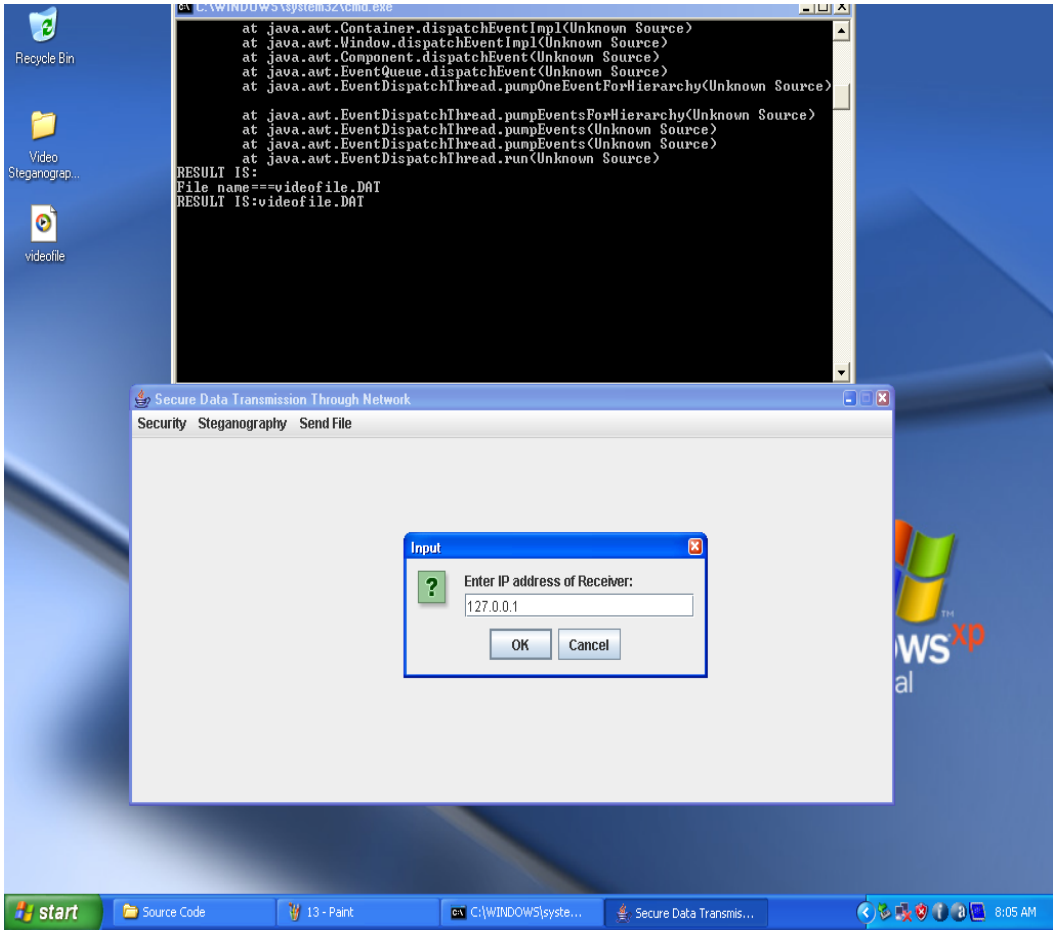
File Successfully Embedded



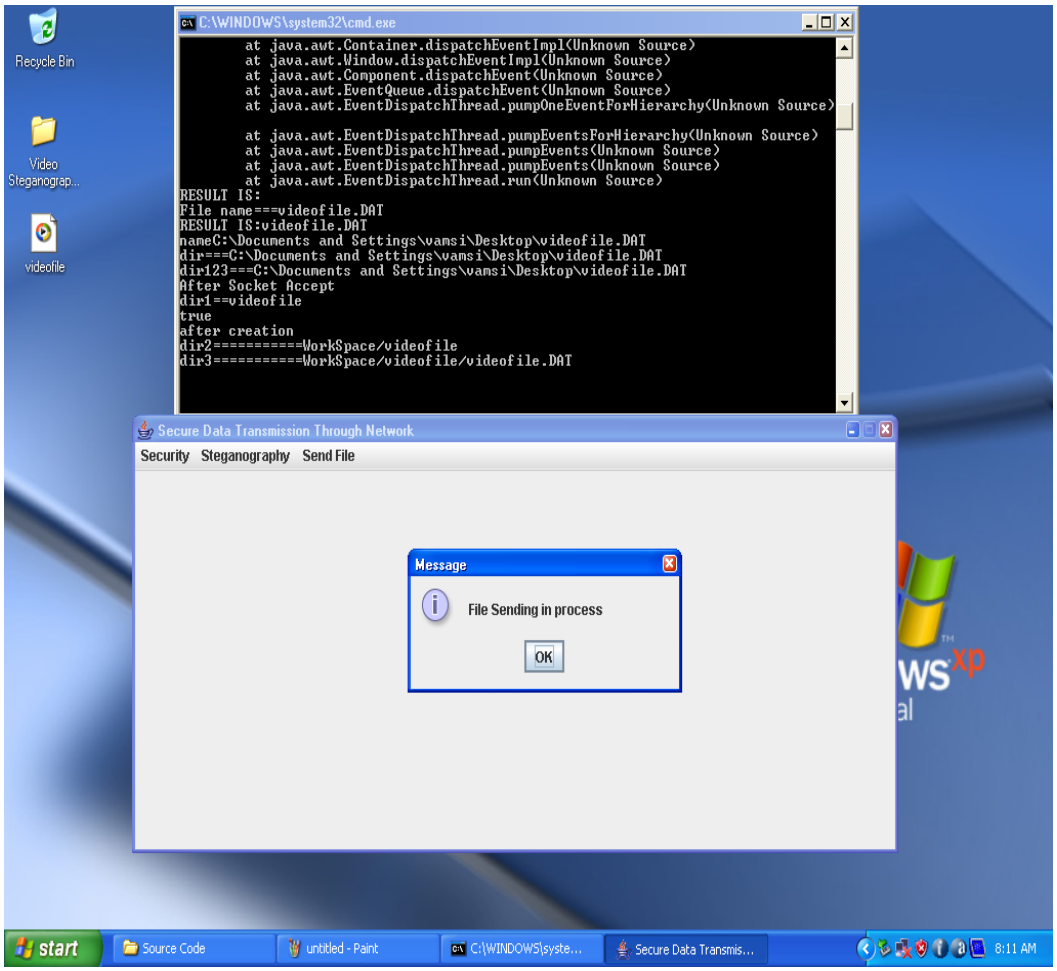
If we enter Null values



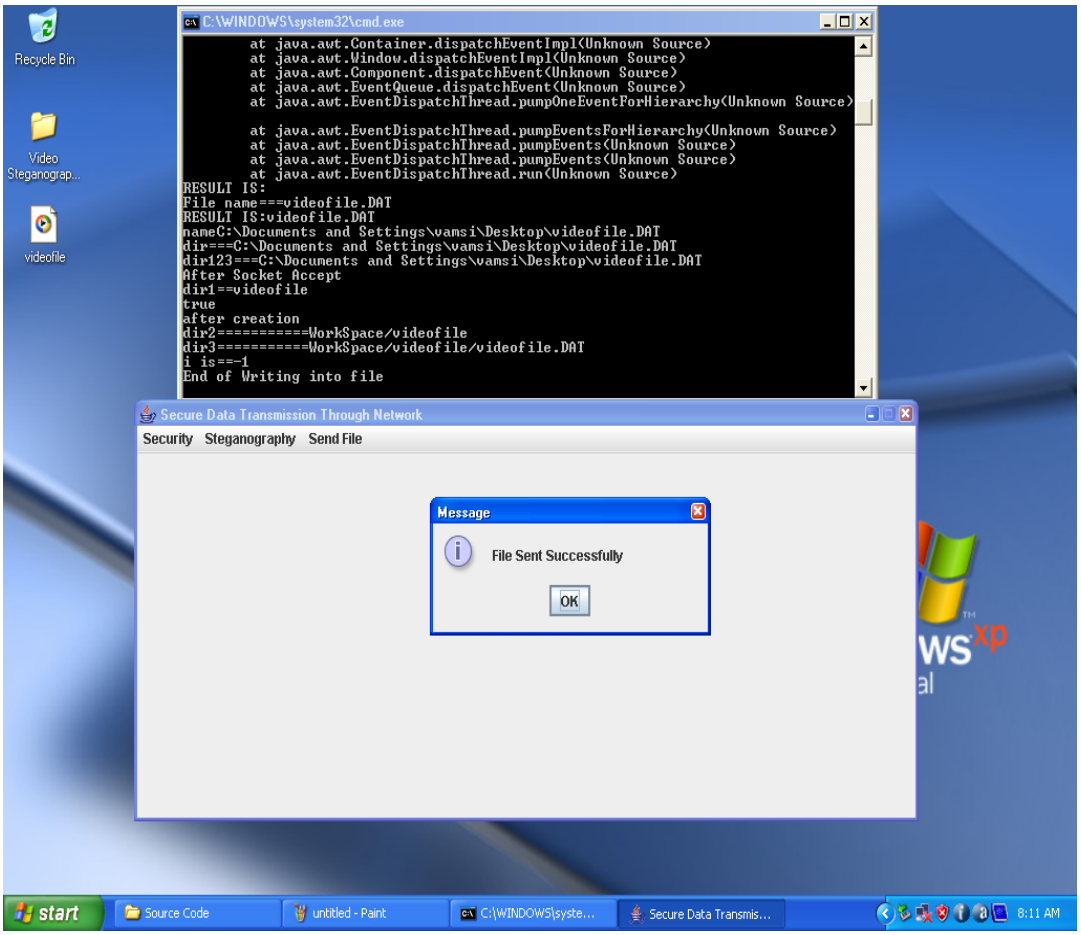
Entering address of Receiver



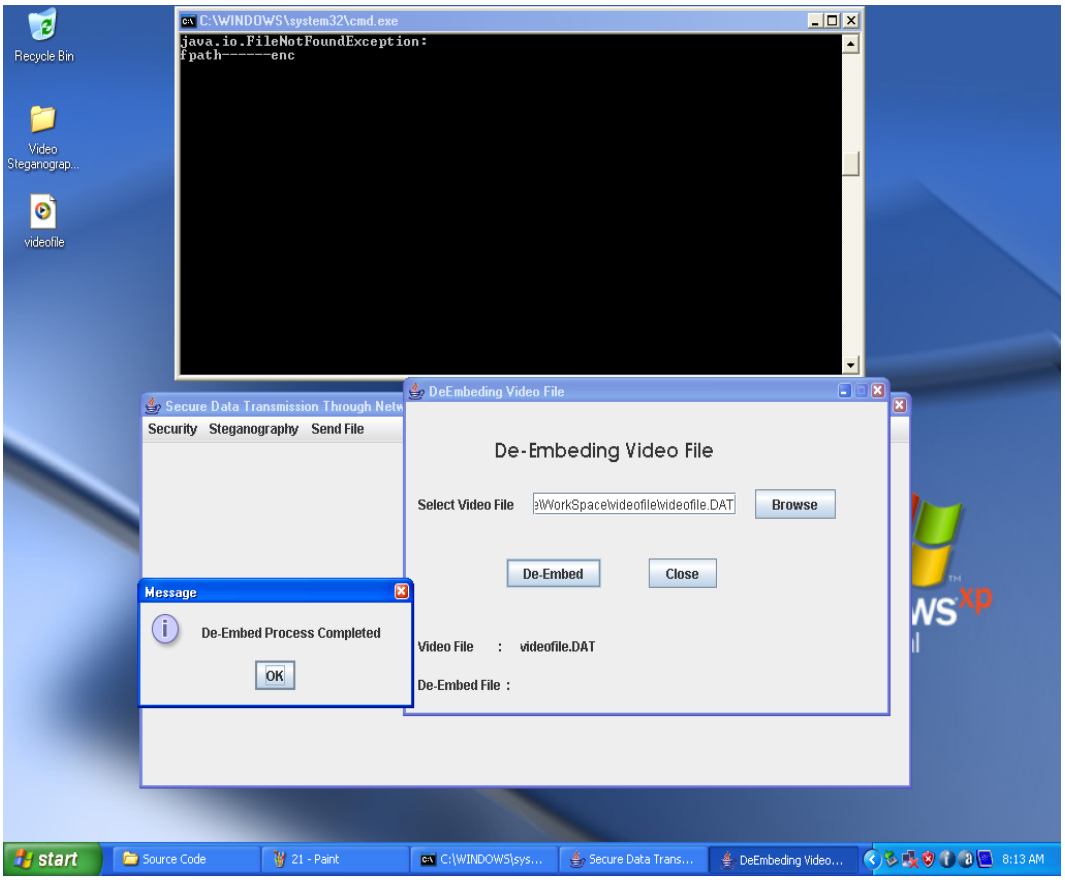
Sending Process across Network



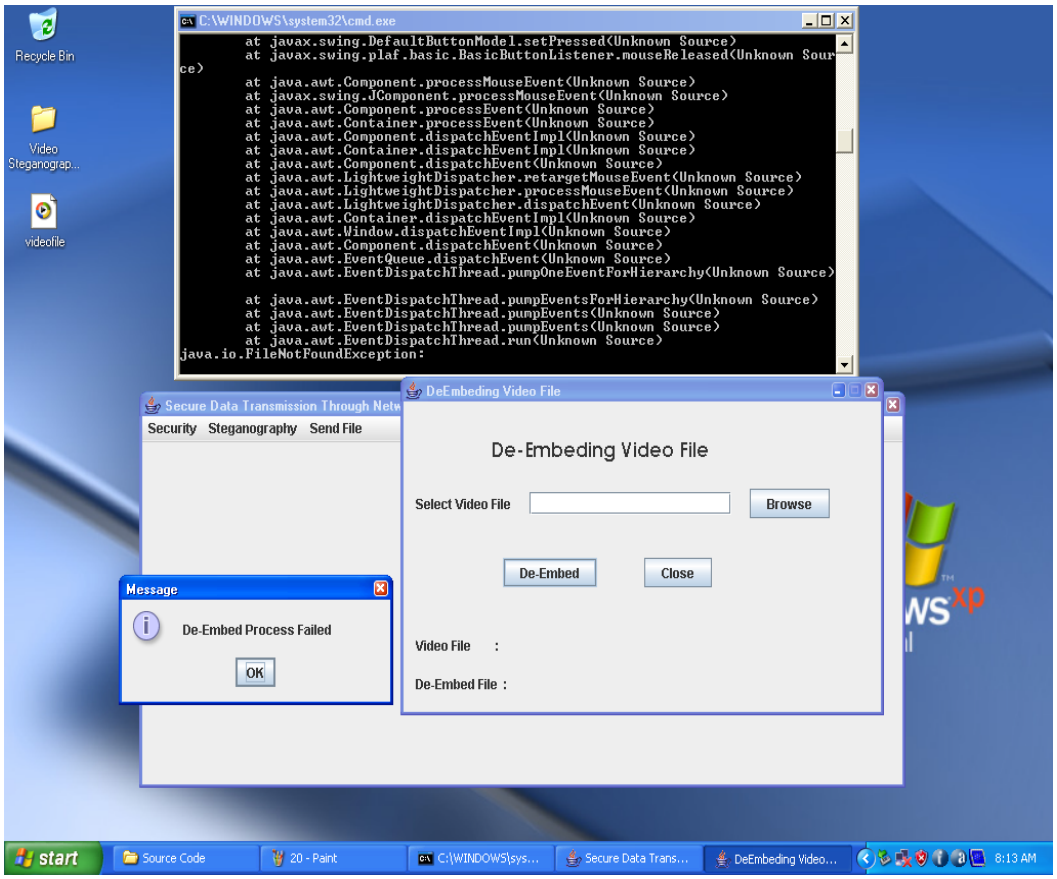
File Successfully Sent



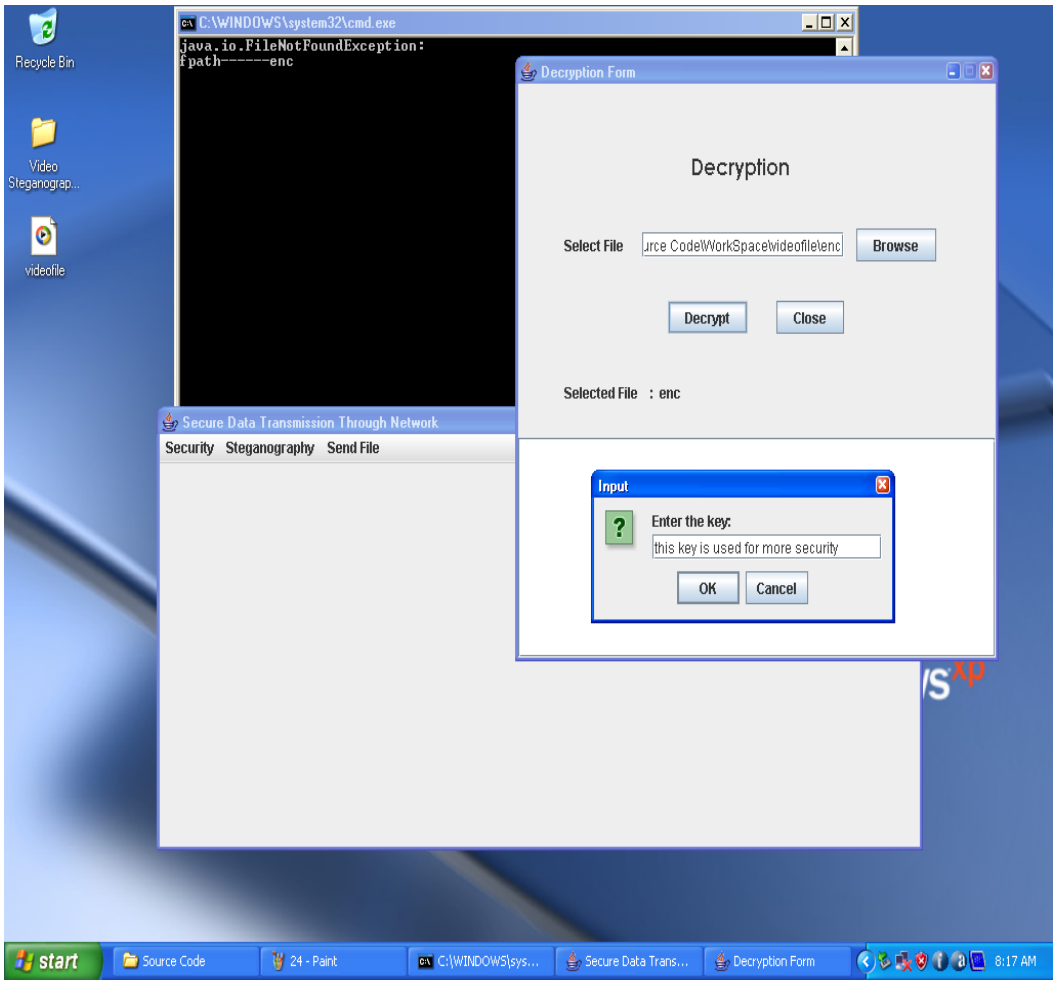
File Successfully De-embedded



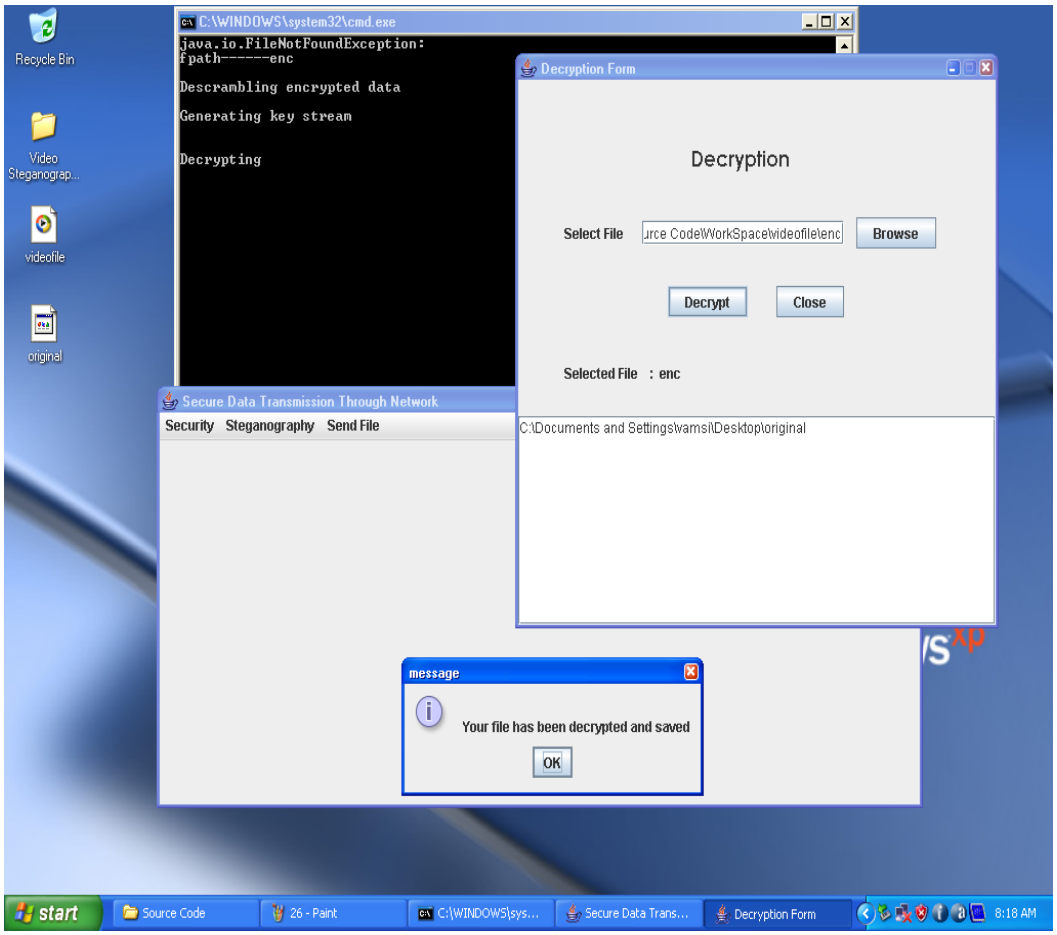
If we Enter Null values



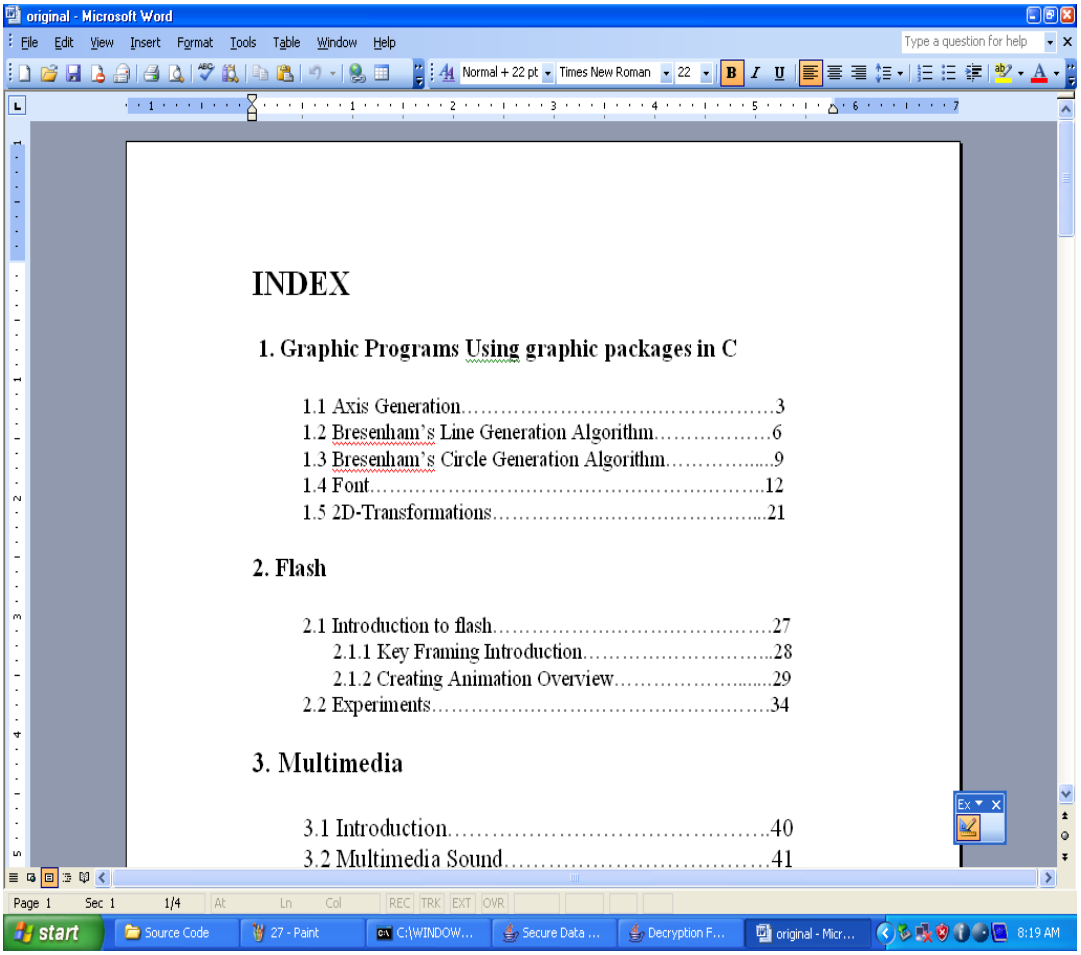
Entering the Appropriate key



File Successfully Decrypted



Original Data File



10. CONCLUSION

The developed software is tested with sample data and outputs obtained according to the requirements. The performance of the system is evaluated and is found to be much more efficient than the existing system. It will meet the primary requirements of the concern. Even though we have tried our level best to make it a dream project, due to time constraints we could not add more facilities to it. So, the project has to be improved by having modifications as and when the necessity arises in dual course.

11. FUTURE ENHANCEMENT

Any specification untraced errors will be concentrated in the coming versions, which are planned to be developed in near future. The following have been identified as the future scope:

- Implementation of the cryptographic algorithm in hardware.
- Error correction using various error correction techniques or development of new techniques.
- Data compression using existing techniques or developing new techniques.

12. REFERENCES

[1] http://www.google.com

[2]http://www.wikipedia.com

[3] Biham, E., & Shamir, "A Differential cryptanalysis of the data encryption Standard". New York: Springer-Verlag,1992.

[4] Kelsey, J., Schneier, B., & Wagner, D. "Related key cryptanalysis of 3-WAY", Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In Information and Communications Security—Proceedings of ICICS, 1334, 1997.

[5] Mirza, F. "Block ciphers and cryptanalysis". England: Royal Holloway University of London, Department of Mathematics, 1998.

[6] Wheeler, D.J., & Needham, R.J. "TEA, a tiny encryption algorithm". In Fast Software Encryption – Proceedings of the 2nd International Workshop,1008, 1998.

[7] Andem, Vikram Reddy . "A Cryptanalysis of the Tiny Encryption Algorithm", 2003

[8] Hernández, Julio César; Isasi, Pedro; Ribagorda, Arturo. "An application of genetic algorithms to the cryptoanalysis of one round TEA". Proceedings of the 2002 Symposium on Artificial Intelligence and its Application, 2002.

[9] Hong, Seokhie; Hong, Deukjo; Ko, Youngdai; Chang, Donghoon; Lee, Wonil; Lee, Sangjin. "Differential cryptanalysis of TEA and XTEA". In Proceedings of ICISC, 2003.

[10] Christian Cachin." An information-theoretic model for steganography". Lecture Notes in Computer Science, 1525:306.318, 1998.

[11] Luis von Ahn and Nicholas J. Hopper. "Public-key steganography" In Lecture Notes in Computer Science, volume 3027,1995.