P-3311

AUDIO STEGANOGRAPHY

A PROJECT REPORT

Submitted by

G.V.GURU PRASAD
A.VARUNVENKATESSH
G.VIJAY

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY:: CHENNAI 600 025

APRIL 2010

ANNA UNIVERSITY:: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report entitled "**AUDIO STEGANOGRAPHY**" is the bonafide work of

G.V.GURU PRASAD	**71206205016**
A.VARUNVENKATESSH	**71206205055**
G.VIJAY	**71206205306**

who carried out the project work under my supervision.

[illegible]

SIGNATURE
Ms.S.Sathyavathi, M.E
Lecturer,
Department of Information Technology,
Kumaraguru College of Technology,
Chinnavedampatti Post,
Coimbatore-641006.

[illegible]

SIGNATURE
Dr.L.S.Jayashree,Ph.D.
HEAD OF DEPARTMENT,
Department of Information Technology,
Kumaraguru College of Technology,
Chinnavedampatti Post,
Coimbatore-641006.

Submitted for viva-voice examination held on 19.4.10.

[illegible] 19/4/20

INTERNAL EXAMINER

[illegible] 19.4.20

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

The' exhilaration achieved on successful completion of any task should be shared with the people behind the venture. At the onset, we thank the management of our college for having provided the excellent facilities to carry out the project.

We express our deep gratitude to our Principal **Dr.S.Ramachandran** for ushering us in the path of triumph.

We are always thankful to our beloved Dean and HEAD of Computer Science and Engineering Department, **Dr.S.Thangasamy**, whose consistent support and enthusiastic involvement helped us a great deal.

We convey our sincere thanks to our project coordinator **Dr.L.S.Jayashree,** for her invaluable assistance.

We are greatly indebted to our beloved guide **Ms.S.Sathyavathi,** Lecturer, Department of Information Technology for his excellent guidance and timely support during the course of this project.

We also feel elated in manifesting our deep sense of gratitude to all the staff and lab technicians in the Department of Information Technology

ABSTRACT

ABSTRACT

This,project entitled "**Audio Steganography**" developed to give protection to the secret information. Steganography is the process of hiding a secret message inside another non-secret message.The main objective of this project is to keep fame among its competitors. The input screens are designed in such a way that it is simple and user-friendly. There are many techniques of doing this the most commonly used are hiding the information in Text file, Image file, Audio file and Video file .While using text file eavesdroppers can easily identify there is a secret message. Image file overcome this by hiding secret message to an image file. It is an attempt to prevent eavesdroppers from knowing the secret message has been at all. In this there is a restriction for an amount of information to be hiding. This project "Audio Steganography" (LSB Coding technique) overcomes this by hiding information in Audio file. Since audio files are too long, it is possible to hide very large amount of information compared to Image. During encryption the audio file and the (secret message) text file are converted to give the destination audio file that hiding the appropriate secret message.During decryption the audio file that hide the secret message are converted to give the audio file and the secret text.

TABLE OF CONTENTS

LIST OF SYMBOLS

SYMBOLS	ABBREVIATION
1. EAS	Enhanced Audio Steganography
2. LSB	Least Significant Bit
3. GUI	Graphical User Interface
4 .JVM	Java Virtual Machine
5. API	Application Programming Interface
6.RMI	Remote Method Invocation
7.JFC	Java Foundation Classes
8.AWT	Abstract Window Toolkit
9.JDK	Java Development Kit
10.URL	Universal Resource Locator

1. INTRODUCTION

1.1 INTRODUCTION

In the current internet community, secure data transfer is limited due to its attack made on data communication. So more robust methods are chosen so that they ensure secured data transfer. One of the solutions which came to the rescue is the audio Steganography. But existing audio Steganography systems have poor interface, very low level implementation, difficult to understand and valid only for certain audio formats with restricted message size. Enhanced Audio Steganography (EAS) is one proposed system which is based on audio Steganography and cryptography, ensures secure data transfer from source to destination. EAS uses most powerful encryption algorithm in the first level of security, which is very complex to break. In the second level it uses a more powerful modified LSB (Least Significant Bit) Algorithm to encode the message into audio. It performs bit level manipulation to encode the message.

The basic idea behind this paper is to provide a good, efficient method for hiding the data from hackers and sent to the destination in a safer manner. Though it is well modulated software it has been limited to certain restrictions. The quality of sound depends on the size of the audio which the user selects and length of the message. Though it shows bit level deviations in the frequency chart, as a whole the change in the audio cannot be determined.

Steganography, coming from the Greek words stegos, meaning roof or covered and graphics which means writing, is the art and science of hiding the fact that communication is taking place. Using steganography, you can embed a secret message inside a piece of unsuspicious information and send it without anyone knowing of the existence of the secret message. Steganography and cryptography are closely related. Cryptography scrambles messages so they cannot be understood. Steganography on the other hand, will hide the message so there is no knowledge of the existence of the message in the first place. In some situations, sending an encrypted message will arise suspicion while an "invisible" message will not do so. Both sciences can be combined to produce better protection of the message. In this case, when the steganography fails and the message can be detected, it is still of no use as it is encrypted using cryptography techniques. Therefore, the principle defined once by Kerckhoffs for cryptography, also stands for steganography: the quality of a cryptographic system should only depend on a small part of information, namely the secret key. The same is valid for good steganographic systems: knowledge of the system that is used, should not give any information about the existence of hidden messages. Finding a message should only be possible with knowledge of the key that is required to uncover it.

1.2 ABOUT STEGANOGRAPHY

STEGANOGRAPHY is the art and science of hiding the existence of information. The term originated from Greek roots that literally mean "covered writing". Unlike cryptography, which simply conceals the content or meaning of a message, steganography conceals the very existence of a message. Digital steganography is the art of inconspicuously hiding data within data. Steganography is a process that involves hiding a message in an appropriate carrier for example, an audio or an image file. The carrier can then be sent to a receiver without anyone else knowing that it contains a hidden message.

Steganography and data hiding are not new concepts. It is believed that steganography was first practiced during the Golden Age in Greece. An ancient Greek record describes the practice of melting wax off wax tablets used for writing messages and then inscribing a message in the underlying wood. The wax was then reapplied to wood giving the appearance of a new, unused tablet. Resulting tablets could be innocently transported without anyone suspecting the presence of a message beneath the wax.

The important concept from this history lesson is that communication does not have to occur over standard open channels using well-known methods. The Internet, in its massive, protocol-laden glory, is a playground for the modern steganographer. For example, think of an IP packet as the wax tablet previously mentioned. The packet's data field is equivalent to the writing in the wax. The headers serve as the wood in this analogy.

1.2 OBJECTIVE OF THE PROJECT

The main motto of this project is to hide a text inside an audio file of any formats. During encryption the audio file and the (secret message) text file are converted to give the destination audio file that hides the appropriate secret message. During decryption the audio file that hide the secret message are converted to give the audio file and the secret text file. The size of the secret message can even exceed the size of the audio file

1.3 SCOPE OF THE PROJECT

Secrets can be hidden inside all sorts of cover information: text, images, audio, video and more. Most steganographic utilities nowadays, hide information inside images, as this is relatively easy to implement. However, there are tools available to store secrets inside almost any type of cover source. It is also possible to hide information inside texts, sounds and video films for example. The most important property of a cover source is the amount of data that can be stored inside it, without changing the noticeable properties of the cover. When an image is distorted or a piece of music sounds different than the original, the cover source will be suspicious and may be checked more thoroughly.

2. SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The existing system of Audio Steganography poses more restrictions on the choosing of audio files. User can select only wav files to encode. It supports water marking method to encode .It complexity arises when more message to be encoded. The message length is restricted to 500 characters. It doesn't shows the variations occurred after encoding the message. The LSB algorithm in the existing system is not efficient because it hides the message in consecutive bytes received from audio files. The disadvantages of existing system is

- Selection of audio formats is restricted to one.
- Non-Provision of encryption key
- Length of the message is limited to 500.
- Absence of frequency chart to show the variations.
- Lack in good user interface
- Consume much time to encode and decode.
- Non-Provision of sending the file to the destination.
- User needs to understand better to know the operations.

These are the disadvantages in the existing system which can be overcome by the proposed system.

2.2 PROPOSED SYSTEM

Basic idea of EAS

The basic idea behind this is to provide a good, efficient method for hiding the data from hackers and sent to the destination in safe manner. Though it is well modulated software it has been limited to certain restrictions. The quality of sound depends on the size of the audio which the user selects and length of the message.

The main two features of this system are :

1) Size of file is not changed after encoding.

2) Since it is bit level manipulation the sound Variations cannot be determined by any current software.

The proposed system over comes all the restrictions made on the existing systems. It provides good looking environment to user .It also provide the user to give secret key for encryption. The length of message is more than the existing system, and provides frequency chart to see the variations after encoding. The quality of the audio doesn't change variably. It cannot detect the lack in quality of sound. The encryption key can be any combination of characters, symbols and numbers. The key which is used for encoding is also used for decoding .This is a secret key where the both user have to agree upon a single common key.

Enhanced Audio Steganography is a method of hiding the message in the audio file of any formats. EAS provides an easy way of implementation of mechanisms when compared with audio Steganography. Apart from the encoding and decoding in Audio Steganography, EAS contain extra layers of encryption and decryption. The four layers in EAS are:

1. Encoding

The audio file contains set of bytes. For e.g. take an audio file which play for 10 secs. It has more than 60,000 bytes. Each byte is received and checked if the received byte is 254 or 255.If it is byte 255 or 254, encoding is done. So for one character to encode we need eight 254 or 255 bytes. One character is hidden in consecutive eight 254 or 255 bytes. In order to mark the end of message, the LSB bit of next eight consecutive 254 or 255 bytes which comes after all the messages have encoded are replaced by 1.Before encoding, message is encrypted using public key.

2. Decoding:

The encoded file is decoded to get the message .The message is decoded first and then decrypted by the public key. The eight consecutive 254 or 255 bytes are taken and decrypted with the public key. This decrypted byte have value less than 128.So if the value is 255 after decrypted then it is said to be end of message.

3. Encryption:

The user allowed entering the public key/shared key in any combination of numbers, symbols and characters. The key contains set of characters. All characters are converted to ASCII value and add all the ASCII value to get single number. And that single number is converted to bit pattern and by simple logical operation (XOR) you can get a single number less than 128.It is a new private key .It is added to the characters one by one in the message, before encoding.

4. **Decryption:**

The LSB bits of consecutive eight 254 or 255 bytes are taken and subtracted with the key to get the original character.

The advantages of proposed system:

- ❖ Different Audio formats are supported by the system.
- ❖ Provision of encryption key and performs simple encryption algorithm.
- ❖ The encryption key is modified by a strong algorithm to get a new key, which is used to encrypt the message. So even if the key is known for an intruder, he cannot break the code with that key.
- ❖ Presence of frequency chart to show the variations that helps the user to determine.
- ❖ Consumption of time to encode and decode is reduced.
- ❖ Provision of sending the file to the destination is given so that after encoding the user can send the file by giving destination IP address.

2.2.2 Powerful encryption algorithm

This algorithm is used to encrypt the message before encoding for further security purpose. The following steps is used to encrypt the message.

a. Adding all ASCII values of characters in the key given by user.

b. Converting the sum into bit pattern

c. Performing logical operation to the bit pattern.

d. Adding to the encoded character.

For more security enhancement the encoding is done only when the byte which is received from the audio is 254 or 255.This selection of particular bytes for encoding will reduce the lack in quality of audio after encoding. It can be proved by seeing the frequency chart indicating the deviations happened after encode. Thought it shows bit level deviations in the chart as a whole the change in the audio cannot be determined.

2.3 AUDIO BASED STEGANOGRAPHY METHODS

LSB (Least Significant Bit)

Algorithm is used to encode the message into audio. It performs bit level manipulation to encode the message. The following steps are:

a. Receives the audio file in the form of bytes and converted in to bit pattern.

b. Each character in the message is converted in bit pattern.

c. Replaces the LSB bit from audio with LSB bit from character in the message.

LSB Encoding:

Least significant bit (LSB) coding is the simplest way to embed information in a digital audio file. By substituting the least significant bit of each sampling point with a binary message, LSB coding allows for a large amount of data to be encoded. The following diagram illustrates how the message 'KEY' is encoded in a 16-bit CD quality sample using the LSB method. In LSB coding, the ideal data transmission rate is 1 kbps per 1 kHz. In some implementations of LSB coding, however, the two least significant bits of a sample are replaced with two message bits. This increases the amount of data that can be encoded but also increases the amount of resulting noise in the audio file as well.

Thus, one should consider the signal content before deciding on the LSB operation to use. For example, a sound file that was recorded in a bustling subway station would mask low-bit encoding noise. On the other hand, the same noise would be audible in a sound file containing a piano solo.

To extract a secret message from an LSB encoded sound file, the receiver needs access to the sequence of sample indices used in the embedding process. Normally, the length of the secret message to be encoded is smaller than the total number of samples in a sound file. One must decide then on how to choose the subset of samples that will contain the secret message and communicate that decision to the receiver. One trivial technique is to start at the beginning of the sound file and perform LSB coding until the message has been completely embedded, leaving the remaining samples unchanged. This creates a security problem, however in that the first part of the sound file will have different statistical properties than the second part of the sound file that was not modified. One solution to this problem is to pad the secret message with random bits so that the length of the message is equal to the total number of samples. Yet now the embedding process ends up changing far more samples than the transmission of the secret required. This increases the probability that a would-be attacker will suspect secret communication. A more sophisticated approach is to use a pseudorandom number generator to spread the message over the sound file in a random manner. One popular approach is to use the random interval method, in which a secret key possessed by the sender is used as a seed in a pseudorandom number generator to create a random sequence of sample indices. The receiver also has access to the secret key and knowledge of the pseudorandom number generator, allowing the random sequence of sample indices to be reconstructed. Checks must be put in place, however, to prevent the pseudorandom number generator from generating the same sample index twice. If this happened, a collision would occur where a sample already modified with part of the message is modified again. The problem of collisions can be overcome by keeping track of all the and choose an appropriatemethod to unstego the message from the audio. The user should then be provided with the hidden message. A graphical user interface will be provided for the user to select the appropriate files and methods. The software provides a GUI, which will allow a user to select a file containing the message, the audio in which to store the message and a file in which to store the stegoed audio.

2.3.2 Detecting Hidden Code

Steganalysis, the official countermeasure to steganography, is the art of detecting and often decoding hidden data within a given medium. Two major tools in steganalysis, information theory and statistical analysis, reveal in clear terms the tremendous potential for hidden information in Internet data — as long as a set of data can be compressed to a smaller size, there is room for hidden data within the medium. Accordingly, the path of seeking hidden data is treacherous and uncertain. Unless hidden data is encoded in a common, well-defined format, it may be virtually impossible to detect in the carrier data. To a steganalysis expert unable to determine the chosen encoding, a bit is just a bit.

2.4 ADVANTAGES AND LIMITATIONS

The advantages of Steganography are:

1. It provides a better security for the sharing of data in local area network.
2. Important files carrying confidential information can be stored in the server in an encrypted form.
3. Using public key or private key can encrypt files.
4. No intruder can get any useful information from the original file during transmit.
5. It provides a better-secured data storage and transmission both at the system level and network level.

The limitations of Steganography are:

It provides the storing of data in an unprotected mode.

Password leakage may occur and it leads to the unauthorized access of data.

The intruders will affect stegos.

2.5 FEATURES OF AUDIO STEGANOGRAPHY

The main features of Audio Steganography are:

1. Messages do not attract attention to themselves.
2. The tool should be easy to use, and should use a graphical user interface.
3. The tool works cross-platform.
4. Small file distortions after hiding information.
5. It is possible to extract info after it has been compressed.
6. Information is protected with password
7. Unique.

3. ANALYSIS

3.ANALYSIS

Requirement Analysis is the first technical step in software engineering process. It is at this point that a general statement of software scope is refined into concrete specification that becomes the foundation for all software engineering activities that follow.

Analysis must focus on information, functional and behavioral domains of the problem. To better understand what is required, models are created and the problem is partitioned. In many cases it is not possible to completely specify a problem at an early stage. Prototyping offers an alternate approach from which requirements can be refined

3.1 Problem Specification

The basic aim of problem analysis is to obtain a clear understanding of the needs of the client and end users, what exactly is desired from software, and what are the constraints on the solutions. A good problem statement should be short and succinct - 1 or 2 statements is best. It is a good idea to define the problem and scope as early as possible, before getting deeper into analysis of the detailed requirements.

In today's society the most practical implementation of steganography is used in the world of computers. Data is the heart of computer communication and over the year a lot of methods have been created to accomplish the goal of using steganography to hide data. The trick is to embed the hidden object into a significantly larger object so the change is undetectable by the human ear. The best object up to this writing is probably a digital audio. It is important to understand the process by which digital steganography takes place, and to make sure your cover audio is large enough to support the byte manipulation.

The basics of embedding data rely on three different facts. These three items are capacity, security, and robustness. Capacity means the amount of data that can be hidden in the cover audio. Security is the interceptor's ability to decipher the data hidden inside the cover audio. Finally, robustness means the amount of manipulation a cover audio can handle before it is obvious a change has taken place. Steganography closely resembles encryption in the fact that it requires the receiver to know the secret which is called the secret key. In order for steganography to remain secure the initial unmodified cover audio must also be kept secret. It would be easy to detect hidden data in an audio if one was to have the original audio side by side with the steganography cover audio

3.2 Feasibility Study

The three important tests of feasibility are:

3.2.1 Technical Feasibility: - There are a number of technical issues, which are generally raised during the feasibility stage of investigation. They are as follows.

• Does the necessary technology exist to do what is suggested?

• Are there technical guarantees of accuracy, reliability, ease of access and data security?

3.2.2 Economic Feasibility: - A system that can be developed technically and that will be used if installed must still be profitable for the organization. Some of the questions for Economic feasibility are:

• Are there sufficient benefits when creating system to make the cost acceptable?

• Are the costs of not creating the system so great that the project must be undertaken?

• Is the cost of hardware and software for the class of application being considered is feasible?

3.2.3 Operational Feasibility:

Proposed projects are beneficial only if they can be turned into information

Systems that will meet the user requirements. Simply stated this test of feasibility asks if the system will work when it is developed and installed. Here are questions that will help test the operational feasibility of a project.

• Is there sufficient support for the project from the management? from the users?
• If the current system is well liked and used to extranet that persons will not be able to see reasons fro change there may be resistance.
• Are current business methods acceptable to the users? If they are not, users may welcome a change that will bring about a more operational and use full system.
• Have the users been involved in the planning and development of the project?
• Early involvement reduces the chances of resistance to the system and changes in general and increases the likelihood of successful projects.
• Will the proposed system cause harm? Will it produce poorer results in any respect or area? Will loss of control result in any area?
• Will customers be affected in an undesirable way? Will the system slow performance in?
Operational feasibility means users should support the project. During the preliminary investigation it is came to know that the proposed system is operationally feasible as many people are investigated in such a system as the benefits are high. As our system is user friendly there canbe no difficulty in operating our system.

3.3 SOFTWARE REQUIREMENTS SPECIFICATION

A software requirements specification is developed as a consequence of analysis. Review is essential to ensure that the developer and customers have the same perception.Software requirements specification (SRS) is the starting point of the software development activity. The Software Requirements Specification is produced at the culmination of the analysis task. The introduction of the software requirements specification states the goals and objectives of the software, describing it in the context of the computer-based system. The software requirements specification includes an information description, functional description, behavioral description, validation .

The purpose of this document is to present the software requirements in a precise and easily understood manner. This document provides the functional, performance, design and verification requirements of the software to be developed.

This is the only document that describes the requirements of the system. This is meant for use by the developers and will also be the basis for validating the final delivered system.

A requirement is a statement about what the proposed system will do that all stakeholders agree must be made true in order for the customer's problem to be adequately solved. Requirements can be divided into two major types, functional and non-functional. Requirements documents normally include both.

3.3.1 Functional Requirements

Functional Requirements describe what the system should do, i.e. the services provided for the users. The functional requirements of this system are: Inputs: The sender should provide a password, the message to embed, and a cover audio file to hold the message. The receiver should provide the same password used for embedding the message, and the carrier audio in which the message is embedded.

Outputs: The output at the sender end is the carrier audio file with the message embedded in it. The output for the receiver is the message contained in the carrier audio, and the secret message will be display in separate notepad.

3.3.2 Non-Functional Requirements

Non-functional requirements are constraints that must be adhered to during development. They limit what resources can be used and set bounds on aspects of the software's quality.

One of the most important things about non-functional requirements is to make them verifiable. The verification is normally done by measuring various aspects of the system and seeing if the measurements confirm to the requirements.

Non-functional requirements are divided into several groups:

The first group of categories reflects the five qualities attributes

1. Usability
2. Efficiency
3. Reliability
4. Maintainability
5. Reusability

These requirements constrain the design to meet specified levels of quality.

The second group of non-functional requirements categories constrains the environment and technology of the system like,

1. Platform
2. Technology to be used

3.3.3 Software Requirements

The minimum requirements to run the proposed system are:

Operating system : Any 32bit OS

Platform : Java, swings

Audio drives

Media player

Text editor

Sound converter

3.3.4 Hardware Requirements

The minimum requirements to run the proposed system are:

Processor : P IV

Hard Disk : 40GB

RAM : 256MB

Audio system : Speakers

3.4 Modules

The modules with in the hiding message using audio stegnography are

1. Embedded.
2. Extraction

Embedded:

In the embedded module we are giving the existed audio and text file. The embedded modules embed the text file into the audio file.

Extraction:

In the extraction module we are extract the message from the carrier audio file.

4. TESTING AND IMPLEMENTATION

4. TESTING AND IMPLEMENTATION

4.1 SYSTEM TESTING

4.1.1 Software Testing

As the coding is completed according to the requirement we have to test the quality of the software. Software testing is a critical element of software quality assurance and represent the ultimate review of specification, design and coding. Although testing is to uncover the errors in the software but it also demonstrates that software functions appear to be working as per the specifications, those performance requirements appear top have been met. In addition, data collected as testing is conducted provide a good indication of software and some indications of software quality as a whole. To assure the software quality we conduct both White Box Testing and Black Box Testing.

White Box Testing

White Box Testing is a test case design method that uses the control structure of the procedural design to derive test cases. As we are using a non-procedural language, there is very small scope for the White Box Testing. Whenever it is necessary , there the control structure are tested and successfully passed all the control structure with a very minimum error.

Black Box Testing

Black Box Testing focuses on the functional requirement of the software. It enables to derive sets of input conditions that will fully exercise all functional requirements for a program.

The Black Box Testing finds almost all errors. If finds some interface errors and errors in accessing the database and some performance errors. In Black

Box Testing we use mainly two techniques Equivalence partitioning the Boundary Volume Analysis Technique.

Equivalence Partitions

In the method we divide input domain of a program into classes of data from which test cases are derived. An Equivalence class represents a set of valid or invalid of a set of related values or a Boolean condition. The equivalence for these is

Input condition requires specific value-specific or non-specific two classes.

• Input condition requires a range or out of range two classes.

• Input condition specifies a number of a set-belongs to a set or not belongs to the set two classes.

• Input condition is Boolean-valid or invalid Boolean condition two classes.

By these types of equivalent classes, we can test for many cases.

Boundary Values Analysis:

Number of errors usually occurs at the boundaries of the input domain generally. In this technique a selection of test cases is exercised using boundary values i.e., around boundaries.

By the above two techniques, we eliminated almost all errors from the software and checked for numerous test values for each and every input value. The results were satisfactory.

4.1.2 Flow of Testing

System testing is designated to uncover weakness that was not detected in the earlier tests. The total system is tested for recovery and fallback after various major failures to ensure that no data are lost. An accepted test is done to validity and reliability of the system. The philosophy behind the testing is to find error in project. There are many test cases designed with this is mind. The flow of testing is as follows.

Code Testing

Specification testing is done to check if the program does with it should do and how it should behave under various conditions or combinations and submitted for processing in the system and it's checked if any overlaps occur during the processing. This strategy examines the logic of the program. Here only syntax of the code is tested. In code testing syntax errors are corrected, to ensure that the code is perfect.

Unit Testing

The first level of testing is called unit testing. Here different modules are tested against the specifications produced during the design of the modules. Unit testing is done to test the working of individual modules with test oracles. Unit testing comprises a set of tests preformed by an individual programmer prior to integration of the units into a large system. A program unit is small enough that the programmer who developed if can test it in a great detail. Unit testing focuses first on the modules to locate errors. These errors are verified and corrected so that the unit perfectly fits to the project.

System Testing

The next level of testing is system testing and acceptance testing. This testing is done to check if the system has its requirements and to find the external behavior of the system.

System testing involves two kinds of activities:

• Integration testing

• Acceptance testing

Integration Testing

The next level of testing is called the Integration Testing. In this many tested modules are combined into subsystems, which were tested. Test case data is prepared to check the control flow of all the modules and to exhaust all possible inputs to the program. Situations like treating the modules when there is no data entered in the text box is also tested.

This testing strategy dictates the order in which modules must be available, and exerts strong influence on the order in which the modules must be written, debugged and unit tested. In integration testing, all the modules / units on which unit testing is performed are integrated together and tested.

Acceptance Testing

This testing is performed finally by user to demonstrate that the implemented system satisfies its requirements. The user gives various inputs to get required outputs.

Specification Testing

Specification testing is done to check if the program does what is should do and how it should behave under various conditions or combination and submitted for processing in the system and it is checked if any overlaps occur during the processing.

Testing Objectives

The following are the testing objectives...

• Testing is a process of executing a program with the intent of finding an error.

• A good test case is one that has a high probability of finding an as yet undiscovered error.

• A successful test is one that uncovers an as yet undiscovered error.

The above objectives imply a dramatic change in view point. They move counter to the commonly held view that a successful test is one in which no errors are found. Our objective is to design tests that systematically verify different clauses of errors and do so with minimum amount of time and effort. If testing is conducted successfully, it will uncover errors in the software. As a secondary benefit, testing demonstrates that software functions appear to be working according to specification and that performance requirements appear to have been met. In addition, data collected as testing is conducted provides a good indication of software. Testing can't show the absence of defects, it can only show that software errors are present. It is important to keep this stated in mind as testing is being conducted.

Testing principles

Before applying methods to design effective test cases, a software engineer must understand the basic principles that guide software testing.

• All tests should be traceable to customer requirements.

• Tests should be planned long before testing begins.

• Testing should begin "in the small" and progress towards testing "in the large".

• Exhaustive testing is not possible.

5. LITERATURE OVERVIEW

5.LITERATURE OVERVIEW

5.1 JAVA

Java was developed at Sun Microsystems. Work on Java initially began with the goal of creating a platform-independent language and OS for consumer electronics. The original intent was to use C++, but as work progressed in this direction, developers identified that creating their own language would serve them better. The effort towards consumer electronics led the Java team, then known as First Person Inc., towards developing h/w and s/w for the delivery of video-on-demand with Time Warner.

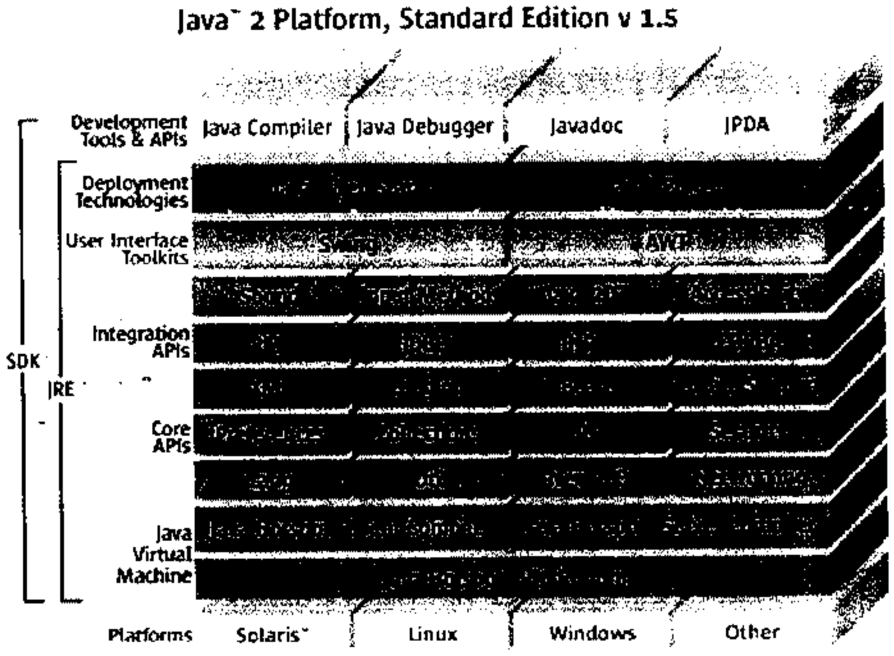


Fig 5.1 Java 2 Platform,Standard Edition v 1.5

Unfortunately (or fortunately for us) Time Warner selected Silicon Graphics as the vendor for video-on-demand project. This set back left the First Person team with an interesting piece of s/w (Java) and no market to place it.

Eventually, the natural synergies of the Java language and the www were noticed, and Java found a market.

Today Java is both a programming language and an environment for executing programs written in Java Language. Unlike traditional compilers, which convert source code into machine level instructions, the Java compiler translates java source code into instructions that are interpreted by the runtime Java Virtual Machine. So unlike languages like C and C++, on which Java is based, Java is an interpreted language.

Java is the first programming language designed from ground up with network programming in mind. The core API for Java includes classes and interfaces that provide uniform access to a diverse set of network protocols. As the Internet and network programming have evolved, Java has maintained its cadence. New APIs and toolkits have expanded the available options for the Java network programmer.

5.1.1 FEATURES OF JAVA

In one of their early papers about the language, Sun described Java as follows: Java: A simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, and dynamic language. Sun acknowledges that this is quite a string of buzzwords, but the fact is that, for the most part, they aptly describe the language. In order to understand why Java is so interesting, let's take a look at the language features behind the buzzwords.

Object-Oriented

Java is an object-oriented programming language. As a programmer, this means that you focus on the data in your application and methods that manipulate that data, rather than thinking strictly in terms of procedures. If you're accustomed to procedure-based programming in C, you may find that you need to change how you design your programs when you use Java. Once you see how powerful this new paradigm is, however, you'll quickly adjust to it. In an object-oriented system, a ***class*** is a collection of data and methods that operate on that data. Taken together, the data and methods describe the state and behavior of an object. Classes are arranged in a hierarchy, so that a subclass can inherit behavior from its super class. A class hierarchy always has a root class; this is a class with very general behavior. Java comes with an extensive set of classes, arranged in packages that you can use in your programs. For example, Java provides classes that create graphical user interface components (the java.awt package), classes that handle input and output (the java.io package), and classes that support networking functionality (the java.net package). The Object class (in the java.lang package) serves as the root of the Java class hierarchy. Unlike C++, Java was designed to be object-oriented from the ground up. Most things in Java are objects; the primitive numeric, character, and Boolean types are the only exceptions. Strings are represented by objects in Java, as are other important language constructs like threads. A class is the basic unit of compilation and of execution in Java; all Java programs are classes. While Java is designed to look like C++, you'll find that Java removes many of the complexities of that language. If you are a C++ programmer, you'll want to study the object-oriented constructs in Java carefully. Although the syntax is often similar to C++, the behavior is not nearly so analogous. For a complete description of the object-oriented features of Java, The object oriented language used to create executable contents such as applications and applets.

Interpreted

Java is an interpreted language: the Java compiler generates byte-codes for the Java Virtual Machine (JVM), rather than native machine code. To actually run a Java program, you use the Java interpreter to execute the compiled byte-codes. Because Java byte-codes are platform-independent, Java programs can run on any platform that the JVM (the interpreter and run-time

system) has been ported to. In an interpreted environment, the standard "link" phase of program development pretty much vanishes. If Java has a link phase at all, it is only the process of loading new classes into the environment, which is an incremental, lightweight process that occurs at run-time. This is in contrast with the slower and more cumbersome compile-link-run cycle of languages
like C and C++.

Dynamic and Distributed

Java is a dynamic language. Any Java class can be loaded into a running Java interpreter at any time. These dynamically loaded classes can then be dynamically instantiated. Native code libraries can also be dynamically loaded. Classes in Java are represented by the Class class; you can dynamically obtain information about a class at run-time. This is especially true in Java 1.1, with the addition of the Reflection API. Java is also called a distributed language. This means, simply, that it provides a lot of high-level support for networking. For example, the URL class and Oar elated classes in the java.net package make it almost as easy to read a remote file or resource as it is to read a local file. Similarly, in Java 1.1, the Remote Method Invocation (RMI) API allows a Java program to invoke methods of remote Java objects, as if they were local objects. (Java also provides traditional lower-level networking support, including datagram's and stream-based connections through sockets.) The distributed nature of Java really shines when combined with its dynamic class loading capabilities. Together, these features make it possible for a Java interpreter to download and run code from across the Internet. (As we'll see below, Java implements strong security measures to be sure that this can be done safely.) This is what happens when a Web browser downloads and runs a Java applet, for example. Scenarios can be more complicated than this, however. Imagine a multi-media word processor written in Java. When this program is asked to display some type of data that it has never encountered before, it might dynamically download a class from the network that can parse the data, and then dynamically download another class (probably a Java "bean") that can display the data within a compound document. A program like this uses distributed resources on the network to dynamically grow and adapt to the needs of its user.

Robust

Java has been designed for writing highly reliable or ***robust*** software. Java certainly doesn't eliminate the need for software quality assurance; it's still quite possible to write buggy software in Java. However, Java does eliminate certain types of programming errors, which makes it considerably easier to write reliable software. Java is a strongly typed language, which allows for extensive compile-time checking for potential type-mismatch problems. Java is more strongly typed than C++, which inherits a number of compile-time laxities from C, especially in the area of function declarations. Java requires explicit method declarations; it does not support C-style implicit declarations. These stringent requirements ensure that the compiler can catch method invocation errors, which leads to more reliable programs. One of the things that makes Java simple is its lack of pointers and pointer arithmetic. This feature also increases the robustness of Java programs by abolishing an entire class of pointer-related bugs. Similarly, all accesses to arrays and strings are checked at run-time to ensure that they are in bounds, eliminating the possibility of overwriting memory and corrupting data. Casts of objects from one type to another are also checked at run-time to ensure that they are legal. Finally, and very importantly, Java's automatic garbage collection prevents memory leaks and other pernicious bugs related to memory allocation and deallocation. Exception handling is another feature in Java that makes for more robust programs. An exception is a signal that some sort of exceptional condition, such as a "file not found" error, has occurred. Using the try/catch/finally statement, you can group all of your error handling code in one place, which greatly simplifies the task of error handling and recovery.

High-Performance

Java is an interpreted language, so it is never going to be as fast as a compiled language like C. Java 1.0 was said to be about 20 times slower than C. Java 1.1 is nearly twice as fast as Java 1.0, however, so it might be reasonable to say that compiled C code runs ten times as fast as interpreted Java byte-codes. But before you throw up your arms in disgust, be aware that this speed is more than adequate to run interactive, GUI and network-based applications, where the application is often idle, waiting for the user to do something, or waiting for data from the network. Furthermore, the speed-critical sections of the Java run-time environment, that do things like string concatenation and comparison, are implemented with efficient native code. As a further performance boost, many Java interpreters now include "just in time" compilers that can translate Java byte-codes into machine code for a particular CPU at run-time. The Java byte-code format was designed with these "just in time" compilers in mind, so the process of generating machine code is fairly efficient and it produces reasonably good code. In fact, Sun claims that the performance of byte-codes converted to machine code is nearly as good as native C or C++. If you are willing to sacrifice code portability to gain speed, you can also write portions of your program in C or C++ and use Java native methods to interface with this native code. When you are considering performance, it's important to remember where Java falls in the spectrum of available programming languages. At one end of the spectrum, there are high-level, fully-interpreted scripting languages such as Tcl and the UNIX shells. These languages are great for prototyping and they are highly portable, but they are also very slow. At the other end of the spectrum, you have low-level compiled languages like C and C++. These languages offer high performance, but they suffer in terms of reliability and portability. Java falls in the middle of the spectrum. The performance of Java's interpreted byte-codes is much better than the high-level scripting languages (even Perl), but it still offers the simplicity and portability of those languages.

Multithreaded

In a GUI-based network application such as a Web browser, it's easy to imagine multiple things going on at the same time. A user could be listening to an audio clip while she is scrolling a page, and in the background the browser is downloading an image. Java is a multithreaded language; it provides support for multiple threads of execution (sometimes called lightweight processes) that can handle different tasks. An important benefit of multithreading is that it improves the interactive performance of graphical applications for the user. If you have tried working with threads in C or C++, you know that it can be quite difficult. Java makes programming with threads much easier, by providing built-in language support for threads. The java.lang package provides a Thread class that supports methods to start and stop threads and set thread priorities, among other things. The Java language syntax also supports threads directly with the synchronized keyword. This keyword makes it extremely easy to mark sections of code or entire methods that should only be run by a single thread at a time. While threads are "wizard-level" stuff in C and C++, their use is commonplace in Java. Because Java makes threads so easy to use, the Java class libraries require their use in a number of places. For example, any applet that performs animation does so with a thread. Similarly, Java does not support asynchronous, non-blocking I/O with notification through signals or interrupts--you must instead create a thread that blocks on every I/O channel you are interested in.

5.2 JAVA PACKAGES USED IN THE PROJECT >Swings (JFC)

A Java toolkit for developing graphical user interfaces (GUIs). It includes elements such as menus, toolbars and dialog boxes. Swing is written in Java and is thus platform independent, unlike the Java Abstract Window Toolkit (AWT), which provides platform-specific code. Swing also has more sophisticated interface capabilities than AWT and offers such features as tabbed panes and the ability to change images on buttons. Swing is included in the Java Foundation Classes (JFC) which is provided in the Java Developers

Toolkit (JDK).

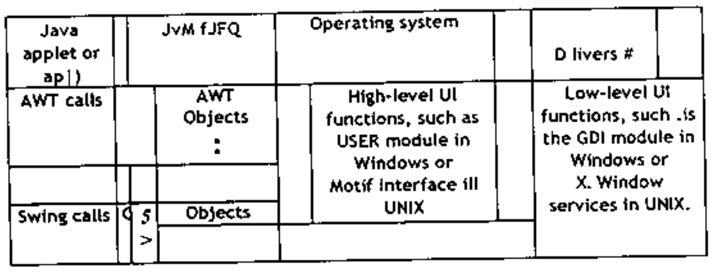

Java applet or ap|)
JvM fJFQ
Operating system
D livers #
AWT calls
AWT Objects
High-level UI functions, such as USER module in Windows or Motif interface ill UNIX
Low-level UI functions, such .is the GDI module in Windows or X. Window services in UNIX.
Swing calls
Objects

Fig : 5.2.1 Swings

The Java Foundation Classes (JFC) is a graphical framework for building portable Java-based graphical user interfaces (GUIs). JFC consists of the Abstract Window Toolkit (AWT), Swing and Java 2D. Together, they provide a consistent user interface for Java programs, regardless whether the underlying user interface system is Windows, Mac OS X or Linux.

AWT is the older of the two interface libraries, and was heavily criticized for being little more than a wrapper around the native graphical capabilities of the host platform. That meant that the standard widgets in the AWT relied on those capabilities of the native widgets, requiring the developer to also be aware of the differences between host platforms.

An alternative graphics library called the Internet Foundation Classes was developed in more platform-independent code by Netscape. Ultimately, Sun merged the IFC with other technologies under the name "Swing", adding the capability for a pluggable look and feel of the widgets. This allows Swing programs to maintain a platform-independent code base, but mimic the look of a native application.

Swing contains components that you'll use to build a GUI. I am listing you some of the commonly used Swing components. To learn and understand these swing programs, AWT Programming knowledge is not required

- JPanel is Swing's version of the AWT class Panel and uses the same default layout, FlowLayout. JPanel is descended directly from JComponent.

• JFrame is Swing's version of Frame and is descended directly from that class. The components added to the frame are referred to as its contents; these are managed by the contentPane. To add a component to a JFrame, we must use its contentPane instead.

• JInternalFrame is confined to a visible area of a container it is placed in. It can be iconified , maximized and layered.

• JWindow is Swing's version of Window and is descended directly from that class. Like Window, it uses BorderLayout by default.

• JDialog is Swing's version of Dialog and is descended directly from that class. Like Dialog, it uses Border Layout by default. Like JFrame and JWindow, JDialog contains a root Pane hierarchy including a contentPane, and it allows layered and glass panes. All dialogs are modal, which means the current thread is blocked until user interaction with it has been completed. JDialog class is intended as the basis for creating custom dialogs; however, some of the most common dialogs are provided through static methods in the class JOptionPane. JLabel, descended from JComponent, is used to create text labels.

• The abstract class AbstractButton extends class JComponent and provides a foundation for a family of button classes, including JButton.

• JTextField allows editing of a single line of text. New features include the ability to justify the text left, right, or center, and to set the text's font.
• JPasswordField (a direct subclass of JTextField) you can suppress the display of input. Each character entered can be replaced by an echo character.

This allows confidential input for passwords, for example. By default, the echocharacter is the asterisk, *.

• JTextArea allows editing of multiple lines of text. JTextArea can be used in conjunction with class JScrollPane to achieve scrolling. The underlying JScrollPane can be forced to always or never have either the vertical or horizontal scrollbar;

JButton is a component the user clicks to trigger a specific action.

• JRadioButton is similar to JCheckbox, except for the default icon for each class. A set of radio buttons can be associated as a group in which only one button at a time can be selected.

• JCheckBox is not a member of a checkbox group. A checkbox can be selected and deselected, and it also displays its current state.

• JComboBox is like a drop down box. You can click a drop-down arrow and select an option from a list. For example, when the component has focus, pressing a key that corresponds to the first character in some entry's name selects that entry. A vertical scrollbar is used for longer lists.

• JList provides a scrollable set of items from which one or more may be selected. JList can be populated from an Array or Vector. JList does not support scrolling directly, instead, the list must be associated with a scrollpane. The view port used by the scroll pane can also have a user-defined border. JList actions are handled using ListSelectionListener.

• JTabbedPane contains a tab that can have a tool tip and a mnemonic, and it can display both text and an image.

• JToolbar contains a number of components whose type is usually some kind of button which can also include separators to group related components within the toolbar.

• FlowLayout when used arranges swing components from left to right until there's no more space available. Then it begins a new row below it and moves from left to right again. Each component in a FlowLayout gets as much space as it needs and no more.

• BorderLayout places swing components in the North, South, East, West and center of a container. You can add horizontal and vertical gaps between the areas.

• GridLayout is a layout manager that lays out a container's components in a rectangular grid. The container is divided into equal-sized rectangles, and one component is placed in each rectangle.

• GridBagLayout is a layout manager that lays out a container's components in a grid of cells with each component occupying one or more cells, called its display area. The display area aligns components vertically and horizontally, without requiring that the components be of the same size.

• JMenubar can contain several JMenu's. Each of the JMenu's can contain a series of JMenuItem 's that you can select. Swing provides support for pull-down and popup menus.

• Scrollable JPopupMenu is a scrollable popup menu that can be used whenever we have so many items in a popup menu that exceeds the screen visible height.

5.3 IO Streams

Most programs use data in one form or another, whether it is as input, output, or both. The sources of input and output can vary between a local file, a socket on the network, a database, variables in memory, or another program. Even the type of data can vary between objects, characters, multimedia, and others.

Data for a program may come from several sources. Data created by a program may be sent to several destinations. The connection between a program and a data source or destination is called a stream. An input stream handles data flowing into a program. An output stream handles data flowing out of a program.

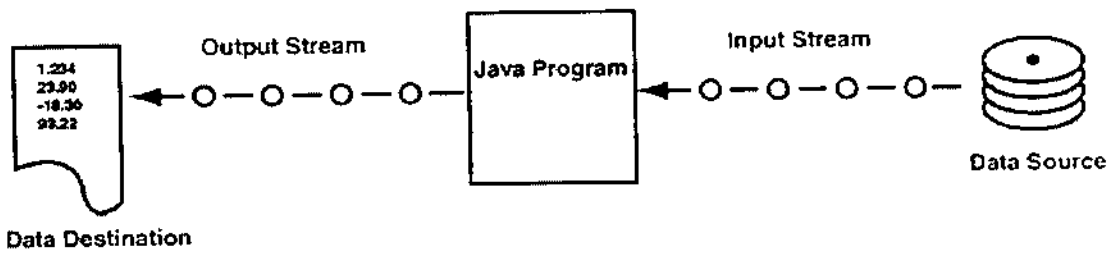


Fig 5.3.1 Data handling

The Java Development Kit (JDK) provides APIs for reading and writing streams of data. These APIs have been part of the core JDK since version 1.0, but are often overshadowed by the more well-known APIs, such as JavaBeans, JFC, RMI, JDBC, and so on. However, input and output streams are the backbone of the JDK APIs.

To bring data into a program, a Java program opens a stream to a data source, such as a file or remote socket, and reads the information serially. On the flip side, a program can open a stream to a data source and write to it in a serial fashion. Whether you are reading from a file or from a socket, the concept of serially reading from, and writing to different data sources is the same.

Prior to JDK 1.1, the input and output classes (mostly found in the java.io package) only supported 8-bit byte streams. The concept of 16-bit Unicode character streams was introduced in JDK 1.1. While byte streams were supported via the java.io.InputStream and java.io.OutputStream classes and their subclasses, character streams are implemented by the java.io.Reader and java.io.Writer classes and their subclasses.

Most of the functionality available for byte streams is also provided for character streams. The methods for character streams generally accept parameters of data type char parameters, while byte streams, you guessed it, work with byte data types. The names of the methods in both sets of classes are almost identical except for the suffix, that is, character-stream classes end with the suffix Reader or Writer and byte-stream classes end with the suffix Input Stream and Output Stream. For example, to read files using character streams, you would use the java.io.FileReader class; for reading it using byte streams you would use java.io.FileInputStream.

Unless you are working with binary data, such as image and sound files, you should use readers and writers.

Multithreading

Multitasking is performing two or more tasks at the same time. Nearly all operating systems are capable of multitasking by using one of two multitasking techniques: process-based multitasking and thread-based multitasking.

Process-based multitasking is running two programs concurrently. Programmers refer to a program as a process. Therefore, you could say that process-based multitasking is program-based multitasking.

Thread-based multitasking is having a program perform two tasks at the same time. For example, a word processing program can check the spelling of words in a document while you write the document. This is thread-based multitasking.

A good way to remember the difference between process-based multitasking and thread-based multitasking is to think of process-based as working with multiple programs and thread-based as working with parts of one program.

The objective of multitasking is to utilize the idle time of the CPU. Think of the CPU as the engine of your car. Your engine keeps running regardless of whether the car is moving. Your objective is to keep your car moving as much as possible so you can get the most miles from a gallon of gas. An idling engine wastes gas.

The same concept applies to the CPU in your computer. You want your CPU cycles to be processing instructions and data rather than waiting for something to process. A CPU cycle is somewhat similar to your engine running.

It may be hard to believe, but the CPU idles more than it processes in many desktop computers. Let's say that you are using a word processor to write a document. For the most part, the CPU is idle until you enter a character from the keyboard or move the mouse. Multitasking is designed to use the fraction of a second between strokes to process instructions from either another program or from a different part of the same program.

Making efficient use of the CPU may not be too critical for applications running on a desktop computer because most of us rarely need to run concurrent programs or run parts of the same program at the same time. However, programs that run in a networked environment, such as those that process transactions from many computers, need to make a CPU's idle time productive.

Networking

The java.net package provides a powerful and flexible infrastructure for networking. java.net provides the classes for implementing networking applications. Using the socket classes, you can communicate with any server on the Internet or implement your own Internet server. A number of classes are provided to make it convenient to use Universal Resource Locators (URLs) to retrieve data on the Internet.

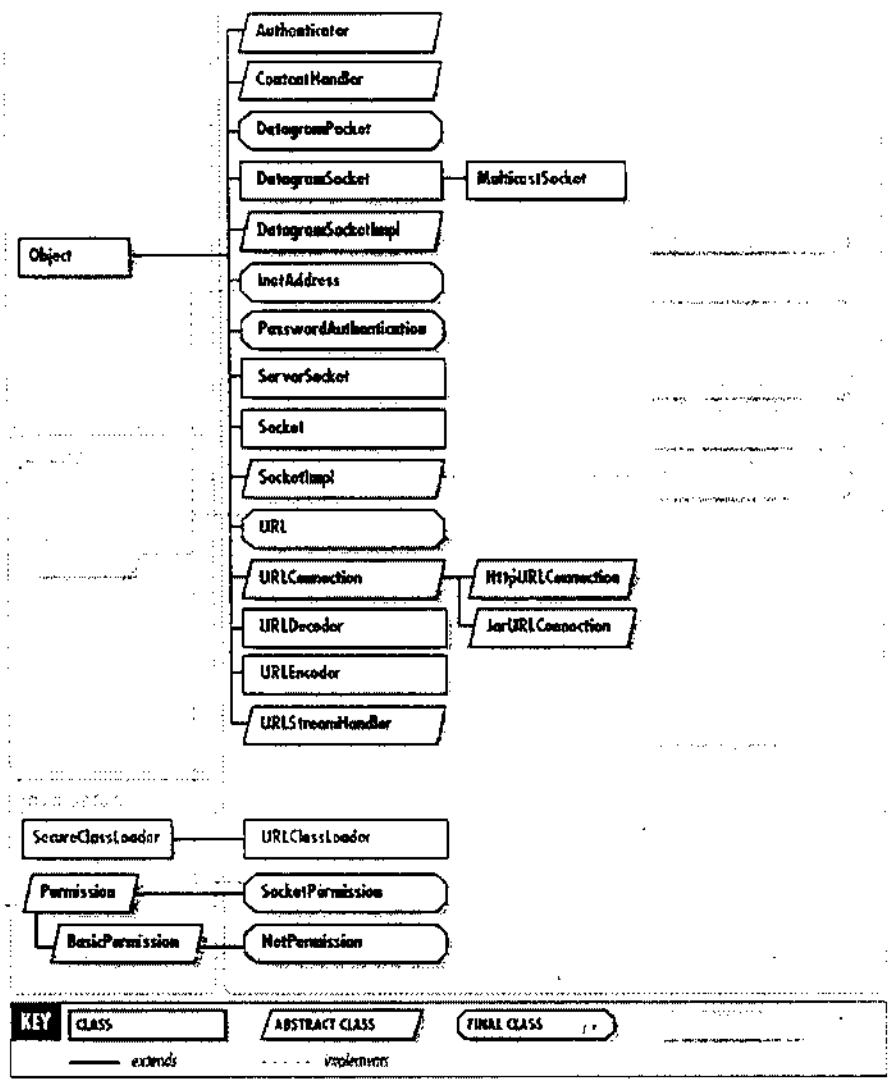


Fig : 5.2.2 The classes of java.netpackage

6. CONCLUSION

6. CONCLUSION

This proposed system is to provide a good, efficient method for hiding the data from hackers and sent to the destination in a safe manner. This proposed system will not change the size of the file even after encoding and also suitable for any type of audio file format. Encryption and Decryption techniques have been used to make the security system robust.

7.FUTURE SCOPE

7.FUTURE SCOPE

Though this method of audio steganography is most secure and efficient one, a minor limitation is preserving the file size when a text file of size greater than the audio file is embedded into it. In future this limitation will be concentrated and worked on about this issue to bring out a well efficient project.

APPENDIX 1

APPENDIX – 1
SOURCE CODE

StegoStick.java

```
import java.awt.BorderLayout;
import javax.swing.BorderFactory;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollBar;
import javax.swing.JScrollPane;
import javax.swing.JTabbedPane;
import javax.swing.JTextPane;
import javax.swing.SwingConstants;
import javax.swing.WindowConstants;
import javax.swing.border.BevelBorder;
import javax.swing.border.LineBorder;
import javax.swing.plaf.metal.*;
public class StegoStick extends javax.swing.JFrame {
{
try {
javax.swing.UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.
WindowsLookAndFeel");
} catch(Exception e) {
e.printStackTrace();
 }
 }
private JLabel titleLabel;
private JTextPane licensePane;
private JScrollPane readmeScrollPane;
private JScrollPane helpScrollPane;
private JTextPane helpPane;
private JScrollPane licenceScrollPane;
private JPanel licencePanel;
private JPanel helpPanel;
private JPanel unhidingPanel;
private JPanel hidingPanel;
private JPanel selectionPanel;
private JTextPane readmePane;
private JPanel readmePanel;
private JTabbedPane containerPane;
private void initGUI() {
try {
setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
getContentPane().setLayout(null);
{
titleLabel = new JLabel();
getContentPane().add(titleLabel);
titleLabel.setText("StegoStick");
titleLabel.setFont(new java.awt.Font("Baskerville Old Face",1,20));
titleLabel.setHorizontalAlignment(SwingConstants.CENTER);
titleLabel.setBounds(0, 0, 595, 49);
}
{
containerPane = new JTabbedPane();
getContentPane().add(containerPane);
containerPane.setBounds(0, 49, 595, 371);
containerPane.setTabPlacement(JTabbedPane.LEFT);
containerPane.setFont(new java.awt.Font("Corbel",0,16));
{
readmePanel = new JPanel();
containerPane.addTab("Readme", null, readmePanel, null);
readmePanel.setLayout(null);
readmePanel.setOpaque(false);
{
readmeScrollPane = new JScrollPane();
readmePanel.add(readmeScrollPane);
readmeScrollPane.setBounds(7, 7, 490, 350);
{
readmePane = new JTextPane();
readmeScrollPane.setViewportView(readmePane);
readmePane.setBounds(364, 147, 126, 196);
readmePane.setEditable(false);
readmePane.setText(readme);
}
}
}
{
```

```
hidingPanel = new HideGUI();
containerPane.addTab("Hiding", null, hidingPanel, null);
hidingPanel.setPreferredSize(new java.awt.Dimension(459,371));
                              hidingPanel.setOpaque(false);
}
{
unhidingPanel = new UnhideGUI();
containerPane.addTab("UnHiding", null, unhidingPanel, null);
unhidingPanel.setOpaque(false);
}
{
licencePanel = new JPanel();
containerPane.addTab("License", null, licencePanel, null);
licencePanel.setLayout(null);
licencePanel.setOpaque(false);
{
licenceScrollPane = new JScrollPane();
licencePanel.add(licenceScrollPane);
licenceScrollPane.setBounds(14, 21, 483, 336);
{
licensePane = new JTextPane();
licenceScrollPane.setViewportView(licensePane);
licenceScrollPane.setOpaque(false);
licensePane.setEditable(false);
licensePane.setText("The GNU General Public License (GPL)\nVersion 1,
March 2010\n\nCopyright (C) 2010 Kumaraguru College Of Technology\n");
}
}
}
}
pack();
this.setSize(610, 461);
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

RSAUnhidePanel.java

```
mport java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;

import javax.swing.BorderFactory;
import javax.swing.JButton;

import javax.swing.WindowConstants;
import javax.swing.border.BevelBorder;
import javax.swing.border.TitledBorder;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.JOptionPane;

public class RSAUnhidePanel extends UnhidePanel {

UnhideGUI parent;

public RSAUnhidePanel(UnhideGUI parent) {
super();
encryptTechnique = SelectionPanel.RSA;
this.parent = parent;
initGUI();
}

private void initGUI() {
try {
this.setPreferredSize(new java.awt.Dimension(504, 364));
this.setLayout(null);
this.setOpaque(false);
this.setForeground(new java.awt.Color(0,128,192));
{
coverFilePanel = new JPanel();
this.add(coverFilePanel);
```

```
coverFilePanel.setBounds(14, 28, 483, 77);
coverFilePanel.setOpaque(false);
coverFilePanel.setBorder(BorderFactory.createTitledBorder(null, "Cover
File", TitledBorder.LEADING, TitledBorder.TOP, new
java.awt.Font("Trebuchet MS",0,12), new java.awt.Color(0,0,255)));
coverFilePanel.setLayout(null);
coverFilePanel.setFont(new java.awt.Font("Trebuchet MS",0,12));
coverFilePanel.setForeground(new java.awt.Color(0,128,192));
{
coverField = new JTextField();
coverFilePanel.add(coverField);
coverField.setBounds(21, 28, 343, 28);
}
{
coverBrowseButton = new JButton();
coverFilePanel.add(coverBrowseButton);
coverBrowseButton.setText("browse");
coverBrowseButton.setBounds(385, 28, 84, 28);
coverBrowseButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {
System.out
.println("coverBrowseButton.actionPerformed, event="
+ evt);
JFileChooser fileChooser = new JFileChooser(currentPath);

//    ExampleFileFilter imgFilter = new ExampleFileFilter(new
String("bmp"),new String("Bit-Map Images"));
//   fileChooser.setFileFilter(imgFilter);

fileChooser.setDialogTitle(new String("Select Cover Image File"));
int result = fileChooser.showOpenDialog(RSAUnhidePanel.this);

if (result == JFileChooser.APPROVE_OPTION) {
File f = fileChooser.getSelectedFile();
currentPath = f.getPath();
coverField.setText(f.getAbsolutePath());
dstField.setText(f.getParent());
}
}
});
```

```
}
}
{
dstPanel = new JPanel();
this.add(dstPanel);
dstPanel.setBounds(14, 133, 483, 84);
dstPanel.setBorder(BorderFactory.createTitledBorder(null, "Destination Path",
TitledBorder.LEADING, TitledBorder.TOP, new java.awt.Font("Trebuchet
MS",0,12), new java.awt.Color(0,0,255)));
dstPanel.setLayout(null);
dstPanel.setOpaque(false);
dstPanel.setFont(new java.awt.Font("Trebuchet MS",0,12));
dstPanel.setForeground(new java.awt.Color(0,128,192));
{
dstField = new JTextField();
dstPanel.add(dstField);
dstField.setBounds(21, 35, 343, 28);
}
{
dstBrowseButton = new JButton();
dstPanel.add(dstBrowseButton);
dstBrowseButton.setText("browse");
dstBrowseButton.setBounds(385, 35, 84, 28);
dstBrowseButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {
System.out
.println("dstBrowseButton.actionPerformed, event="
+ evt);

JFileChooser fileChooser = new JFileChooser(currentPath);
fileChooser.setDialogTitle(new String("Select Destination Folder"));
fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
int result = fileChooser.showSaveDialog(RSAUnhidePanel.this);

if (result == JFileChooser.APPROVE_OPTION) {
File f = fileChooser.getSelectedFile();
currentPath = f.getPath();
dstField.setText(f.getAbsolutePath());
}
}
});
}
}
{
backButton = new JButton();
this.add(backButton);
backButton.setText("Back");
backButton.setBounds(24, 294, 105, 49);
backButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {
backButtonActionPerformed(evt);
}
});
}
{
unhideButton = new JButton();
unhideButton.setFocusable(true);
this.add(unhideButton);
unhideButton.setText("UnHide");
unhideButton.setBounds(259, 294, 105, 49);
unhideButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {
unhideButtonActionPerformed(evt);
}
});
}
{
clearButton = new JButton();
this.add(clearButton);
clearButton.setText("Clear");
clearButton.setBounds(385, 294, 105, 49);
clearButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {
clearButtonActionPerformed(evt);
}
});
}
{
dLabel = new JLabel();
this.add(dLabel);
```

```
dLabel.setText("Enter D");
dLabel.setBounds(21, 245, 63, 28);
dLabel.setFont(new java.awt.Font("Trebuchet MS",1,14));
dLabel.setForeground(new java.awt.Color(0,0,255));
}
{
dField = new JTextField();
this.add(dField);
dField.setBounds(98, 245, 112, 28);
}
{
nLabel = new JLabel();
this.add(nLabel);
nLabel.setText("Enter N");
nLabel.setBounds(245, 245, 84, 28);
nLabel.setFont(new java.awt.Font("Trebuchet MS",1,14));
nLabel.setForeground(new java.awt.Color(0,0,255));
}
{
nField = new JTextField();
this.add(nField);
nField.setBounds(329, 245, 161, 28);
}
} catch (Exception e) {
e.printStackTrace();
}
}

private void unhideButtonActionPerformed(ActionEvent evt) {
//     System.out.println("unhideButton.actionPerformed, event=" + evt);

String coverFileName = coverField.getText();
String dstFileName = dstField.getText();
String dVal = dField.getText();
String nVal = nField.getText();
String tempDst = "C:\\Windows\\Temp";

Unhide unhider = new Unhide(this);
unhider.unhideUsingRSA(coverFileName, dstFileName, dVal, nVal);
}
```

```
private void clearButtonActionPerformed(ActionEvent evt) {
System.out.println("clearButton.actionPerformed, event=" + evt);
clearFields();
}

private void backButtonActionPerformed(ActionEvent evt){
parent.reset();
}
```

Encryptor.java

```
import java.io.*;
import java.math.BigInteger;

public class Encryptor {
private BigInteger n;
private BigInteger d;
private BigInteger e;

private FileInputStream fin;
private DataOutputStream fout;
private String outputFileName;

public Encryptor() {
}

* @param eVal
 * @return boolean
 */
 void setE(String eVal) {
 e = new BigInteger(eVal);
 }

 BigInteger getE() {
 return e;
 }
 void setN(String nVal){
 n = new BigInteger(nVal);
 }
```

```
/**
* it sets source file path
*
* @param finStr
* @return boolean
*/
boolean setSrc(String finStr) {
try {
File f = new File(finStr);
if(!f.exists())    // file does not exists
return false;
fin = new FileInputStream(f);
} catch (FileNotFoundException ex) {
return false;
}
return true;
}

/**
* it sets destination file path
*
* @param foutStr
* @return boolean
*/
boolean setDst(String foutStr) {
try {
fout = new DataOutputStream(new FileOutputStream(foutStr));
} catch (FileNotFoundException ex) {
return false;
} catch (IOException ex) {
return false;
}
return true;
}
private String getExtension(String fileName){
return fileName.substring(fileName.lastIndexOf("."));
}

public String getOutputFileName(){
```

```
return outputFileName;
}
int encrypt(String src, String eVal, String nVal) {
BigInteger data;
BigInteger cipher;

setE(eVal);
setN(nVal);
setSrc( src );
//JOptionPane.showMessageDialog(null,src);
outputFileName    =    new    String("C:\\Windows\\Temp\\tempe")    +
getExtension(src);
setDst( outputFileName  );
try {
while (fin.available() > 0) {
int input;
input = fin.read();
data = new BigInteger(String.valueOf(input));
cipher = data.modPow(e, n);
fout.writeShort(cipher.intValue()) ;
}
fin.close();
fout.close();
} catch (IOException ex) {
return 0;
}
return 1;
}
}
```

Decryptor.java

```
import java.io.*;
import java.math.BigInteger;
public class Decryptor {
private BigInteger d;
private BigInteger n;
private FileOutputStream fout;
private DataInputStream fin;
private String outputFileName;
private String getExtension(String fileName){
return fileName.substring(fileName.lastIndexOf("."));
}

public String getOutputFileName(){
return outputFileName;
}
/**
* It sets d (private key) value
*
* @param dVal(d value as String)
* @return boolean(success or failure)
*/
public void setD(String dVal) {
d = new BigInteger(dVal);
}

/**
* It sets n (private key) value
*
* @param nVal(nvalue as String)
* @return boolean(success or failure)
*/
public void setN(String nVal) {
n = new BigInteger(nVal);
}

/**
* It sets source file path (encrypted File) value
*
* @param finStr(finStr value as String)
* @return boolean(success or failure)
*/
public boolean setSrc(String finStr) {
try {
fin = new DataInputStream(new FileInputStream(finStr));
} catch (FileNotFoundException ex) {
return false;
} catch (IOException ex) {
return false;
```

```
}
return true;
}

/**
* It sets destination file path (decrypted File) value
*
* @param foutStr(foutStr value as String)
* @return boolean(success or failure)
*/
public boolean setDst(String foutStr) {
try {
fout = new FileOutputStream(foutStr);
} catch (FileNotFoundException ex) {
return false;
}
return true;
}

*
* @return boolean(indicating success or failure)
*/
public boolean decrypt(String src, String dVal, String nVal) {


setD(dVal);
setN(nVal);
setSrc( src );

outputFileName    =    new    String("C:\\Windows\\Temp\\tempd")    +
getExtension(src);
setDst( outputFileName );

try {
BigInteger input;
BigInteger output;
int data;
while (fin.available()>0) {
data = fin.readShort();
input = new BigInteger(String.valueOf(data));
output = input.modPow(d, n);
fout.write(output.intValue());
}
fout.close();
} catch(EOFException ex){
} catch (IOException ex) {
}

return true;
}
}
```

Cryptor.java

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

public class Cryptor{

static {
Security.addProvider(new com.sun.crypto.provider.SunJCE());
}

private Cipher cipher;
private PBEParameterSpec paramSpec;
private PBEKeySpec keySpec;
private KeyGenerator keygen;
private SecretKey secretKey;

public String algorithm;
public String password;
private int encryptTechnique;
private byte[] key; // for Triple DES
IvParameterSpec IvParameters = new IvParameterSpec(
new byte[] { 12, 34, 56, 78, 90, 87, 65, 43 });
```

```
// Salt
byte[] salt =
{(byte)0xc7,(byte)0x73,(byte)0x21,(byte)0x8c,(byte)0x7e,(byte)0xc8,(byte)0x
ee,(byte)0x99};
// Iteration count
int count = 20;
String outputFileName;

public Cryptor(){
setAlgorithm(1);
}

public void setAlgorithm(int choice){
encryptTechnique = choice;

switch(choice){
case 1: algorithm = new String("PBEWithMD5AndDES"); //DES
break;
case 2: algorithm = new String("DESede/CBC/PKCS5Padding"); //Triple DES
break;
default: algorithm = new String("PBEWithMD5AndDES"); //DES
break;
}
}

private void InitiateCipher(int mode){
try{
if(encryptTechnique!=2){   // if not triple des
paramSpec = new PBEParameterSpec(salt, count);
keySpec = new PBEKeySpec(password.toCharArray());
secretKey =
SecretKeyFactory.getInstance(algorithm).generateSecret(keySpec);
}
else{
key = toKey(password);
// Create a DESede key spec from the key
DESedeKeySpec spec = new DESedeKeySpec( key );
// Get the secret key factor for generating DESede keys
SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DESede");
secretKey = keyFactory.generateSecret(spec);
}

cipher = Cipher.getInstance(algorithm);
if(encryptTechnique==2)
cipher.init(mode, secretKey, IvParameters);
else
cipher.init(mode, secretKey, paramSpec);
}
catch(Throwable t) {t.printStackTrace();}
}

private String getExtension(String fileName){
return fileName.substring(fileName.lastIndexOf("."));
}

int encrypt(String inputFileName, String password){
this.password = password;
try{
//   JOptionPane.showMessageDialog(null,"algorithm: "+algorithm);
InitiateCipher(Cipher.ENCRYPT_MODE);
FileInputStream fin = new FileInputStream(inputFileName);
BufferedInputStream in = new BufferedInputStream(fin);

outputFileName = new String("C:\\Windows\\Temp\\tempe") +
getExtension(inputFileName);
//   String outputFileName = new String("tempe") +
getExtension(inputFileName);

FileOutputStream fout = new FileOutputStream(outputFileName);
CipherOutputStream out = new CipherOutputStream(fout, cipher);
int buffer;
while ((buffer = in.read()) != -1)
out.write(buffer);
out.close();
in.close();

return 0;
}
catch(Exception ex){
return 1;
```

```
}
}

int decrypt(String inputFileName, String dstPathName, String password){
this.password = password;

try{
InitiateCipher(Cipher.DECRYPT_MODE);
FileInputStream fin = new FileInputStream(inputFileName);
CipherInputStream in = new CipherInputStream(fin, cipher);

outputFileName = new String(dstPathName + "\\tempd") +
getExtension(inputFileName);
//    String outputFileName = new String("tempd") +
getExtension(inputFileName);
FileOutputStream fout = new FileOutputStream(outputFileName);
BufferedOutputStream out = new BufferedOutputStream(fout);

int buffer;
while ((buffer = in.read()) != -1)
out.write(buffer);
in.close();
out.close();

return 0;
}catch(Exception ex){
return 1;
}
}

public String getOutputFileName(){
return outputFileName;
}


public byte[] toKey(String password){
int pwdlen = password.length();

int i=pwdlen;
while(i < 25){
```

```
password = password + password;
i = password.length();
}
byte[] encryptKey = password.getBytes();

return encryptKey;
}
}
```

PasswordUnhidePanel.java

```
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;

import javax.swing.BorderFactory;
import javax.swing.JButton;

import javax.swing.WindowConstants;
import javax.swing.border.BevelBorder;
import javax.swing.border.TitledBorder;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.JOptionPane;

public class PasswordUnhidePanel extends UnhidePanel {

UnhideGUI parent;

public PasswordUnhidePanel(int encryptTechnique, UnhideGUI parent) {
super();
this.encryptTechnique = encryptTechnique;
this.parent = parent;
initGUI();
}

private void initGUI() {
```

```
try {
this.setPreferredSize(new java.awt.Dimension(504, 364));
this.setLayout(null);
this.setOpaque(false);
this.setForeground(new java.awt.Color(0,128,192));
{
coverFilePanel = new JPanel();
this.add(coverFilePanel);
coverFilePanel.setBounds(14, 28, 483, 77);
coverFilePanel.setOpaque(false);
coverFilePanel.setBorder(BorderFactory.createTitledBorder(null, "Cover
File", TitledBorder.LEADING, TitledBorder.TOP, new
java.awt.Font("Trebuchet MS",0,12), new java.awt.Color(0,0,255)));
coverFilePanel.setLayout(null);
coverFilePanel.setFont(new java.awt.Font("Trebuchet MS",0,12));
coverFilePanel.setForeground(new java.awt.Color(0,128,192));
{
coverField = new JTextField();
coverFilePanel.add(coverField);
coverField.setBounds(21, 28, 343, 28);
}
{
coverBrowseButton = new JButton();
coverFilePanel.add(coverBrowseButton);
coverBrowseButton.setText("browse");
coverBrowseButton.setBounds(385, 28, 84, 28);
coverBrowseButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {
System.out
.println("coverBrowseButton.actionPerformed, event="
+ evt);
JFileChooser fileChooser = new JFileChooser(currentPath);
if (result == JFileChooser.APPROVE_OPTION) {
File f = fileChooser.getSelectedFile();
currentPath = f.getPath();
coverField.setText(f.getAbsolutePath());
dstField.setText(f.getParent());
}
}
});
```

```
}
}
{
dstPanel = new JPanel();
this.add(dstPanel);
dstPanel.setBounds(14, 133, 483, 84);
dstPanel.setBorder(BorderFactory.createTitledBorder(null, "Destination Path",
TitledBorder.LEADING, TitledBorder.TOP, new java.awt.Font("Trebuchet
MS",0,12), new java.awt.Color(0,0,255)));
dstPanel.setLayout(null);
dstPanel.setOpaque(false);
dstPanel.setFont(new java.awt.Font("Trebuchet MS",0,12));
dstPanel.setForeground(new java.awt.Color(0,128,192));
{
dstField = new JTextField();
dstPanel.add(dstField);
dstField.setBounds(21, 35, 343, 28);
}
{
dstBrowseButton = new JButton();
dstPanel.add(dstBrowseButton);
dstBrowseButton.setText("browse");
dstBrowseButton.setBounds(385, 35, 84, 28);
dstBrowseButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {
System.out
.println("dstBrowseButton.actionPerformed, event="
+ evt);

JFileChooser fileChooser = new JFileChooser(currentPath);
fileChooser.setDialogTitle(new String("Select Destination Folder"));
fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
int result = fileChooser.showSaveDialog(PasswordUnhidePanel.this);

if (result == JFileChooser.APPROVE_OPTION) {
File f = fileChooser.getSelectedFile();
currentPath = f.getPath();
dstField.setText(f.getAbsolutePath());
}
}
});
}
}
{
pwdLabel = new JLabel();
this.add(pwdLabel);
pwdLabel.setText("Enter Password");
pwdLabel.setBounds(21, 245, 196, 35);
pwdLabel.setFont(new java.awt.Font("Trebuchet MS",1,14));
pwdLabel.setForeground(new java.awt.Color(0,0,255));
}
{
pwdField = new JPasswordField();
this.add(pwdField);
pwdField.setBounds(252, 245, 238, 28);
}
{
backButton = new JButton();
this.add(backButton);
backButton.setText("Back");
backButton.setBounds(24, 294, 105, 49);
backButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {
backButtonActionPerformed(evt);
}
});
}
{
unhideButton = new JButton();
pwdField.setNextFocusableComponent(unhideButton);
unhideButton.setFocusable(true);
this.add(unhideButton);
unhideButton.setText("UnHide");
unhideButton.setBounds(259, 294, 105, 49);
unhideButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {
unhideButtonActionPerformed(evt);
}
});
}
```

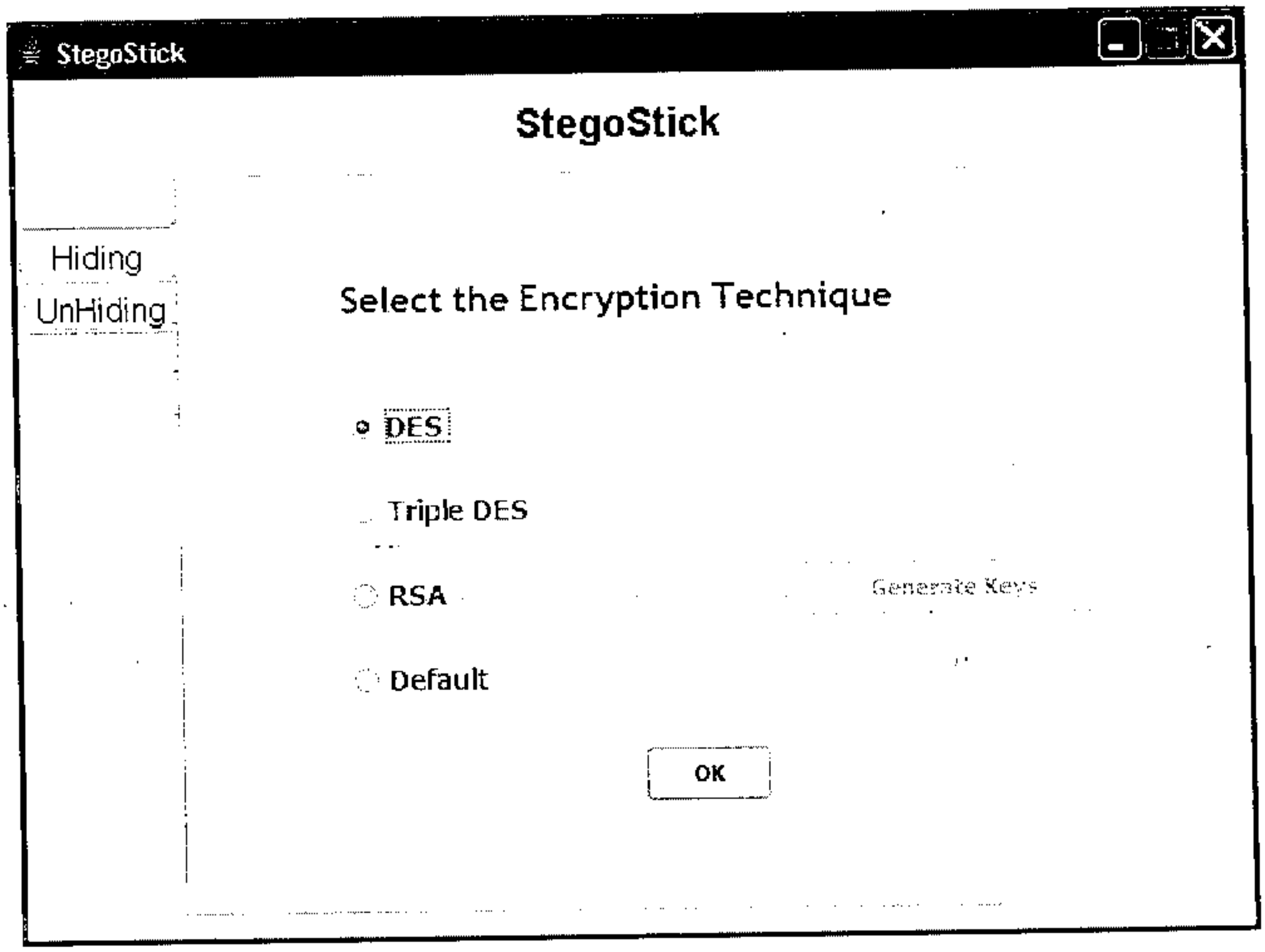
```
{
clearButton = new JButton();
this.add(clearButton);
clearButton.setText("Clear");
clearButton.setBounds(385, 294, 105, 49);
clearButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {
clearButtonActionPerformed(evt);
}
});
}
} catch (Exception e) {
e.printStackTrace();
}
}
```

APPENDIX 2

APPENDIX -2
SCREEN SHOTS

1.Selecting Algorithm:

The user can select the algorithm to be used for the encryption of the password.

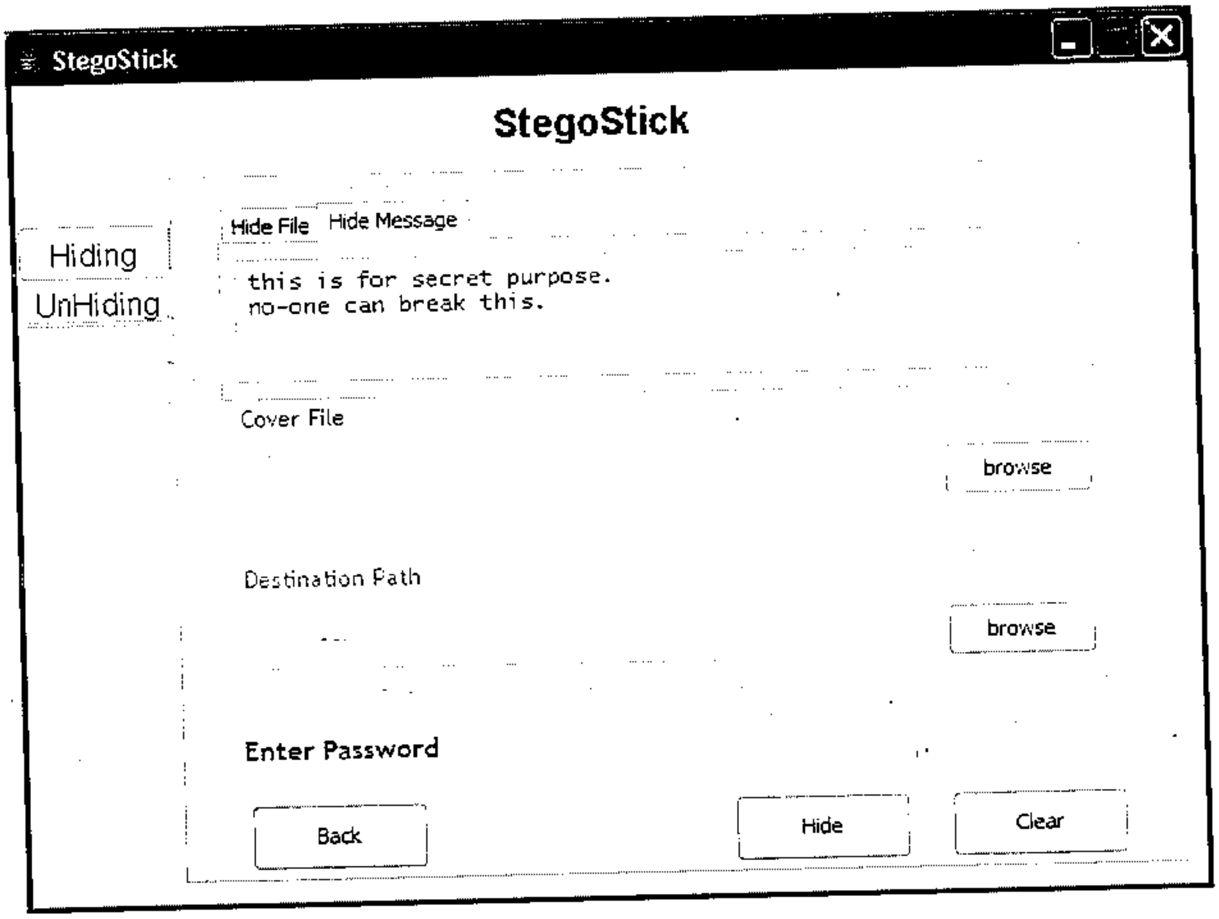


Select the Encryption Technique

[✓ DES] Triple DES RSA Default

2.Hide Message:

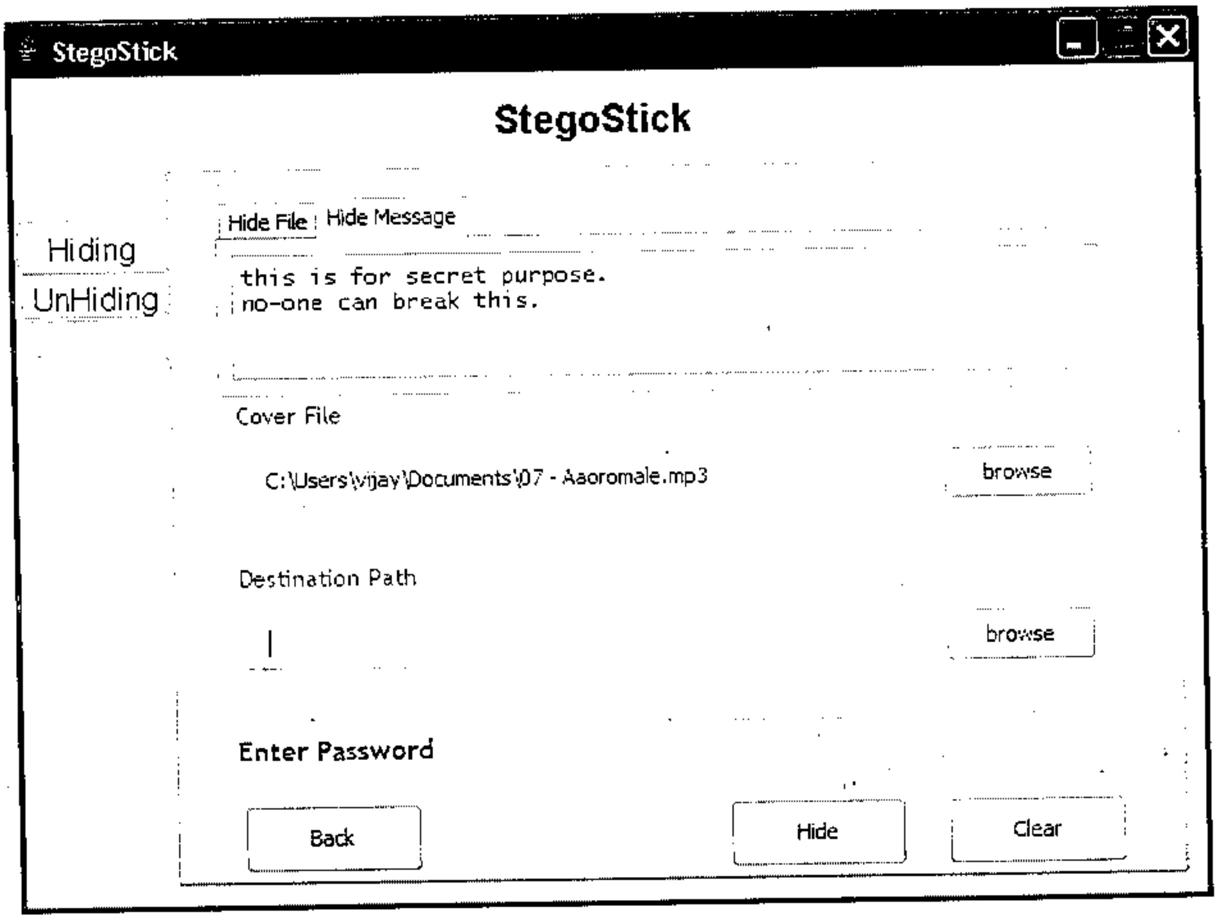
The user can enter the message that has to be hidden inside an audio file.



```
this is for secret purpose.
no-one can break this.
```

3. Selecting cover file:

The user can select the type of cover file namely mp3 in this case, over which the text can be embedded.



```
this is for secret purpose.
no-one can break this.
```

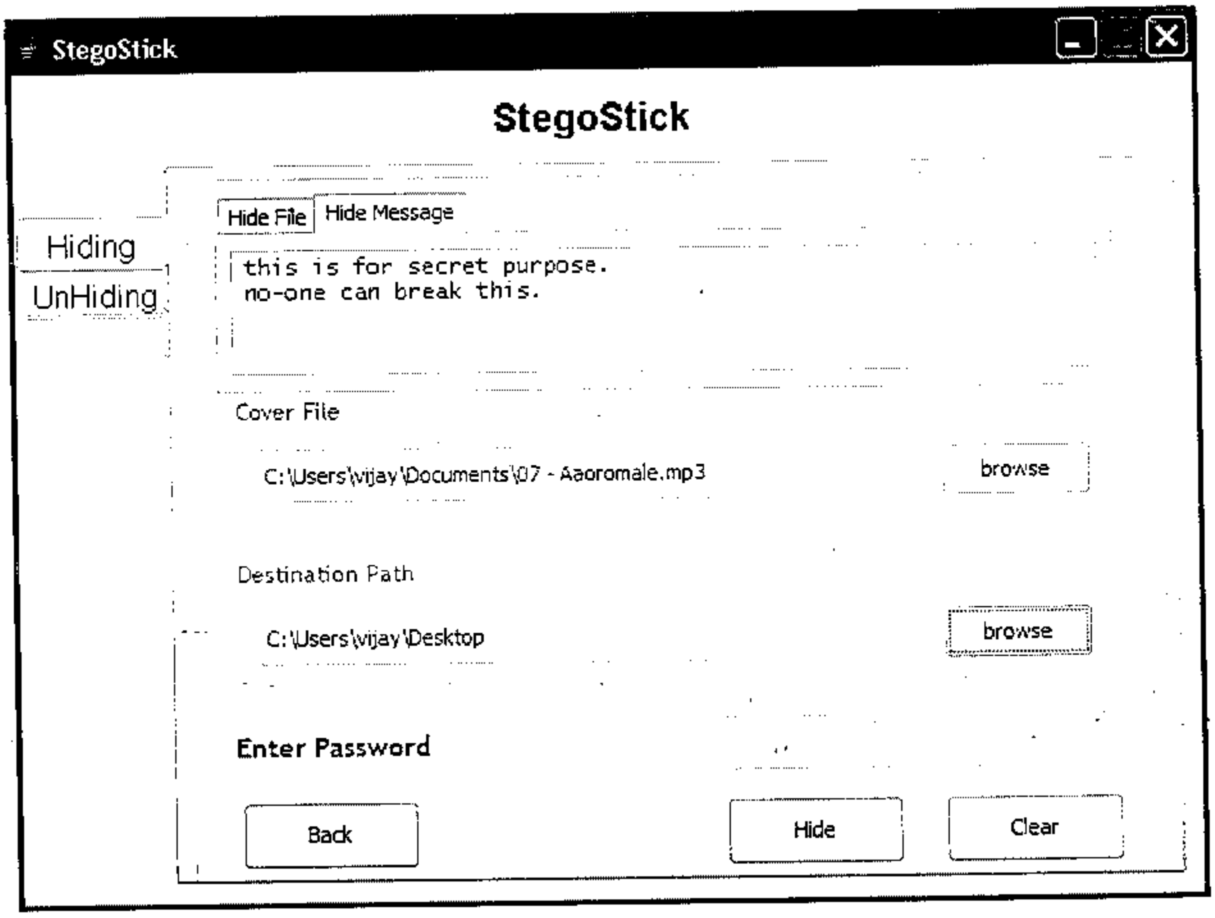
Cover File

C:\Users\vijay\Documents\07 - Aaoromale.mp3

Destination Path

4.Choose destination path:

The user can select the destination path where the embedded audio file with the text has to be saved.



```
this is for secret purpose.
no-one can break this.
```

Cover File

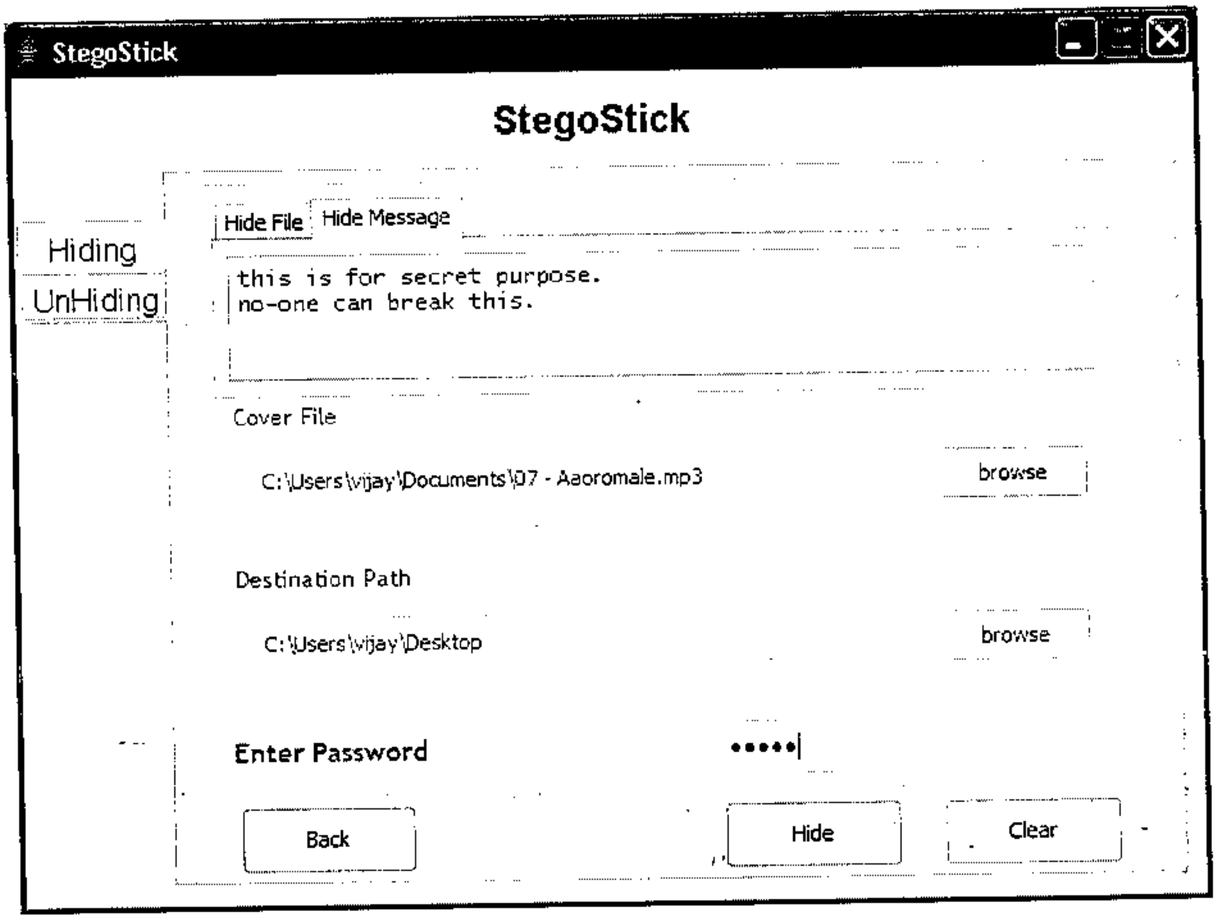
C:\Users\vijay\Documents\07 - Aaoromale.mp3

Destination Path

C:\Users\vijay\Desktop

5.Password:

The user can enter his own password. This password is for security purpose.



```
this is for secret purpose.
no-one can break this.
```

Cover File

C:\Users\vijay\Documents\07 - Aaoromale.mp3

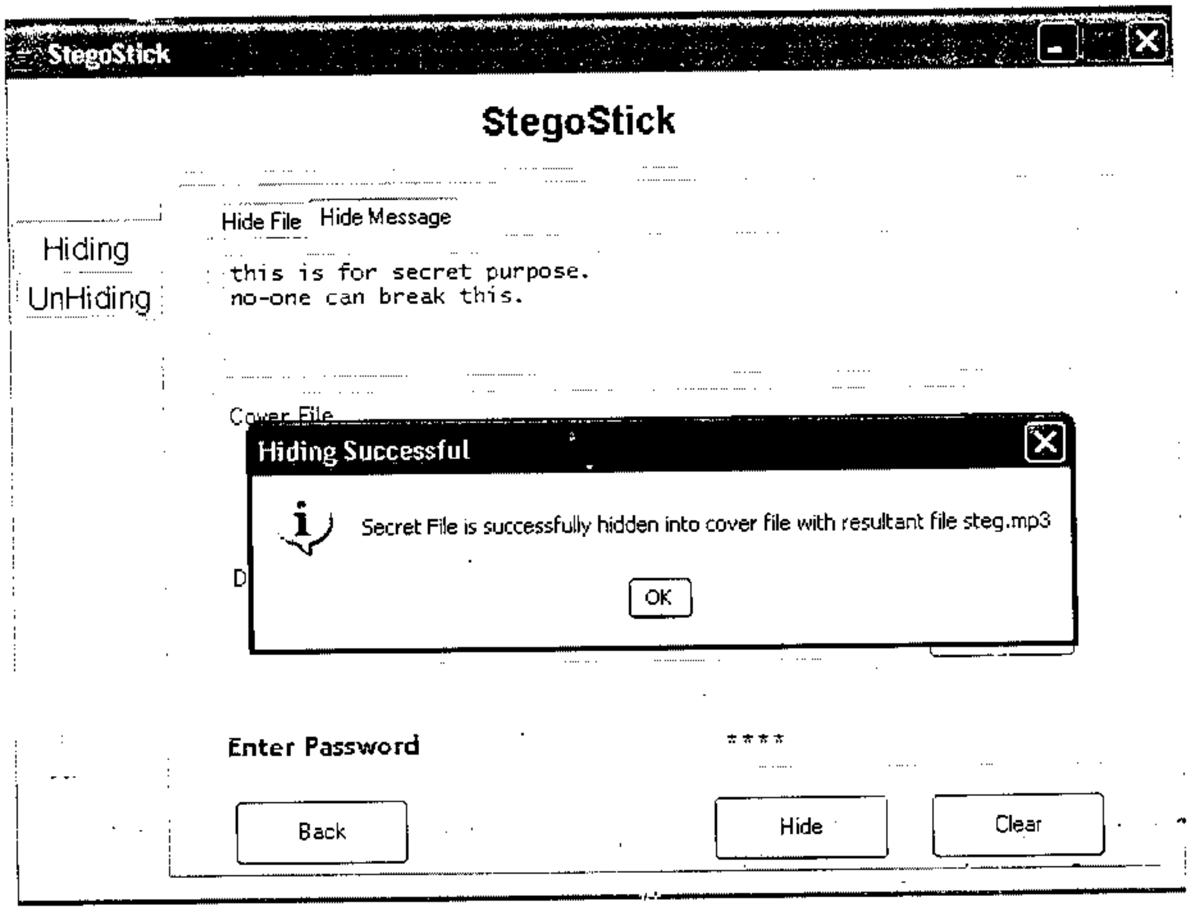
Destination Path

C:\Users\vijay\Desktop

Enter Password ••••••

6. Resultant audio file:

The file has been successfully hidden into the cover file.



```
this is for secret purpose.
no-one can break this.
```

Secret File is successfully hidden into cover file with resultant file steg.mp3