STEGANOCRYPT

P-1569

A PROJECT REPORT

Submitted by

CHANDRASEKAR.K	71201205010
RANGANATHAN.B	71201205044

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 2005

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"STEGANOCRYPT"** is the bonafide work of **"Mr.CHANDRASEKAR.K - 71201205010** and **Mr.RANGANATHAN.B - 71201205044"** who carried out the project work under my supervision.

[illegible]	[illegible] 15/04/05
SIGNATURE	**SIGNATURE**
Dr.S.Thangasamy	**Ms.S.Vani**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
	Lecturer
Dept of Computer Science and Engg,	Dept of Information Technology,
Kumaraguru College of Technology,	Kumaraguru College of Technology,
Coimbatore – 641 006.	Coimbatore – 641 006.

[illegible]	[illegible]
INTERNAL EXAMINER	EXTERNAL EXAMINER
	13/04/2005.

ACKNOWLEDGEMENT

We express our sincere thanks to **Dr.K.Arumugam, B.E (Hons), M.S. (USA), MIE.,** Correspondent, Kumaraguru College of Technology and the management, for proving us an opportunity to undertake this project work.

We express our profound gratitude to **Dr.K.K.Padmanabhan B.Sc. (Engg)., M.Tech., Ph.D.,** Principal, Kumaraguru College of Technology, Coimbatore, for permitting us to undergo a project work.

We are greatly indebted to **Dr. S. Thangasamy, Ph.D.,** Professor and Head of the Department of Computer Science and Engineering, for the immense support he has provided us throughout our project.

We extend our sincere thanks to our project coordinator **Prof.K.R.Baskaran, B.E., M.S.,** Assistant Professor, Department of Information Technology, for his constant support and encouragement.

We would like to express our heart felt thanks to our, **Ms.S.Vani, M.E.,** Lecturer, Department of Information Technology, for her everlasting counseling and untiring help throughout our project.

We like to express our deep sense of gratitude to our families and friends whose help aided the successful completion of the project.

DECLARATION

We,

CHANDRASEKAR.K.	**71201205010**
RANGANATHAN.B.	**71201205044**

Declare that the project entitled **"STEGANOCRYPT"**, submitted in partial fulfillment to Anna University as the project work of Bachelor Of Technology (Information Technology) Degree, is a record of original work done by us under the supervision and guidance of **Ms.S.Vani, M.E.,** Lecturer, Department of Information Technology , Kumaraguru College Of Technology, Coimbatore.

Place: Coimbatore

Date : 18.04.2005

[illegible]
[Chandrasekar.K.]

[illegible]
[Ranganathan.B.]

Project Guided by

[illegible] 15/04/05
[Ms.S.Vani, M.E.,]

ABSTRACT

The project Steganocrypt serves to hide secret message in image files, such that the its very existence is concealed. This project makes use of the concept of Steganography.The main objective is to provide security for the data passed across the network.

Embedding and extraction of information using steganography deals with the concept of hiding secret messages behind other multimedia file. The secret message is the text file and it can be hidden behind another image file moreover this project uses JPEG encoding to minimize the change in image quality to the maximum extent .This process of concealing secret messages provides security for the data to be transferred.

The size of the hidden message should not exceed the size of the image. The sender checks this condition and if it is valid ,the message is hidden and embedded. The sender enhances the security by including the password while sending the message .

The receiver on giving the appropriate password retrieves the hidden message. The embedded data and stegomedium gets separated and cipher text is extracted. Thus the sender is able to send confidential message securely to the receiver.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

• SRS	- Software Requirement Specification
• DWT	- Discrete Wavelet Transform
• DCT	- Discrete Cosine Transform
• ECC	- Error Correction Codes
• QSWT	- Qualified Significant Wavelet Tree
• JPEG	- Joint Photographic Experts Group

1. INTRODUCTION

The word steganography comes from the Greek steganos (covered or secret) and -graphy (writing or drawing) and thus means, literally, covered writing. When using steganography on a computer, you actually hide a message within another file. That resulting file is called a "stego file". The technique to computer steganography is to choose a file capable of hiding a message. A picture, audio, or video file is ideal secure steganographic algorithms hide confidential messages within other, more extensive data (carrier media). An attacker should not be able to find out, that something is embedded in the steganogram (i. e., a steganographically modified carrier medium).

1.1 Existing System

Visual attacks on steganographic systems are based on essential information in the carrier medium that steganographic algorithms overwrite. Adaptive techniques (that bring the embedding rate in line with the carrier content) prevent visual attacks; however, they also reduce the proportion of steganographic information in a carrier medium. Lossy compressed carrier media (JPEG, MP3, . . .) are originally adaptive and immune against visual (and auditory respectively) attacks.

The steganographic tool Jsteg embeds messages in lossy compressed JPEG files. It has a high capacity—e. g., 12 % of the steganogram's size—and, it is immune against visual attacks. However, a statistical attack discovers changes made by Jsteg because Jsteg replaces bits and, thus, it introduces a dependency between the value's frequency of occurrence, that only differ in this bit position (here: LSB). Jsteg influences pairs of the coefficient's frequency of occurrence.

Limitations of the existing system

- Weak against visual or statistical attacks

- Offer only a relatively small capacity for steganographic messages.
- Any statistical attack reliably discovers the existence of embedded message
- Identifying the key can easily decrypt the encrypted text.
- Less security.

1.2 Proposed System

The proposed system is based on an algorithm, which is very strong against statistical and visual attacks. To prevent attacks, the embedding function should use the carrier medium as regular as possible. The embedding density should be the same everywhere.

To accomplish a uniform embedding density Permutative Straddling is used. Moreover a password is used to generate the permutation sequence and while retrieving the same password is essential to retrieve the information. After permutative straddling and embedding it undergoes Huffman encoding which acts as an additional security.

Advantages of the proposed system:

- Strong against visual or statistical attacks
- Offer relatively large capacity for steganographic messages.
- Additional security due to inclusion of password.
- Very high security.

2. SYSTEM REQUIREMENT ANALYSIS

2.1 Project Definition

The project "SteganoCrypt" aims in hiding a secret message (a text file) within an image file using the steganographic technique. The unique feature of this project is the steganographic implementation in DCT Domain with password protection; moreover the project uses JPEG compression to minimize the quality change.

2.2. Project Plan

The requirement phase deals with the sequence of activities in producing the Software Requirement Specification (SRS) document. It must be ensured that all the requirements of the software are elicited and analyzed. In other words the needs of the system are identified.

In the problem analysis phase, the current system is analyzed by study of the existing materials, and the changes to be made in the proposed system are decided upon. A clear understanding of the needs of the system must be framed that leads to actual specification.

Designing aims at how to satisfy the needs of the system. The different modules of the system and the interaction between these modules to produce the desired functionality are identified. During detailed design, the internal logic of each module and their algorithmic design is specified. The major and important decisions are made in this phase.

Coding is the process of translating the design into code. The code developed must be easy to understand. Well-written code can reduce testing and maintenance effort.

Testing is done to check if the requirements of the user are met. It also involves in checking the functionality of each module as per the design document.

2.3 Software Requirement Specification

2.3.1 Introduction

2.3.1.1 Purpose:

The purpose of this document is to specify the requirements of project "SteganoCrypt". It describes the interfaces for the system. The document also bridges the communication gap between the customer and the analyst.

2.3.1.2 Scope:

SRS forms the basics for agreement between the client and the supplier and what the software product will do. It also provides a reference for the validation of the final project.

Any changes made to the SRS in the future will have to go through formal change approval process.

2.3.2 Abbreviation:

SRS	- Software Requirement Specification
DWT	- Discrete Wavelet Transform
DCT	- Discrete Cosine Transform
ECC	- Error Correction Codes
QSWT	- Qualified Significant Wavelet Tree
JPEG	- Joint Photographic Experts Group

2.3.3 Intended Audience and Reading Suggestions:

The document is mainly intended for developers, project managers, users, testers and documentation writers. The information can be understood correctly and easily by start reading from the project scope.

2.3.4 Project Scope:

This project gives a way for secured transmission by using enhanced steganography technique. This is achieved by adopting image processing techniques like Discrete Cosine Transformation. Here the data is embedded into the cover image. The cover object can be text, image, audio or video so that the appearance of the cover object doesn't vary even after the information is hidden [4]. The main objective is secure transmission. So the data to be transmitted is hidden in such a way that a hacker cannot predict the existence of the hidden data.

2.3.5 Perspective Overall Description:

2.3.5.1 Product:

Views on controlling steganography usage have greatly conflicted with the opinions of individuals and businesses. The steganography technique used before was implemented for multimedia objects seperately. This increased the work effort and also the data that is hidden can be obtained by detecting the key through trial and error method. There was no guarantee for safe transmission of the information. The project uses DCT based steganography scheme with password protection It embeds different parts of a single text file into different position of image under the wavelet domain. It is mainly used for secured transmission of secret message.

2.3.5.2 Product Features:

The main function that the product does is transmission of the secret information that is hidden in the cover object. The steganography is robust to basic signal processing operations, including compression, scaling, filtering and the addition of noise.

2.3.5.3 User Classes and Characteristics :

The product will be used mainly in the areas where security is highly needed. It is used in defense where very secured information is transmitted through internet. Also the protection of secret material which is a great challenge for business needs this for protection.

2.3.6 Operating Environment :

Hardware Specification

Processor	: Pentium IV 2.2 GHZ
Ram	: 128 MB
Hard Disk Drive	: 20 GB
Software Specification	
Operating System	: Windows 2000
Software	: Java

2.3.7 Design and Implementation Constraints :

There are not any specific items or issues that will limit the options available to the developers.

2.3.8 User Documentation :

The user will be provided with a user-manual which help to retrieve the secured information. This will be given to the user before the retrieval of the image in which the secret information is hidden.

2.3.9 Assumptions and Dependencies :

The product is developed assuming that the hacker tries to detect and retrieve the secret information from the cover object. Transmission of the cover object is therefore necessary.

2.4 Information Hiding :

2.4.1 Description And Priority :

The information which is to be securely transferred is hidden within the cover image using steganography technique that includes DCT along with password protection. This feature is of very high priority. This feature should be done with great care such that the hacker cannot detect the hidden secret information.

2.4.2 Stimulus/Response Sequences :

The user will hide the information which is to be securely transferred within the cover object. The cover object can be text , image or even an audio file. The project uses an image as a cover object. The cover object will be seen unchanged even after the information is hidden within it.

2.4.3 Functional Requirements :

The action of hiding the secret information inside the cover object should not change the appearance of the cover object. If so it should be detected properly and clarified in such a way that the secret information should not be detected.

2.5 Information Retrieving :

2.5.1 Description and Priority :

The receiver of the information should know the way to retrieve the hidden information from the cover object. This is done by some specified standard which is known only to the sender and the receiver. The retrieved information should be free from error or any other changes.

2.5.2 Stimulus/Response Sequences :

The user will apply the standard way which is decided between the sender and the received. The secret information will be retrieved from the cover object and used for further purpose.

2.6 Functional Requirements :

The information which is retrieved from the cover image should be free from any defects such as error in data or changes from the original format.

2.7 External Interface Requirements :

2.7.1 User Interfaces :

The user interacts with the cover object which is the image where the information is hidden. This image is securely transferred to the destination. The outsiders will see only the image and not the secret information. This cover object will be seen unchanged even after embedding the information.

2.7.2 Software Interfaces :

The project runs on any platform and it includes JAVA class libraries. Here the cover image is transferred between the sender and the receiver. The cover image is seen explicitly and not the secret information. This is done to prevent the detection of the secret information from hackers.

2.7.3 Communications Interface :

The secret information that is hidden within an image can be sent through e-mail or through some other communication interface. The message formatting will be according to HTTP format. The communication standards that will be used may be FTP or HTTP.

2.8 Other Nonfunctional Requirements :

2.8.1 Performance Requirements :

The information should reach the destination within the allotted time. The product should avoid detection of the secret information. The cover object should not change after embedding the secret information with it.

2.8.2 Safety Requirements :

The product should not reveal the hidden information. Even if the hacker detects the existence of some information within the cover object , the hacker should not be able to retrieve the secret information. This is done by making the hiding of information within the cover object in a more complicated way.

2.8.3 Security Requirements :

The user of the product should have proper authorization for accessing the hidden information. The authorization is given in a secure manner(password).

2.8.4 Software Quality Attributes :

The quality characteristics of the product is it is portable, adaptable to changes, reusable. The usage of this product is easy.

3.SYSTEM STUDY

3.1 Steganography :

The word steganography comes from the Greek steganos (covered or secret) and -graphy (writing or drawing) and thus means, literally, covered writing. Steganography is usually given as a synonym for cryptography but it is not normally used in that way. Through recent usage, steganography has come to mean hidden writing, i.e., writing that is not readily discernible to the casual observer. For example, the childhood practice of writing messages in 'invisible ink' would qualify as steganography since the writing is hidden in the sense that it is not obvious that it is there unless you know to look for it.

When using steganography on a computer, you actually hide a message within another file. That resulting file is called a "stego file " The trick to computer steganography is to choose a file capable of hiding a message. A picture, audio, or video file is ideal for several reasons:

These types of files are already compressed by an algorithm. For example, .jpeg, .mp3, .mp4, and .wav formats are all examples of compression algorithms.

These files tend to be large, making it easier to find spots capable of hiding some text.

These files make excellent distracters. That is, few people expect a text message to be hidden within a picture or an audio clip. If the steganographic utility does its job well, a user shouldn't notice a difference in the quality of the image or sound, even though some of the bits have been changed in order to make room for the hidden message.

3.2 Steganography process.

Here, we are hiding the data inside images stored in the JFIF format of the JPEG standard. It was believed that this type of

steganography was impossible, or at least infeasible, since the JPEG standard uses lossy encoding to compress its data. Any hidden data would be overwhelmed by the noise.

3.3 Introduction to JPEG compression

The file format defined by the Joint Photographic Experts Group (JPEG) stores image data in lossy compressed form as quantized frequency coefficients. Fig. 1 shows the compressing steps performed. First, the JPEG compressor cuts the uncompressed bitmap image into parts of 8 by 8 pixels. The discrete cosine transformation (DCT) transfers 8 × 8 brightness values into 8 × 8 frequency coefficients (real numbers). After DCT, the quantization suitably rounds the frequency coefficients to integers in the range 2048 . . . 2047 (lossy step). The histogram in Fig. 2 shows the discrete distribution of the coefficient's frequency of occurrence.

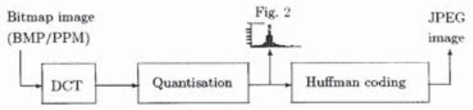


Fig. 1. The flow of information in the JPEG compressor

If we look at the distribution in Fig. 2, we can recognize two characteristic
properties:
1. The coefficient's frequency of occurrence decreases with increasing absolute
value.

2. The decrease of the coefficient's frequency of occurrence decreases with increasing

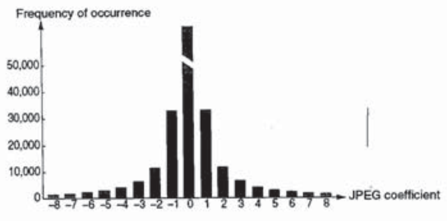
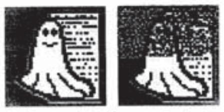


Fig. 2. Histogram for JPEG coefficients after quantisation

absolute value, i. e. the difference between two bars of the histogram in the middle is larger than on the margin.

The technique used by this steganographic implementation is to recognize that JPEG encoding is split into lossy and non-lossy stages. The lossy stages use a discrete cosine transform and a quantization step to compress the image data; the non-lossy stage then uses Huffman coding to further compress the image data. As such, we can insert the steganographic
data into the image data between those two steps and not risk corruption. This insertion step involves more steps such as permutative Straddling with password protection.


Continuous embedding concentrates changes (×)

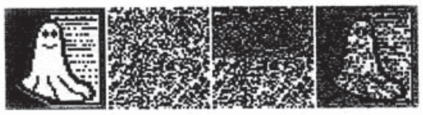

Permutative embedding scatters the changes (×)

Fig. 3 Continuous Embedding And Permutative Embedding

3.4 Permutative Straddling

This method has several benefits. First, the JPEG/JFIF image format has become the de-facto standard for transmission across USENET and for storage on FTP sites. Steganography using these formats would be innocuous compared to that with most other formats. Second, steganographic data in JPEG images is harder to detect with the naked eye than the same data in raw 8-bit or 24-bit images. Just as the aforementioned lossy raw->encoded conversion tends to wipe out steganographic data, the reversed, and encoded -> raw conversion tends to do the same thing. The steganographic inaccuracies in the image are wiped over. In addition, the wide control available over image quantization makes it very difficult to establish whether or not the inaccuracies which do appear are caused by steganographic data or by lower-quality quantization.

The JPEG encoding procedure divides an image into 8x8 blocks of pixels in the YC_bC_r color space. Then they are run through a discrete cosine transform (DCT) and the resulting frequency coefficients are scaled to remove the ones which a human viewer would not detect under normal conditions. If steganographic data is being loaded into the JPEG image, the loading occurs after this step. The lowest-order bits of all non-zero frequency coefficients are replaced with successive bits from the steganographic source file, and these modified coefficients are sent to the Huffman coder.

3.5 Java

3.5.1 Introduction:

Java is an object-oriented programming language with a built-in application programming interface (API) that can handle graphics and user interfaces and that can be used to create applications or applets. Because of its rich set of API's, similar to Macintosh and Windows, and its platform independence, Java can also be thought of as a platform in itself. Java also has standard libraries for doing mathematics.

Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web. Small Java applications are called Java applets and can be downloaded from a Web server and run on your computer by a Java-compatible Web browser, such as Netscape Navigator or Microsoft Internet Explorer

The inventors are Java wanted to design a language, which could offer solution to some of the problems encountered in modern

programming. They wanted the language to be not only reliable, portable and distributed.

Although the above appears to be a list of buzzwords, they aptly describe the full potential of the language. These features have made Java the first application language of the World Wide Web. Java will also become the primer language for general-purpose stand-alone applications.

3.5.2 Properties of java:

- Compiled and interpreted
- Platform-Independent and Portable
- Robust and Secure
- Multithreading and Interactive
- Dynamic and Extensible

3.5.2.1 Compiled and Interpreted:

Usually a computer language is either compiled or interpreted. Java combines both these approaches thus making Java a two-stage system. First, Java compiler translates source code into what is known as byte code instructions.

3.5.2.2 Platform Independent and Portable:

The most significant contribution of Java over other languages is its portability. Java programs can be easily moved from one computer system to another, anywhere anytime. Changes and upgrades in operating systems, processors and system resources will not force any changes in Java programs. This is the reason why Java has become a popular language for programming on Internet, which inter connects different kinds of systems worldwide. We can download a Java applet

from a remote computer on to our local system via Internet an extension of the user's basic system providing practically unlimited number of accessible applets and applications.

3.5.2.3 Robust And Secure:

Security becomes an important issue for a language that is used for programming on Internet. Threat of virus and abuse of resources is everything. Java systems not only verify all memory access but also ensure that no viruses are communicated with an applet. The absence of pointers in Java ensures that programs cannot gain access to memory locations without proper authorization.

3.5.2.4 Distributed:

Java is designed as a distributed language for creating applications on networks. It has the ability to share both data programs. Java applications can open and access remote objects on Internet as easily as they can in a local system. This enables multiple programmers at multiple remote locations to collaborate and work together on a single project.

3.5.2.5 Multithreaded:

Multithreaded means handling multiple tasks simultaneously. Java supports multithreading programs. This means that we need not wait for the application to finish one task before beginning another. For example, we can listen to an audio clip time download an applet from a distance computer. This feature greatly improves the interactive performance of graphical applications.

3.5.2.6 Dynamic and Extensible:

Java is a dynamic language. Java is capable of dynamically linking in new class libraries methods and objects. Java can also determine the type of class through a query, making it possible to either dynamically link or abort the program, depending on the response

3.5.3 Advantages of JAVA

Java has gained enormous popularity since it first appeared. Its rapid ascension and wide acceptance can be traced to its design and programming features, particularly in its promise that we can write a program once, and run it anywhere. Java was chosen as the programming language for network computers (NC) and has been perceived as a universal front end for the enterprise database. As stated in Java language white paper by Sun Microsystems: "Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, multithreaded, and dynamic."

4. SYSTEM DESIGN

4.1 INPUT DESIGN

This project needs the user to give the stego medium file (image where message is to be embedded) and the text file containing the message to be embedded .the user also specifies the output File name and the location where it need to be stored.

To make the man machine interface more easier all the needed input are got by appropriate buttons with self explanatory labels .The file locations are got from the user using Filechooser dialog with open option this helps to browse and choose the required file.

In addition to this whenever the user selects image file that particular image is displayed in the GUI . The selected file name is displayed using text fields.

Appropriate buttons for embedding or extracting are placed with labels.

4.2 OUTPUT DESIGN

After giving the required file names the embed option is used ,when embedding is finished the steganogram file saved in the given location with given file name.

The steganogram image is also displayed in the GUI near the original image, thus both the images can be compared. When extraction is done the extracted text file is stored in the appropriate location .

4.3 PROCESS DESIGN

Secured file transfer with watermarking deals with the concept of hiding secret messages behind other messages. The secret message is text and it can be hidden behind another text file. This process of concealing secret messages provides security for the data to be transferred.

The sender with secret message to be transferred hides it behind another image. The size of the hidden message should not exceed the size of the image. The sender checks this condition and if it is valid the message is hidden and embedded. The sender enhances the security by including the password while sending the message.

The receiver on giving the appropriate password retrieves the hidden message. The embedded data and cover gets separated and cipher text is decrypted. Thus the sender is able to send confidential message securely to the receiver.

The detailed description of the above process is given below:
The image is on a Jpeg format. Initially the cover image and the text file which is embedded within it are selected. The image is compressed using Discrete Cosine Transformation technique. The next step is to embed the text file into the image which is done by using Least Significant Insertion method. Also for maintaining the quality of the image a JPEG encoder is used. Now this is sent to the receiver. At the receiver side an extraction process is done. Then decompression is done using inverse Discrete Cosine Transform technique.

This process is explained in detail here.

STEP 1:

The cover object which is the image and the text file which is embedded within it are selected to be transmitted.

STEP 2:

The second step is the compression of the image which is the cover object, within which the text file is embedded. The discrete cosine transform (DCT) is an important transform in 2D signal processing.

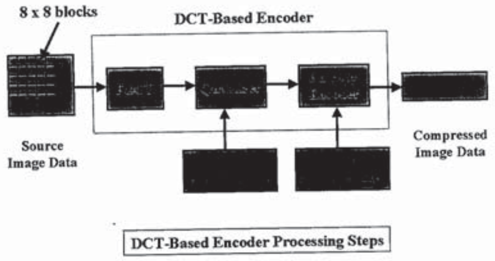


Figure 4.

It is known to be close to optimal in terms of its energy compaction capabilities and can be computed via a fast algorithm. The DCT is used in two international image/video compression standards, Joint Photographic Experts Group (JPEG), and Motion pictures Expert Group (MPEG). Particularly the concentration of large DCT coefficients is in the low-frequency zone. The DCT is known to have excellent energy compaction properties.

STEP 3:

The third step is embedding the text file in the image which is the cover object. The embedding is done by using the Least Significant Bit Insertion method. In a binary code, the least significant bit is, the bit or bit position that is assigned to, or represents the smallest quantity or increment that can be represented by the code.

The Least significant bit (LSB) insertion method is probably the most well known image watermarking technique. It is common, simple approach to embedding information in a graphical image file.

Unfortunately, it is extremely vulnerable to attacks, such as image manipulation. When applying LSB techniques to each byte of a 24 bit image, three bits can be encoded into each pixel. Any changes in the pixel bits will be indiscernible to the human eye. For example, the letter A can be hidden in three pixels. Assume the original pixels are represented by the three 24 bit words below:

(00100111 11101001 11001000) (00100111 11001000 11101001) (11001000 00100111 11101001)

The binary value for the letter A is (10000011). Inserting the binary value of A into the three pixels, starting from the top left byte, would result in:

(00100111 11101000 11001000) (00100110 11001000 11101000) (11001000 00100111 11101001)

The emphasized bits are the only bits that actually changed. The main advantage of LSB insertion is that data can be hidden in the least and second to least bits and still the human eye would be unable to notice it.

When using LSB techniques on 8 bit images, more care needs to be taken .as 8 bit formats are not as forgiving to data changes as 24 bits formats are. Care needs to be taken in the selection of the cover image, so that changes to the data will be visible in the stego-image.

When modifying the LSB bits in 8 bit images, the pointers to entries in the palette are changed. It is important to remember that a change of even one bit could mean the difference between a shade of red and a shade of blue. Such a change would be immediately noticeable on the displayed image, and is thus unacceptable. For this reason, data-hiding experts recommend using grey-scale palettes, where the difference between shades is not pronounced.

STEP 4:

An intermediate step here is the usage of JPEG encoder. This is done to maintain the quality of the image. A JPEG encoder performs a series of transformations on raw image data to produce a compressed output stream, which is then transmitted to a JPEG decoder that executes these transformations in the inverse order to decompress the image.

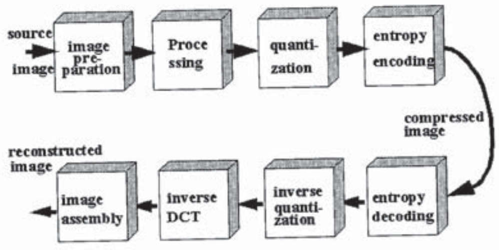


Figure 5. JPEG Encoder Process

JPEG deals with colors in the YUV color space. For each separate color component (Y, U and V), the image is divided into 8x8 pixel blocks of picture elements. Each block is then transformed into a two dimensional DCT matrix. Coefficients with lower frequencies (typically higher values) are encoded first, followed by higher frequencies (with typically small values, near to zero).

The coefficients of the DCT matrix are then quantized. The final step of the JPEG compression consists of an entropy coding.

STEP 5:

The fifth step is the extraction of the text file from the cover object which is the image. Here the exact inverse operation of the embedding process is done.

STEP 6:

The final step is the decompression of the image file which is done by applying inverse discrete cosine transformation technique. One important factor that should be considered in implementing the IDCT is the finite-precision effect. The MPEG compression standard places a strict limit on the error resulting from using a finite number of bits during the IDCT calculation. Such a restriction is necessary to guarantee a consistency in the quality of the reconstructed images regardless of the IDCT algorithm employed. For the 16-bit fixed-point arithmetic as used in our implementation, it was found that at least 4 bits need to be assigned to the fractional part to meet the error requirement. This leaves 12 bits for the integer part, including the sign bit. Although the input value to the IDCT is 12 bits and the output is 9 bits, it is possible for an arithmetic operation to exceed 12 bits during intermediate stages. IDCT implementation meets the precision requirement by using 12 bits of integer and 4 bits of fractional parts.

File Name	File size(bytes)	Embedded size(bytes)	Ratio Embedded to steganogram size	Embedding Efficiency	Quantiser Quality
Lena.jpg	1,562,030	0	(carrier Medium)	-	-
Expo.jpg	129,760	213	0.2%	3.8	80%
Maxsteg.jpg	115,685	15,480	13.4%	1.5	80%

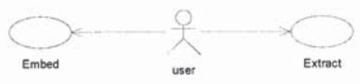
Table 1. Comparison of several JPEG Files Created using SteganoCrypt

4.4 UML DIAGRAMS:

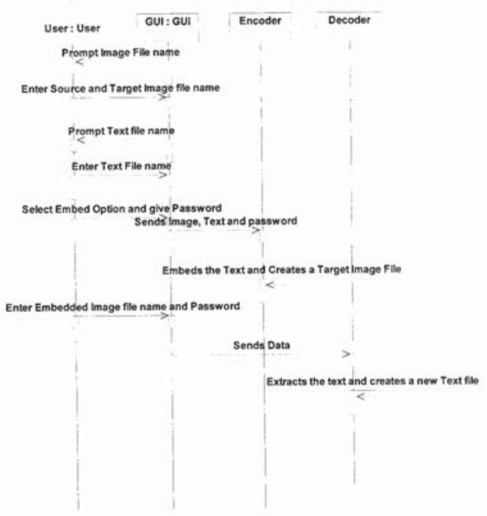
The processes of the system are designed using interaction diagrams and activity diagrams. Interaction diagrams (sequence and collaboration diagrams) describe how group of objects collaborate in some behavior.

The sequence diagram given below is for positive scenario describing the complete process.

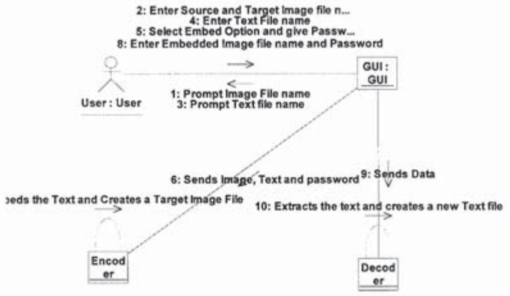
4.4.1 USE CASE DIAGRAM



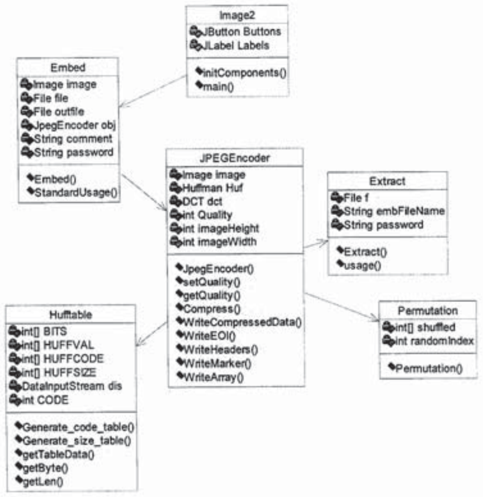
4.4.2 SEQUENCE DIAGRAM


User : User
GUI : GUI
Encoder
Decoder
Prompt Image File name
Enter Source and Target Image file name
Prompt Text file name
Enter Text File name
Select Embed Option and give Password
Sends Image, Text and password
Embeds the Text and Creates a Target Image File
Enter Embedded Image file name and Password
Sends Data
Extracts the text and creates a new Text file

4.4.3 COLLABORATION DIAGRAM


2: Enter Source and Target Image file n...
4: Enter Text File name
5: Select Embed Option and give Passw...
8: Enter Embedded Image file name and Password
GUI : GUI
User : User
1: Prompt Image File name
3: Prompt Text file name
6: Sends Image, Text and password
9: Sends Data
eds the Text and Creates a Target Image File
10: Extracts the text and creates a new Text file
Encoder
Decoder

4.4.4 Class Diagram



5. TESTING

Testing can be separated into Verification & Validation and functional & structural testing.

5.1 UNIT TESTING

The testing of a single program, module or code. Each module of our project were executed separately and checked for the correct working of the module. The output of the message insertion module is the steganogram file without any change in quality of the file. Extraction module is expected to give the steganogram file inside the file given as input to the module. The modules were tested separately and the intended output for both the modules were obtained.

5.2 VERIFICATION

Verification methods are used to ensure the system (software, hardware) compiles with an organization standard and processes relying on review. In our system the modules were tested for correct functionality.

5.3 VALIDATION TESTING

Validation ensures that the system operates according to the plans by executing real-life function. Validation is done for the modules.

5.3.1 Insertion module validation

Insertion module takes the stego medium file as input and produces steganogram file without any change in file's quality. If the file dialog is left empty without selecting any text file then corresponding error message is displayed.

The file chosen inside which text file is to be embedded is analyzed, if the file structure does not match the expected file structure then an error is thrown.

5.3.2 Extraction module validation

The steganogram file is given as input and the message is obtained as output. If the file selected for extraction is not a manipulated file (i.e.) not steganogramed, corresponding message is displayed. If the password given for decoding the extracted text message does not match with the password in the encrypted process "password not match" error is thrown.

5.4 INTEGRATED TESTING

The testing of related modules, program, validates that multiple parts of the system Interact according to the plan. The various modules in our project are integrated, such that Insertion and extraction are done correctly and the tracing of message is verified correctly on time.

5.5 FUNCTIONAL AND STRUCTURAL TESTING

5.5.1 Functional Testing

Functional testing is called Black Box Testing because of no knowledge. They verify all validation tests and inspect how the system performs. Each module is tested for expected output, with out tracing the path of code. Insertion module takes stego medium file and text file as input and outputs the steganogram image file. Extraction Module takes the steganogram image file as input and outputs the decoded text file inserted at the time of insertion process.

5.5.2 Structural testing

Structural testing is called White Box Testing because the internal logic of the system is known. In our project, the Insertion and Extraction of message according to the correct format is tested by tracing the code structurally. The validation of the text file, stego medium file

size validation and validation of steganogram file, password validation for decoding the message in extraction module are all done in structural testing.

5.6 TEST CASES FOR UNIT TESTING

Test Case	Visible State of Component	Input	Expected Output	Actual Output	Comments
Select image	Selecting cover object	Cover image	Selected image is displayed	**Ok**-display the corresponding image	Test whether the correct image is selected
Select text	Selecting the required text to be embedded	Required secret text	Proceed with selected text	**Ok**-display the corresponding path	Test whether the correct path is selected
Set target file	Creating a new target file	Any file name	Saving the target file	**Ok**-display the corresponding text file	Test whether the correct text file is selected
Select embed option	Embedded file displayed to the user	Selected image and text	Proceed with the embedding operation	**Ok**-Proceed with embedding	Test whether the embedding operation is done
Select extract option	Extracted text visible to user	Target file	Text extracted to the target file	**Ok**-Obtained the required text	Test whether the text is received

5.7 TEST CASES FOR INTEGRATION TESTING

The purpose of this integration testing is to test the project as a whole after combining the individual modules..

Test Case	Visible State of Component	Input	Expected Output	Actual Output	Comments
Select image	Selecting cover object	Cover image	Selected image is displayed	**Ok**-display the corresponding image	Test whether the correct image is selected
Select text	Selecting the required text to be embedded	Required secret text	Proceed with selected text	**Ok**-display the corresponding path	Test whether the correct path is selected
Set target file	Creating a new target file	Any file name	Saving the target file	**Ok**-display the steganogram file	Test whether the correct steganogram file is selected
Select embed option	Embedded file displayed to the user	Selected image and text	Proceed with the embedding operation	**Ok**-Proceed with embedding	Test whether the embedding operation is done

5.8 BLACK BOX TESTING

The purpose of black box testing is to check out the inputs and the expected output for the system. The input is a JPEG image that is to be

kept as a cover image and a text file which is the secret information to be embedded. The output is the extraction of the secret information from the cover image.

5.9 WHITE BOX TESTING

The purpose of white box testing is to check out whether the logical operation of the system works well. Here the processing of the input is considered. A threshold frequency is set and the values which are lower than this limit are made to zero and the values higher than the threshold values are considered for embedding and extracting. The information is embedded on the cover image by concentrating on the high frequency components. The extraction operation involves the inverse of the embedding operation. This is done for the proper retrieval of the secret information.

6. CONCLUSION

Several methods for hiding data in text, images, and audio exist with an appropriate introduction to the environments of each medium, as well as strengths and weaknesses of each method. Most data-hiding systems take advantage of human perceptual weaknesses, but have weaknesses of their own. Generally in such methods it seems that no system of data-hiding is totally immune to attack.

However, steganography has its place in security. It, in no way can replace cryptography, but is intended to supplement it. Its application in fingerprinting, for use in detection of unauthorized, illegally copied material, is continually being realized and developed.

Also, in places where standard cryptography and encryption is outlawed, steganography can be used to convert data transmission. Steganography, formerly just an interest of the military, is now gaining popularity among the masses. Soon, any computer user will be able to put his own secret message on his artistic creations.

7. FUTURE ENHANCEMENT

In the proposed, only one file can be hidden behind an image. The future enhancement is to hide more number of files in a single image. At present, only bmp image and JPEG is used. But, later TIFF and GIF can also be used. Authentication can also be given using fingerprints and voice recognition.

APPENDIX - 1

Sample code

1. DCT Coefficient Extractor Class

```
class DCT {
public int N = 8;
    public int QUALITY = 80;
      public Object[] quantum = new Object[2];
      public Object[] Divisors = new Object[2];
   public int[] quantum_luminance = new int[N*N];
      public double[] DivisorsLuminance = new double[N*N];
      public int[] quantum_chrominance = new int[N*N];
      public double[] DivisorsChrominance = new double[N*N];
      /**
   * Constructs a new DCT object. Initializes the cosine transform
matrix
   * these are used when computing the DCT and it's inverse. This
also
   * initializes the run length counters and the ZigZag sequence.
Note that
   * the image quality can be worse than 25 however the image will
be
   * extemely pixelated, usually to a block size of N.
   *
   */
   public DCT(int QUALITY) {
     initMatrix(QUALITY);
   }
    private void initMatrix(int quality) {
```

```
      ouble[] AANscaleFactor = { 1.0, 1.387039845, 1.306562965,
 1.175875602,
 1.0, 0.785694958, 0.541196100, 0.275899379};
 int i;
 int j;
 int index;
 int Quality;
 int temp;
 // converting quality setting to that specified in the
jpeg_quality_scaling
 // method in the IJG Jpeg-6a C libraries
  Quality = quality;
 if (Quality <= 0)
    Quality = 1;
 if (Quality > 100)
    Quality = 100;
 if (Quality < 50)
    Quality = 5000 / Quality;
 else
    Quality = 200 - Quality * 2;
 // Creating the luminance matrix
  quantum_luminance[0]=16;
 quantum_luminance[1]=11;
 quantum_luminance[2]=10;
 quantum_luminance[3]=16;
 quantum_luminance[4]=24;
 quantum_luminance[5]=40;
 quantum_luminance[6]=51;
 quantum_luminance[7]=61;
```

```
quantum_luminance[8]=12;
quantum_luminance[9]=12;
quantum_luminance[10]=14;
quantum_luminance[11]=19;
quantum_luminance[12]=26;
quantum_luminance[13]=58;
quantum_luminance[14]=60;
quantum_luminance[15]=55;
quantum_luminance[16]=14;
quantum_luminance[17]=13;
quantum_luminance[18]=16;
quantum_luminance[19]=24;
quantum_luminance[20]=40;
quantum_luminance[21]=57;
quantum_luminance[22]=69;
quantum_luminance[23]=56;
quantum_luminance[24]=14;
quantum_luminance[25]=17;
quantum_luminance[26]=22;
quantum_luminance[27]=29;
quantum_luminance[28]=51;
quantum_luminance[29]=87;
quantum_luminance[30]=80;
quantum_luminance[31]=62;
quantum_luminance[32]=18;
quantum_luminance[33]=22;
quantum_luminance[34]=37;
quantum_luminance[35]=56;
quantum_luminance[36]=68;
```

```
quantum_luminance[37]=109;
quantum_luminance[38]=103;
quantum_luminance[39]=77;
quantum_luminance[40]=24;
quantum_luminance[41]=35;
quantum_luminance[42]=55;
quantum_luminance[43]=64;
quantum_luminance[44]=81;
quantum_luminance[45]=104;
quantum_luminance[46]=113;
quantum_luminance[47]=92;
quantum_luminance[48]=49;
quantum_luminance[49]=64;
quantum_luminance[50]=78;
quantum_luminance[51]=87;
quantum_luminance[52]=103;
quantum_luminance[53]=121;
quantum_luminance[54]=120;
quantum_luminance[55]=101;
quantum_luminance[56]=72;
quantum_luminance[57]=92;
quantum_luminance[58]=95;
quantum_luminance[59]=98;
quantum_luminance[60]=112;
quantum_luminance[61]=100;
quantum_luminance[62]=103;
quantum_luminance[63]=99;
 for (j = 0; j < 64; j++) {
   temp = (quantum_luminance[j] * Quality + 50) / 100;
```

```
        if ( temp <= 0) temp = 1;
        if (temp > 255) temp = 255;
        quantum_luminance[j] = temp;
     }  index = 0;
     for (i = 0; i < 8; i++) {
        for (j = 0; j < 8; j++) {
           // The divisors for the LL&M method (the slow integer method
used in
           // jpeg 6a library).
           //                        DivisorsLuminance[index] = ((double)
quantum_luminance[index]) << 3;
           // The divisors for the AAN method (the float method used in
jpeg 6a library.
           DivisorsLuminance[index] = (double) ((double)1.0/((double)
quantum_luminance[index] * AANscaleFactor[i] * AANscaleFactor[j] *
(double) 8.0));
           index++;
        }
     }

      // Creating the chrominance matrix
      quantum_chrominance[0]=17;
     quantum_chrominance[1]=18;
     quantum_chrominance[2]=24;
     quantum_chrominance[3]=47;
     quantum_chrominance[4]=99;
     quantum_chrominance[5]=99;
     quantum_chrominance[6]=99;
     quantum_chrominance[7]=99;
```

```
quantum_chrominance[8]=18;
quantum_chrominance[9]=21;
quantum_chrominance[10]=26;
quantum_chrominance[11]=66;
quantum_chrominance[12]=99;
quantum_chrominance[13]=99;
quantum_chrominance[14]=99;
quantum_chrominance[15]=99;
quantum_chrominance[16]=24;
quantum_chrominance[17]=26;
quantum_chrominance[18]=56;
quantum_chrominance[19]=99;
quantum_chrominance[20]=99;
quantum_chrominance[21]=99;
quantum_chrominance[22]=99;
quantum_chrominance[23]=99;
quantum_chrominance[24]=47;
quantum_chrominance[25]=66;
quantum_chrominance[26]=99;
quantum_chrominance[27]=99;
quantum_chrominance[28]=99;
quantum_chrominance[29]=99;
quantum_chrominance[30]=99;
quantum_chrominance[31]=99;
quantum_chrominance[32]=99;
quantum_chrominance[33]=99;
quantum_chrominance[34]=99;
quantum_chrominance[35]=99;
quantum_chrominance[36]=99;
```

```
quantum_chrominance[37]=99;
quantum_chrominance[38]=99;
quantum_chrominance[39]=99;
quantum_chrominance[40]=99;
quantum_chrominance[41]=99;
quantum_chrominance[42]=99;
quantum_chrominance[43]=99;
quantum_chrominance[44]=99;
quantum_chrominance[45]=99;
quantum_chrominance[46]=99;
quantum_chrominance[47]=99;
quantum_chrominance[48]=99;
quantum_chrominance[49]=99;
quantum_chrominance[50]=99;
quantum_chrominance[51]=99;
quantum_chrominance[52]=99;
quantum_chrominance[53]=99;
quantum_chrominance[54]=99;
quantum_chrominance[55]=99;
quantum_chrominance[56]=99;
quantum_chrominance[57]=99;
quantum_chrominance[58]=99;
quantum_chrominance[59]=99;
quantum_chrominance[60]=99;
quantum_chrominance[61]=99;
quantum_chrominance[62]=99;
quantum_chrominance[63]=99;

for (j = 0; j < 64; j++) {
```

```
        temp = (quantum_chrominance[j] * Quality + 50) / 100;
        if ( temp <= 0) temp = 1;
        if (temp >= 255) temp = 255;
        quantum_chrominance[j] = temp;
    }
    index = 0;
    for (i = 0; i < 8; i++) {
        for (j = 0; j < 8; j++) {
            // The divisors for the LL&M method (the slow integer method
used in
            // jpeg 6a library).
            //          divisorsChrominance[index]          =          ((double)
quantum_chrominance[index])<< 3;
            // The divisors for the AAN method (the float method used in
jpeg 6a library.
            DivisorsChrominance[index]=(double)((double)1.0/((double)
quantum_chrominance[index] * AANscaleFactor[i] * AANscaleFactor[j]
* (double)8.0));
            index++;
        }
    }
    // quantum and Divisors are objects used to hold the appropriate
matices
    quantum[0] = quantum_luminance;
    Divisors[0] = DivisorsLuminance;
    quantum[1] = quantum_chrominance;
    Divisors[1] = DivisorsChrominance;
    }
    public double[][] forwardDCTExtreme(float[][] input) {
```

```
    double output[][] = new double[N][N];
    double tmp0, tmp1, tmp2, tmp3, tmp4, tmp5, tmp6, tmp7;
    double tmp10, tmp11, tmp12, tmp13;
    double z1, z2, z3, z4, z5, z11, z13;
    int i;
    int j;
    int v, u, x, y;
    for (v = 0; v < 8; v++) {
      for (u = 0; u < 8; u++) {
        for (x = 0; x < 8; x++) {
          for (y = 0; y < 8; y++) {
            output[v][u]                                        +=
((double)input[x][y])*Math.cos(((double)(2*x                          +
1)*(double)u*Math.PI)/(double)16)*Math.cos(((double)(2*y              +
1)*(double)v*Math.PI)/(double)16);
          }
        }
        output[v][u]     *=     (double)(0.25)*((u     ==     0)     ?
((double)1.0/Math.sqrt(2))     :     (double)     1.0)*((v     ==     0)     ?
((double)1.0/Math.sqrt(2)) : (double) 1.0);
      }
    }
    return output;
  }
    public double[][] forwardDCT(float[][] input) {
    double output[][] = new double[N][N];
    double tmp0, tmp1, tmp2, tmp3, tmp4, tmp5, tmp6, tmp7;
    double tmp10, tmp11, tmp12, tmp13;
    double z1, z2, z3, z4, z5, z11, z13;
```

```
    int i;
    int j;
    // Subtracts 128 from the input values
    for (i = 0; i < 8; i++) {
      for(j = 0; j < 8; j++) {
        output[i][j] = ((double)input[i][j] - (double)128.0);
        //                input[i][j] -= 128;
      }
    }
    for (i = 0; i < 8; i++) {
      tmp0 = output[i][0] + output[i][7];
      tmp7 = output[i][0] - output[i][7];
      tmp1 = output[i][1] + output[i][6];
      tmp6 = output[i][1] - output[i][6];
      tmp2 = output[i][2] + output[i][5];
      tmp5 = output[i][2] - output[i][5];
      tmp3 = output[i][3] + output[i][4];
      tmp4 = output[i][3] - output[i][4];
      tmp10 = tmp0 + tmp3;
      tmp13 = tmp0 - tmp3;
      tmp11 = tmp1 + tmp2;
      tmp12 = tmp1 - tmp2;
      output[i][0] = tmp10 + tmp11;
      output[i][4] = tmp10 - tmp11;
      z1 = (tmp12 + tmp13) * (double) 0.707106781;
      output[i][2] = tmp13 + z1;
      output[i][6] = tmp13 - z1;
      tmp10 = tmp4 + tmp5;
      tmp11 = tmp5 + tmp6;
```

```
        tmp12 = tmp6 + tmp7;
        z5 = (tmp10 - tmp12) * (double) 0.382683433;
        z2 = ((double) 0.541196100) * tmp10 + z5;
        z4 = ((double) 1.306562965) * tmp12 + z5;
        z3 = tmp11 * ((double) 0.707106781);
        z11 = tmp7 + z3;
        z13 = tmp7 - z3;
        output[i][5] = z13 + z2;
        output[i][3] = z13 - z2;
        output[i][1] = z11 + z4;
        output[i][7] = z11 - z4;
      }
      for (i = 0; i < 8; i++) {
        tmp0 = output[0][i] + output[7][i];
        tmp7 = output[0][i] - output[7][i];
        tmp1 = output[1][i] + output[6][i];
        tmp6 = output[1][i] - output[6][i];
        tmp2 = output[2][i] + output[5][i];
        tmp5 = output[2][i] - output[5][i];
        tmp3 = output[3][i] + output[4][i];
        tmp4 = output[3][i] - output[4][i];
        tmp10 = tmp0 + tmp3;
        tmp13 = tmp0 - tmp3;
        tmp11 = tmp1 + tmp2;
        tmp12 = tmp1 - tmp2;
        output[0][i] = tmp10 + tmp11;
        output[4][i] = tmp10 - tmp11;
        z1 = (tmp12 + tmp13) * (double) 0.707106781;
        output[2][i] = tmp13 + z1;
```

```
        output[6][i] = tmp13 - z1;
        tmp10 = tmp4 + tmp5;
        tmp11 = tmp5 + tmp6;
        tmp12 = tmp6 + tmp7;
        z5 = (tmp10 - tmp12) * (double) 0.382683433;
        z2 = ((double) 0.541196100) * tmp10 + z5;
        z4 = ((double) 1.306562965) * tmp12 + z5;
        z3 = tmp11 * ((double) 0.707106781);
        z11 = tmp7 + z3;
        z13 = tmp7 - z3;
        output[5][i] = z13 + z2;
        output[3][i] = z13 - z2;
        output[1][i] = z11 + z4;
        output[7][i] = z11 - z4;
      }
      return output;
    }
    public int[] quantizeBlock(double[][] inputData, int code) {
      int outputData[] = new int[N*N];
      int i, j;
      int index;
      index = 0;
      for (i = 0; i < 8; i++) {
        for (j = 0; j < 8; j++) {
          // The second line results in significantly better compression.
          outputData[index]  =  (int)(Math.round(inputData[i][j]  *
(((double[]) (Divisors[code]))[index])));
          //              outputData[index] = (int)(((inputData[i][j] *
(((double[]) (Divisors[code]))[index])) + 16384.5) -16384);
```

```
          index++;
        }
      }
      return outputData;
    }
    public int[] quantizeBlockExtreme(double[][] inputData, int code) {
      int outputData[] = new int[N*N];
      int i, j;
      int index;
      index = 0;
      for (i = 0; i < 8; i++) {
        for (j = 0; j < 8; j++) {
          outputData[index]  =  (int)(Math.round(inputData[i][j]  /
(double)(((int[]) (quantum[code]))[index])));
          index++;
        }
      }
      return outputData; } }
```

2. Jpeg Encoder

```
import java.awt.*;
import java.io.*;
import java.util.*;
import java.lang.*;
public class Jpeg {
  public static void StandardUsage() {
    System.out.println("JpegEncoder for Java(tm) Version 0.9");
    System.out.println("");
```

```
    System.out.println("Program usage: java Jpeg \"InputImage\".\"ext\"
Quality [\"OutputFile\"[.jpg]]");
    System.out.println("");
    System.out.println("Where \"InputImage\" is the name of an existing
image in the current directory.");
    System.out.println("  (\"InputImage may specify a directory, too.)
\"ext\" must be .tif, .gif,");
    System.out.println("  or .jpg.");
    System.out.println("Quality is an integer (0 to 100) that specifies
how similar the compressed");
    System.out.println("  image is to \"InputImage.\"  100 is almost
exactly like \"InputImage\" and 0 is");
    System.out.println("  most dissimilar.  In most cases, 70 - 80 gives
very good results.");
    System.out.println("\"OutputFile\" is an optional argument.  If
\"OutputFile\" isn't specified, then");
    System.out.println("  the input file name is adopted.  This program
will NOT write over an existing");
    System.out.println("  file.  If a directory is specified for the input
image, then \"OutputFile\"");
    System.out.println("  will be written in that directory.  The extension
\".jpg\" may automatically be");
    System.out.println("  added.");
    System.out.println("");
  System.exit(0);
  }
  public static void main(String args[]) {
      Image image = null;
      FileOutputStream dataOut = null;
```

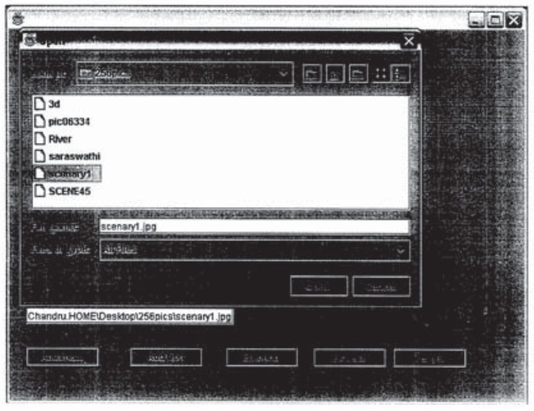
```
        File file, outFile;
        JpegEncoder jpg;
        String string = new String();
        int i, Quality = 80;
        if (args.length < 2)
            StandardUsage();
        if (!args[0].endsWith(".jpg") && !args[0].endsWith(".tif") &&
!args[0].endsWith(".gif"))
            StandardUsage();
        if (args.length < 3) {
            string = args[0].substring(0, args[0].lastIndexOf(".")) +
".jpg";
        }
        else {
            string = args[2];
            if (string.endsWith(".tif") || string.endsWith(".gif"))
                string = string.substring(0, string.lastIndexOf("."));
            if (!string.endsWith(".jpg"))
                string = string.concat(".jpg"  }
        outFile = new File(string);
        i = 1;
        while (outFile.exists()) {
      outFile = new File(string.substring(0, string.lastIndexOf(".")) +
(i++) + ".jpg");
            if (i > 100)
                System.exit(0);
        }
        file = new File(args[0]);
            if (file.exists()) {
```

```
            try {
                dataOut = new FileOutputStream(outFile);
            } catch(IOException e) {}
            try {
                Quality = Integer.parseInt(args[1]);
            } catch (NumberFormatException e) {
                StandardUsage();
            }
            image                                        =
Toolkit.getDefaultToolkit().getImage(args[0]);
            jpg = new JpegEncoder(image, Quality, dataOut,"");
            jpg.Compress();
            try {
                dataOut.close();
            } catch(IOException e) {}
        }
        else {
            System.out.println("I couldn't find " + args[0] + ". Is
it in another directory?"); }
    System.exit(0
```

Appendix - 2

Screen Shots:

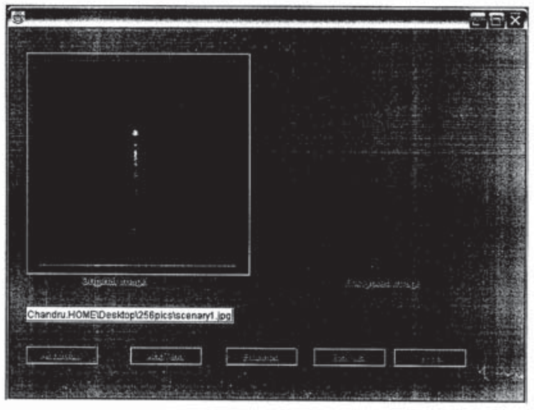
GUI for File Choosing:



scenary1.jpg

Chandru.HOME\Desktop\256pics\scenary1.jpg

GUI With the display of selected image:



Chandru.HOME\Desktop\256pics\scenary1.jpg

GUI with both output and input mage:

Chandru.HOME\Desktop\256pics\scenary1.jpg

F:\Academics\Project\stegno\input.txt.txt

9. REFERENCES

[1] Karen Korhorn, **"Steganography Uses And Effects On Society",**Volume 1,2002 Edition : Rana Publications.

[2] Chun-Hsiang Huang and Ja-Ling Wu, **"Attacking Visible Watermarking schemes"** (2004) , IEEE Transactions on Multimedia, VOL. 6, NO. 1, February 2004.

[3] Neil F. Johnson and Sushil Jajodia , **"Steganalysis of Images Created Using current Steganography Software"** Edition : TATA Mc GrawHill Ltd.

[4] Stefan Hetel, **"A Survey of Steganography",**Volume 1,2002 Edition :Scrabel Lotaa Ltd.

[5] Michael Swanson, Kobayashi and Ahmed Twefik (1998), **"Multimedia Data-Embedding and Watermarking Technologies",** Published in Proceedings of the IEEE, VOL.86, NO. 6, June 1998.

[6] Pic-Wah Chan and Michael R.Lyu, **"DWT Based Digital Video Watermarking Scheme With Error Correcting Code"** ,Edition 2,2004.

[7] Amir Schricker and Erica Richstad, **"Wavelet Based Watermarking Algorithms For Digital Images"** ,Volume 3,2000 .