

ENTERPRISE SYSTEM SOLUTION FOR C-DAC BY PROVIDING STORAGE LEVEL SECURITY USING VISUAL CRYPTOGRAPHY

A DISSERTATION

*Submitted in partial fulfilment of the
requirements for the award of the degree*

of

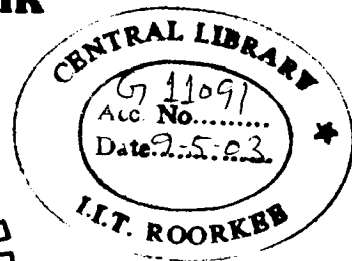
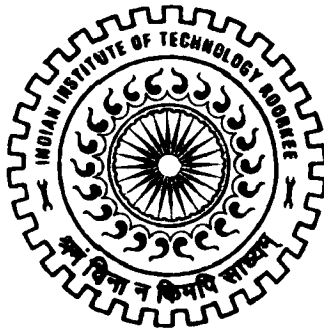
MASTER OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

By

SHILPA KAUSHIK



**ER & DCI
NOIDA**

**IIT Roorkee-ER&DCI, Noida
C-56/1, "Anusandhan Bhawan"
Sector 62, Noida-201 307**

FEBRUARY, 2003

Enrolment No. 019049

621.380285
KAU


CANDIDATE'S DECLARATION

I hereby declare that the work presented in this dissertation titled "ENTERPRISE SYSTEM SOLUTION FOR C-DAC BY PROVIDING STORAGE LEVEL SECURITY USING VISUAL CRYPTOGRAPHY", in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Information Technology**, submitted in IIT, Roorkee – ER&DCI Campus, Noida, is an authentic record of my own work carried out during the period from August 2002 to February, 2003 under the guidance of **Mr. Rajiv Phougat**, Project Leader, Centre for Development of Advanced Computing, Hauz Khas, New Delhi.

The matter embodied in this dissertation has not been submitted by me for award of any other degree or diploma.

Date: 24.02.03

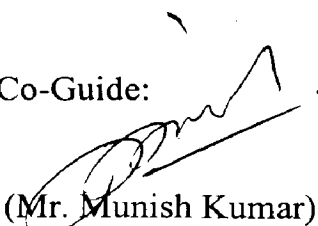
Place: Noida


(Shilpa Kaushik)

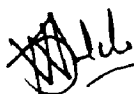
CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

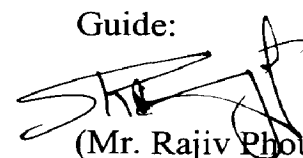
Co-Guide:


(Mr. Munish Kumar)

Project Engineer,
ER&DCI, Noida



Guide:


(Mr. Rajiv Phougat)

Project Leader,
C-DAC, New Delhi.

Date: 24-02-03

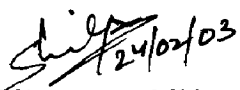
Place: Noida

ACKNOWLEDGEMENT

The work presented in this report would not have been completed without the guidance and support of many people. I take this opportunity to thank **Prof. Prem Vratt**, Director, **Prof. A.K.Awasthi**, Dean PGS&R and **Dr. R.P. Aggarwal**, Course Coordinator, M.Tech. (IT) of IIT Roorkee for providing me facilities and guidance to work on this project and support throughout the period of this study. This course was a joint effort of IIT Roorkee and ER&DCI, Noida. My sincere thanks are also due to **Sh. R.K.Verma**, Executive Director and **Mr. V.N.Shukla**, Course Coordinator, M.Tech (IT), ER&DCI, Noida for their critical support and providing the necessary infrastructure and other facilities at every stage of this course.

I would like to thank **Mr. Rajiv Phougat**, Project Leader, C-DAC, Hauz Khas, New Delhi, my Advisor, for his constant support, incredible enthusiasm and encouragement throughout the course of this project. My sincere thanks are also due to **Mr. Sudershan**, C-DAC, Hauz Khas for providing valuable suggestions during the course of my project work. I am also grateful to **Mr. Munish Kumar**, my co-guide, Project Engineer, ER&DCI, Noida, for the cooperation extended by him, which led to the successful completion of this project report.

Most of all I would like to thank my mother and father who provided me a perfect environment for my studies and supported me throughout, and my brother, Anuj who was always there to encourage me. I would like to express my deep sense of gratitude to my Masi and her family, who gave me all the support and encouragement required during the course of this study. Last but not the least, I would like to thank my friends who have always been a constant source of encouragement for me. Finally, I would like to extend my gratitude to all those persons who directly or indirectly helped me in the process and contributed towards this work.


(Shilpa Kaushik)

Enrolment No. 019049

LIST OF FIGURES

Figure 1	Basic Visual Cryptography Technique	10
Figure 2	Encryption (2 out of 2 share)	12
Figure 3	Decryption (2 out of 2 share)	12
Figure 4	Flowchart for the Encryption process	16
Figure 5	Flowchart for the Decryption process	18
Figure 6	Screenshot of initial screen	20
Figure 7(a)	Test 1 - Sample encryption file (Black and White)	24
Figure 7(b)	Test 1 - After encrypting (1 st out of 2 shares)	25
Figure 7(c)	Test 1 - After encrypting (2 nd out of 2 shares)	25
Figure 7(d)	Test 1 - After decrypting from shares	26
Figure 8(a)	Test 2 - Sample encryption file (Grayscale)	27
Figure 8(b)	Test 2 - After encrypting (1 st out of 2 shares)	28
Figure 8(c)	Test 2 - After encrypting (2 nd out of 2 shares)	28
Figure 8(d)	Test 2 - After decrypting from shares	29
Figure 9(a)	Test 3 - Sample encryption file (Coloured)	30
Figure 9(b)	Test 3 - After encrypting (1 st out of 2 shares)	31
Figure 9(c)	Test 3 - After encrypting (2 nd out of 2 shares)	31
Figure 9(d)	Test 3 - After decrypting from shares	32
Figure 10	Test results of Encryption and Decryption	34

ABSTRACT

Security, in today's digital world, of stored data is of prime concern, as the data has to be prevented from falling into the hands of unscrupulous people. In view of the security threats, **cryptography**, the art and science of concealing information is a possible solution. There are various standard cryptography techniques that can be used for simple encryption and decryption of the data (usually text). The problem (defined in this project) consisted of concealing the information that is in the form of images, so that it is not possible to read the original image by unauthorized people, as these images are of highly confidential nature.

To solve the above stated problem a relatively new technique, called **visual cryptography**, was applied to encrypt and decrypt images. In this technique, the original image is divided into shares so that it is not possible to decipher the original image from the shares. The division is based on the fact that a pixel is the basic component that stores the information about an image and if the pixel value is changed, the image gets distorted. In this project, the original image is divided into shares such that the shares contain parts of information (pixel values) about the original image. The division is done based on a logical function and does not require any mathematical function to be applied to encrypt or decrypt the image.

This technique has an advantage over the conventional cryptography techniques, because it doesn't require any mathematical computations and is based on the division of pixels. Hence it is a faster, more efficient and highly secure cryptography technique for protecting images.

Table of Contents

CANDIDATE'S DECLARATION	i
ACKNOWLEDGEMENT	ii
LIST OF FIGURES	iii
ABSTRACT	iv
1. INTRODUCTION	1- 4
1.1 About C-DAC	1
1.2 About Project	1- 2
1.3 About Client	2- 3
1.4 Organization of Dissertation	3- 4
2. PROBLEM SCOPE AND ANALYSIS	5- 8
2.1 Problem Definition	5
2.2 Possible Solutions	5- 6
2.3 Selected Solution – Why?	6- 8
3. LITERATURE SURVEY	9-14
3.1 Visual Cryptography (Related Work)	9-14
4. DESIGN	15-20
5. CODING AND TESTING	21-32
5.1 Coding	21-23
5.2 Testing	23-32
6. IMPLEMENTATION	33-34
7. RESULTS AND DISCUSSION	35-36
7.1 Results	35
7.2 Discussion	35-36
8. CONCLUSION AND FUTURE WORK	37
REFERENCES	39
APPENDIX I	i -ix

INTRODUCTION

1.1 About C-DAC

C-DAC, Centre for Development of Advanced Computing, was setup in 1998 by the Scientific Society of Government of India [1].

Objective: The Design and Development of massively Parallel Supercomputing Technology. C-DAC recently unveiled PARAM Padma with peak processing power of 1 Terra Flops.

Strengths: Providing turn-key (end-to-end) solution through the emerging technologies like Real time, Telecom, Healthcare, Finance, E-Governance etc.

Mission and Objectives:

- Development of advanced computing technology based on parallel processing
- Setting up facilities for the proliferation for use of parallel processors
- Generalization of high quality man power for addressing needs of IT sector
- Addressing required of the IT industry in advanced computing areas
- Commercialization of spin-off technology developed indigenously

1.2 About the Client

The project at C-DAC aims at digitizing and automating the work of the IPO (Indian Patent Office)[1]. The IPO is an institution in India providing patents to various works and designs. The basic functions of the IPO are:

- To accept and process the request for Grant of Patent and registration of these Patents and Designs, along with the fee and also process refusals, renewals, cancellations, change of address, assignment of patents etc.
- Publish patents and designs lists
 - Search facility

- Classification (internal and external)
- Maintain statistics and records
 - Number of patents and designs
 - Status of new applications
 - Library

The project is a turn-key (end-to-end) solution for modernization and automation of IPO (Indian Patent Office) through phase wise computerization process. Various phase have been identified for the same as:

- Basic Training
- Procurement of hardware and system software
- Technical support and services
- Network implementation
- Digitization of existing records: The existing records need to be digitized for the computerization purpose.
- IPO website development
- ITSP (Information Technology Strategy Plan)
- Digitization and Processing of Patent requests

1.3 About the project

The project aims at provide security to the stored files, which are in the formats like TIFF (Tagged Image File Format). In the digitizing process (as will be seen in the coming sections), the old documents are to be digitized and this is done by scanning the documents. The scanned documents are basically in the form of images and these images are of highly confidential nature. This whole data has to be kept away from unscrupulous people who are not authenticated or legal users. Thus there is a need for a highly secure method of storing these images so that there is no unauthorized viewing of this data or images.

In this case the data being digitized is stored in the form of images. Therefore there is a need to provide a cryptography technique, which is best suited for the images. For this purpose the technique called **visual cryptography**, which is based on cryptography of

images, is used. The process of visual cryptography involves splitting the original image into shares. These shares are such that each share is different from the original image and seeing the shares it is not possible to decipher the original image. Thus the images can be stored in the encrypted form and decrypted only when an authorized user is accessing the data. During the decryption of these images the shares are combined to get back the original image.

1.4 Organization of Dissertation

The dissertation entitled “Enterprise system solution for C-DAC by providing storage level security using visual cryptography” has been organized in the following format:

The first chapter gives an introduction of C-DAC where the project was carried out and about the client for whom the project is being designed. The dissertation is part of a larger project, which is being developed at C-DAC. A brief introduction about the project is also given.

The second chapter, Problem Scope And Analysis, defines the problem and gives the possible solutions to the problem. After this the most appropriate solution selected is described.

The third chapter, Literature Survey, gives an idea of the related work already done in this field.

The fourth chapter, Design, gives the approach followed to carry out the project along with the design considerations that had to be kept in mind while developing the project.

The fifth chapter, Coding and Testing, gives the detail of the coding process and gives some test sample carried out on different types of images.

The sixth chapter, Implementation, tells where, when and how the project will be implemented. As we know the project is a small module of a larger project and will be implemented in the main project.

The seventh chapter, Results and Discussion, discusses about the performance of the project developed and also discusses the results on implementing the project.

The last chapter, Conclusion and Future work, concludes the whole dissertation and the scope of the project in the future.

In the end is given a list of references used during the course of study of this dissertation and then is the Appendix giving the details of the methods and functions that have been used in the code.

This is the whole compilation of the dissertation given in brief as a guide through the report.

PROBLEM SCOPE AND ANALYSIS

2.1 Problem Definition

C-DAC requires setting up a secure network expanding over a few cities. In this setup the data that has to be digitized is of highly confidential nature. Thus, there needs to be security at each level so that there is no leakage of information. Security features like authentication of the user were being taken care of by the already available authentication mechanisms like prompting for username and password. By this mechanism the user can be allowed access to information only when he/she has been authenticated. Hence the problem was defined as that of providing security at the storage level.

As the data is digitized and is in the form of images, there is a need to provide security to the stored files, having the file formats like TIFF (Tagged Image File Format). Thus, an efficient cryptography technique was to be developed so that at the time of storage the images could be encrypted and then stored such that it was not possible to decipher the original image from the encrypted form. Similarly, a decryption technique also needed to be developed, in accordance with the encryption technique, which could decrypt the encrypted images giving back the original image.

Therefore, the problem was defined as providing security at the storage level with the help of an efficient cryptography technique. It will be seen in the next section, what were the possible solutions and which one was the best suited and why?

2.2 Possible Solutions

Now that the problem has been defined, the possible solutions that are available are listed, and the best among the available solutions is chosen. The possible solutions are given in the next page:

- Firstly, the conventional encryption and decryption techniques are available. In this process the original message, referred to as 'plaintext', is converted into a random message known as 'ciphertext' [2]. The encryption process consists of an algorithm and a key, where the key value is independent of the plaintext. Based on this key the algorithm gives an output, which is in the form of a random message not understandable to anyone. Using the same key the decryption algorithm is applied to the ciphertext to get back the original message. Some of these algorithms involve various mathematical functions in which calculations are carried out to generate the ciphertext. Some of these standard techniques are DES, RC5, and IDEA *etc.*
- Secondly, a relatively new technique known as **Visual Cryptography** [3] is there, which, as the name suggests, is used for images. In this technique the original image is divided into shares and these shares are relatively different from the original image. The shares are basically the division of the pixels so as to distort the original image. This is the encryption process. The decryption process involves combining these shares to get the original image. This technique does not involve any mathematical computations or functions, as we need to work with the pixels and these shares themselves act as keys.

Among these available solutions the most optimal solution was chosen as best suited for our needs and implemented.

2.3 Selected Solution – Why?

From the given possible solutions Visual Cryptography was chosen as the best suited for this problem. The main advantages of visual cryptography due to which it was chosen over the other conventional encryption algorithms are listed below:

- The problem was to deal with images and as the name suggests, visual cryptography deals with the cryptography of images. The basic concept of visual cryptography is to hide the images in such a form, which the human eye cannot decode. The problem was to encrypt the images in a form that they are not deciphered by the human visual system, unless decrypted. Therefore this method

was preferred over the conventional encryption algorithms, which deal basically with text data.

- Visual cryptography can be defined as a cryptographic scheme in which the images can be encoded and decoded without any cryptographic computations [3]. The standard algorithms require a lot of mathematical functions to be performed to carry out the necessary encryption. As compared to these, visual cryptography doesn't require any mathematical computation to be carried out as it is totally based on the division of pixels. Thus by applying simple logical functions we can divide the pixels and get the required results.

In the next chapter a study of the already available literature on Visual Cryptography is carried out and the visual cryptography process is described in detail.

LITERATURE SURVEY

As defined in the problem definition phase, the main concern was that of security of the digitized images at the storage level. Among the conventional techniques of cryptography already available and the new techniques coming up, Visual Cryptography was chosen as the best possible solution. The technique of visual cryptography is a relatively new technique and very less work has been done in this field. Research is still going on and it is under constant development unlike the other cryptography techniques, which have set standard methods to be applied.

As discussed earlier the standard cryptography techniques use mathematical functions to encrypt and decrypt information. Visual Cryptography is simply based on the division of pixels to shares. This technique has been specifically designed for images and is highly secure and reliable and above all very easy to implement. Papers have been published on visual cryptography and based on these papers the study for this project has been carried out. The process of visual cryptography is described in the next section.

3.1 Visual Cryptography (Related Work)

What is visual cryptography?

The problem of encrypting written material (printed text, hand-written notes, pictures, etc.) in a perfectly secure way, which can be decoded directly by the human visual system, is considered [3].

The basic model consists of a printed page of ciphertext (which can be sent by mail or faxed) and a printed transparency (which serves as a secret key). Placing the transparency with the key over the page with the ciphertext reveals the original cleartext, even though each one of them is indistinguishable from random noise. The system is similar to a one-time pad in the sense that each page of ciphertext is decrypted with a different transparency.

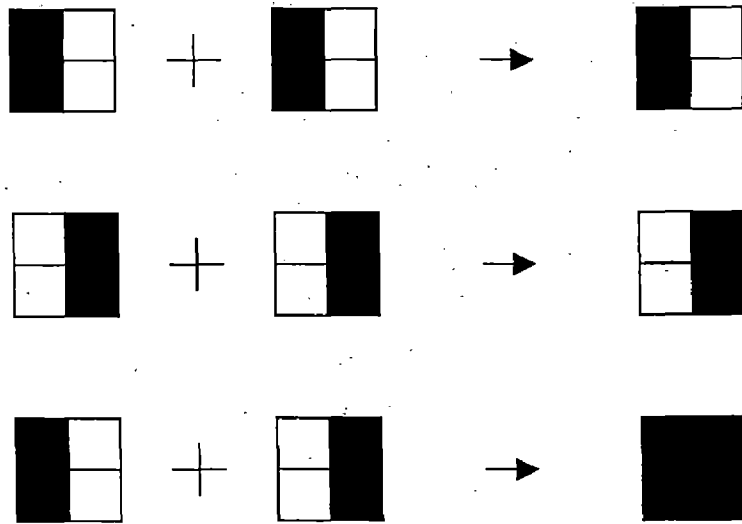


Figure 1. Basic Visual Cryptography Technique

Due to its simplicity, the system can be used by anyone without any knowledge of cryptography and without performing any cryptographic computations.

Visual Cryptography Scheme can be thought of as a private key cryptosystem. The secret printed message is encoded into two random looking shares [4]. One of the two shares will be a printed page of ciphertext which can be sent by mail or fax, whereas the other share serves as the secret key. Stacking together the two transparencies reveals the original image. This can be seen in the work by Moni Noar and Adi Shamir [4].

In its simplest form it is assumed that the images are composed of black and white pixels. Each share is a collection of m black and white subpixels. Hence we can divide the image into black and white transparencies that are formed from the subpixels. These transparencies can then be combined together to get the original image. The breaking and the combining can be as simple as the XOR gate in which the similar bits give a '0' bit and dissimilar bits give a '1' bit whereas in this case the similar images can give the similar output whereas the dissimilar part gives the output based on the input image shares.

Figure 1 shows the basic visual cryptography technique where the image is considered to be composed of just black and white pixels. The original image can be divided into a number of shares (≥ 2) depending on the level and type of security required. The figure shows the cryptography technique known as 2 out of 2 secret sharing scheme in which the original image is divided into two shares. Generally speaking the shares can be n out of k ($n \leq k$) which means that the image can be divided into k shares such that we get the original image only if 'n' or more than 'n' of the shares are combined. Major work has been done in this field by M. Noar and A. Shamir [3][4].

How is it useful to us?

The basic fundamental of visual cryptography revolves around the fact that images are composed of pixels, which can be broken down into parts or subpixels called shares thus distorting the image. This scheme can be implemented in the project to provide for the security of the digitized data. The image can be broken into shares and store these shares separately. These shares are in the encrypted form now as it is not possible to decipher the original image from the shares, when viewed separately.

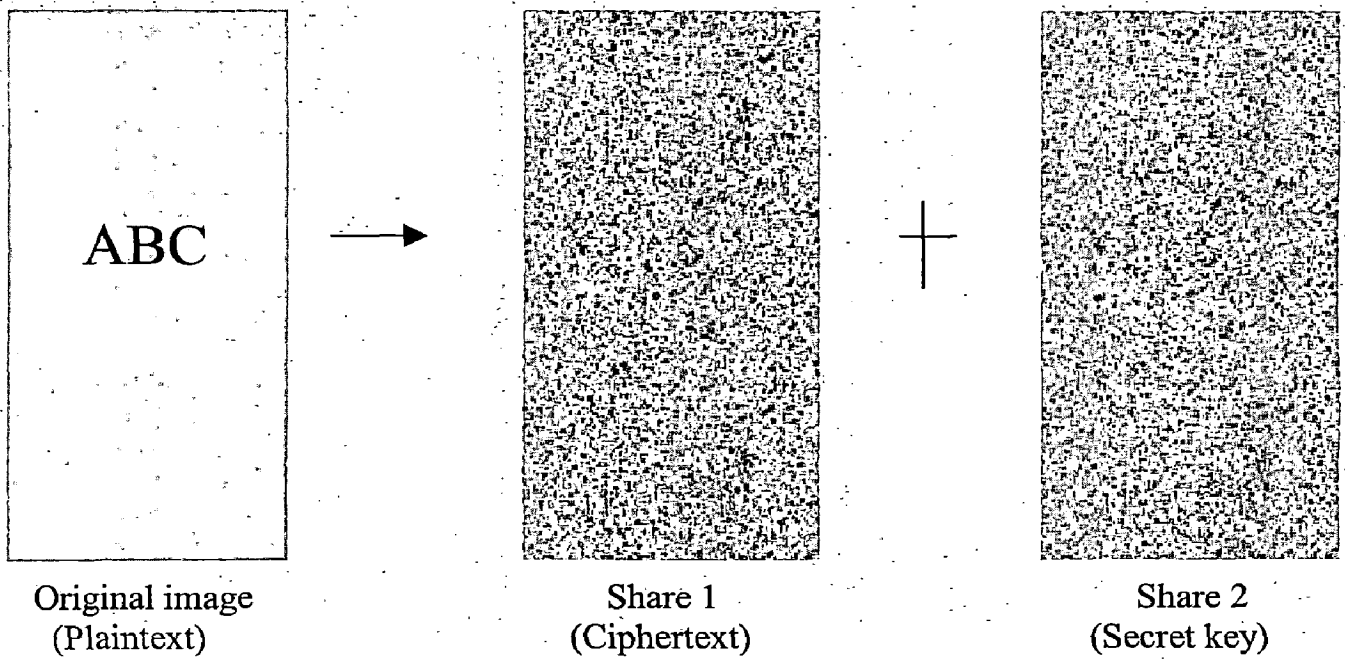


Figure 2. Encryption (2 out of 2)

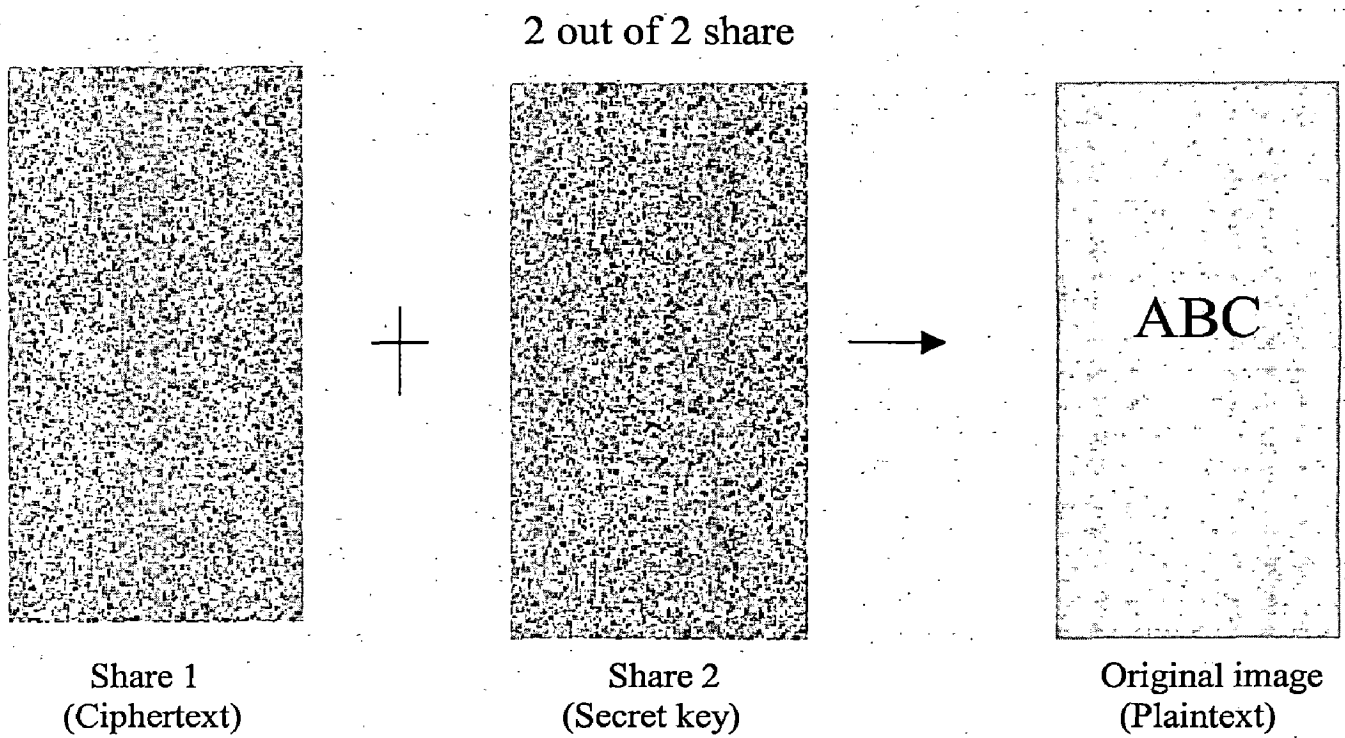


Figure 3. Decryption (2 out of 2)

Decryption is required to combine the shares. So, whenever a person tries to access these images the authenticity of the user is first established. Once it has been established the shares are combined based on an algorithm that further depends on the encryption algorithm used.

The user is asked for the username and password, provided at the time of registration, when he/she tries to access the images. Once the authenticity of the user is established the required image demanded by the user is decrypted and then provided to the user in the original form. Hence, it is not possible for any unauthorized person to view the data even if he/she has hold of all the shares of the image.

The fundamental used in the encryption and decryption techniques is that the image is composed of pixels and a particular number of bits are assigned to store a pixel. They are varying in number. To carry out the encryption process the image is divided by splitting the pixels to different shares. The shares are formed such that some of the subpixels are in one share and some in the other share. The subpixels are such that it contains part of the pixel value. As an example, if a pixel is made up of 8 bits then it can be divides as 4 LSB's in one share and the 4 MSB's in the other share.

Thus, the complexity of the algorithm can be increased or decreased based on how the shares are divided. At the time of decryption there is a need to combine the subpixels to form the original pixel with the help of an algorithm based on the algorithm used to encrypt the image.

Consider an example in which the plaintext 'ABC' is the printed text as shown in figure 2. It is required to encrypt the image and divide it into two shares so that one share acts as the ciphertext and the other as the secret key. As a result two random images are formed from the original image and none of these shares resembles the original share. Dividing the pixel to subpixels forms these shares. Seeing the two images it is not possible to decipher the original image. To form the original image back again it is needed to combine the two images based on the algorithm used to encrypt the image.

In figure 3 it can be seen that combining the shares forms the original image. The decryption algorithm used to decrypt the images is based on the encryption algorithm used. We combine the 2 out of 2 share to form the original image. When the two shares are again combined with the help of the decryption algorithm based on the encryption algorithm

used, it is possible to get the original image back. Thus, it is seen that the encryption and decryption process does not involve any mathematical calculations like the standard techniques making it a faster cryptography mechanism for images. This can be seen in the paper presented by M. Noar and B.Pinkas [10].

The technique used in this project is slightly different from the techniques stated above. The above techniques rely on the visual perception power of the human eye. The process defines a plaintext, which is divided into ciphertext and a secret key (transparencies). When the secret key, which is in the form of a transparency, is placed on the ciphertext it is possible to get back the original image. In this project the existing model was modified. As the requirement was to apply the cryptography technique to digitally stored images, it was not possible to divide these digital images to physical transparencies. Therefore, it was needed to divide the images digitally i.e. divide the image pixels into different images such that the divided images can be referred to as the secret key and the ciphertext.

In the visual cryptography technique implemented the original image file (in TIFF format) is divided into two or more different image files. The divided images are such that the pixel information in the original image is divided into these images based on a simple logical function. The divided images are also in the same format as the original image and contain the same header; only the pixel information is different in each divided file. Looking at these files separately it is not possible to decipher the original image. This is the encryption technique. Similarly, in the decryption technique the different parts of an image are combined together in one file based on the logic used for encryption. As all the files have the same header the original image file can be combined back in the same format after combining the pixel information.

DESIGN

Now that the problem as well as the solution have been defined, the next step in the software development is the designing. The steps are designed that need to be carried out to develop the project. The design is one of the most important phases of a project development life cycle as a good design leads to the development of a good project. In this process the sequence of steps is determined to develop the project. Along with this the other considerations are also laid down that have to be kept in mind while designing the project.

The first step in this process is to design the necessary algorithm, which describes the sequence of steps to be followed steps to encrypt and decrypt a file. The algorithm gives an idea of how the program has to be developed. After designing the algorithm it is needed to develop the flowchart based on these steps for both the processes. The flowchart determines the flow of the process that we have to be followed. Apart from the flowchart certain points have to be kept in mind, which are necessary in the development process, like the type of file being handled, the format of the file, etc.

The algorithm for the encryption process is described as follows.

Algorithm for encryption:

- Ask for the file to be encrypted from the user
- Open the said file and other files (shares)
- Scan header and write to the shares
- Apply logical function to the pixel values
- Continue the above step till the end of input file
- Close all files

The flowchart for this is shown in figure 4.

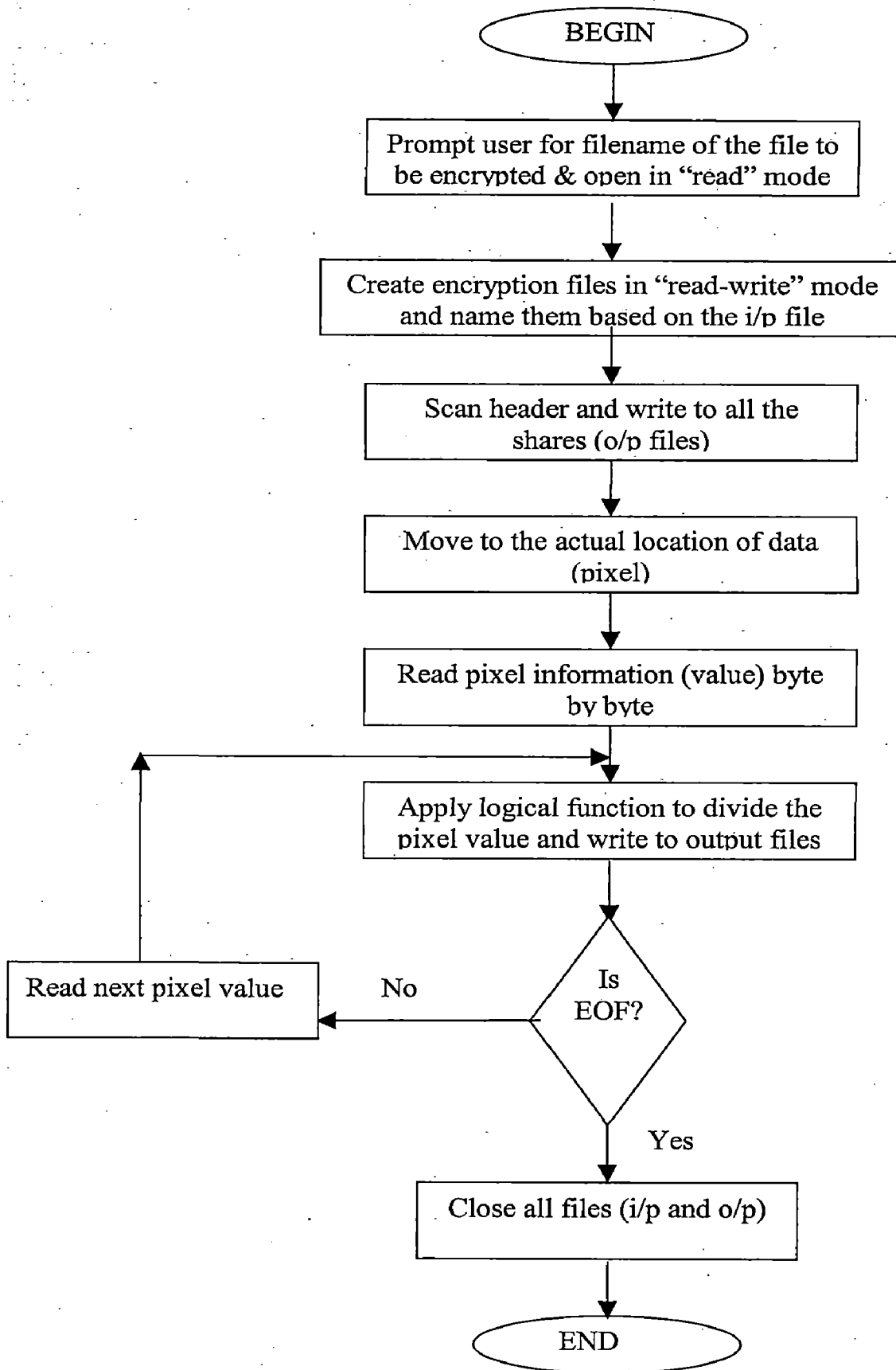


Figure 4 Flowchart for the Encryption process

Algorithm for decryption:

- Ask for the file to be decrypted from the user (the original file)
- Open the related files (shares) based on the filename
- Open the original filename (decryption file)
- Scan header from the shares and write to the decryption file
- Apply reverse logical function to the pixel values from the shares
- Continue the above step till the end of input files
- Close all files

The flowchart is shown in **figure 5**.

Some design considerations have to be kept in mind, apart from the basic flowchart, while developing the project. These are:

- **The file format:** The files or the images that are being considered here are in the **TIFF (Tagged Image File Format)**[7]. Thus, there is a need to encrypt the file such that the file encrypted is also in the TIFF format and at the same time different from the original image file. And for this the study of the tiff file format is necessary. TIFF is an image file format. A file is defined to be a sequence of 8-bit bytes, where the bytes are numbered from 0 to N. The largest possible TIFF file is 2^{32} bytes in length.

A TIFF file begins with an 8-byte “image file header” that points to an “image file directory (IFD)”. An image file directory contains information about the image, as well as pointers to the actual image data. Thus, the pixel values are not in serial order but located randomly in the file and pointers pointing to the location give the address of values in the file. Thus, to locate the actual data there is a need to move in the file according to the pointers.

- **The header and the tag values** have to be handled carefully so as not to alter their value while encrypting. Only the pixel values need to be altered and not any other information that can change the meaning of the original image. The same header and tag values have to be present in all the shares so that they could be reconstituted to the original image, while decrypting, with ease and without any discrepancy in the image.

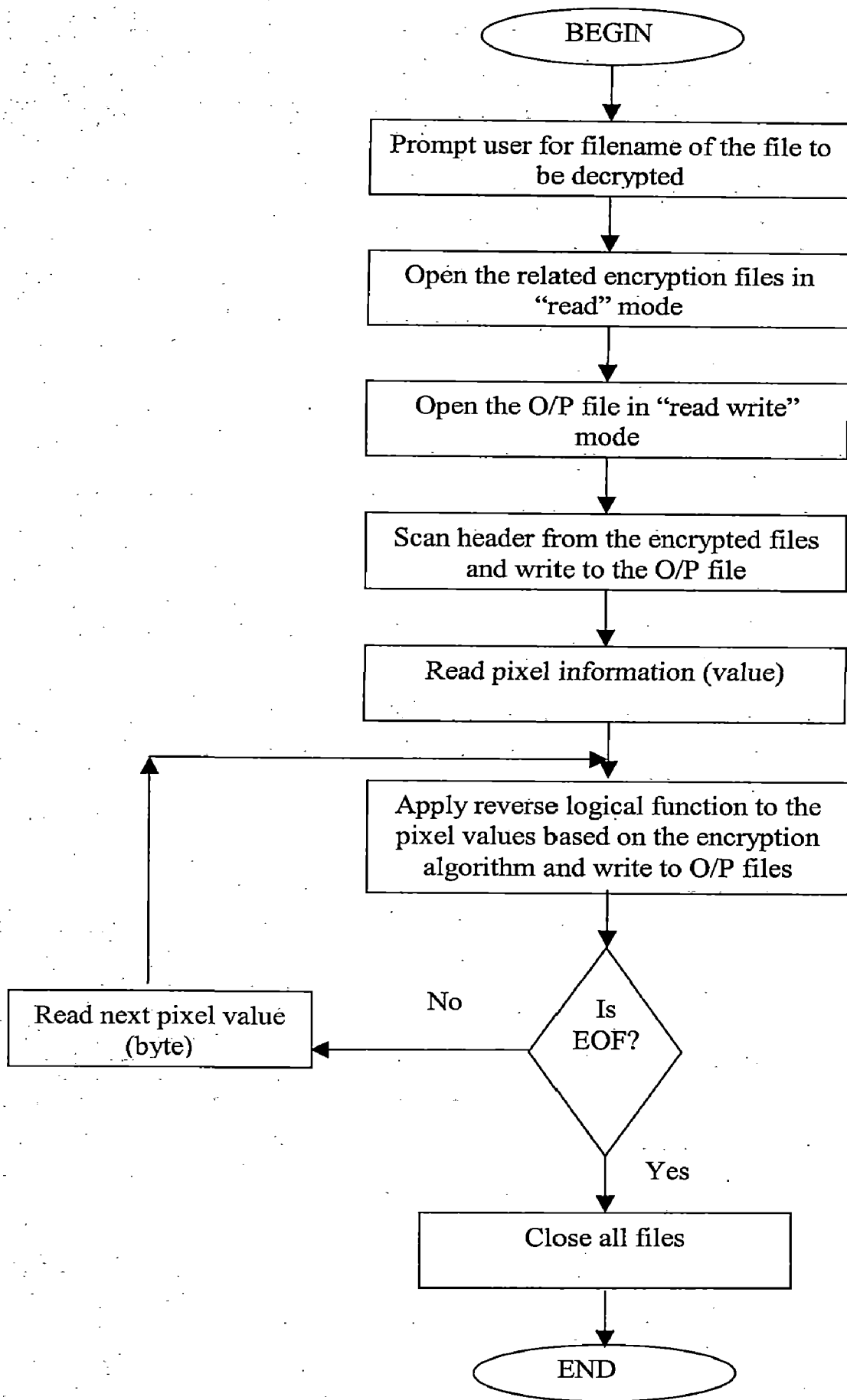


Figure 5 Flowchart for the Decryption process

- **The type of file:** The files that have to be handled may be of the type Black and White or Grayscale or Coloured. Thus, depending on the type of file the encryption and decryption process has to be carried out. Because for each type of file the number of bits being used to depict a pixel value are different and so processing of each file is different. For example the black & white file format require only 2 bits per sample whereas for the gray scale, the value varies from 2- 8 bits per sample and for the coloured files the bits per pixel range from 8–24 bits.
- **The number of shares:** Once the type of file has been fixed a decision has to be taken on the number of shares of the original file to be made. The number of shares depends on the file type because the bits per pixel are different for each type. Like, in case of coloured files if the image is divided into 2 shares it will not be of much use as the distortion will be very less. Similarly, in case of black and white files the image, if divided into two shares gives a fully distorted image thus solving our purpose. The decryption thus depends on the bits per sample of the file.
- **Division of pixels:** The pixels have to be divided in such a manner that the shares so formed do not tell or tell the minimum about the original image. A logical function has to be defined that can provide the maximum distortion to the given image as the whole complexity of the system depends on this function. By doing so it will be possible to achieve a highly efficient mechanism to encrypt and decrypt the images.
- **The file names:** In the visual cryptography technique the image is divided into shares *i.e.* files (in this case). These shares have to be different (different file names) as well as distinguishable so that they can be separate from each other and at the same time they are similar to the original file so that it is easy to decrypt the files using the file names. Thus, the file names were so designed that they were different from each other and also distinguishable from the other shares. The shares were differentiated by giving them the names of the files followed by four digits ‘_en1’ and ‘_en2’ and then the extension of the file. Thus, it was easy to distinguish the shares of one file from that of the others as they had the name of the original file and at the same time the number or the index differentiated the different shares of the same original image.

- **Hiding file structure from user:** The user does not have to know the file structure of the files being processed. The role of the user is limited to give the name of the file, which is to be encrypted/decrypted. The file naming mechanism is restricted only to the knowledge of the program and not to the user. The user just has to enter the name of the file to be processed and then tell whether to encrypt or decrypt the file and for this purpose a simple user interface has been designed. The interface is shown in figure 6.

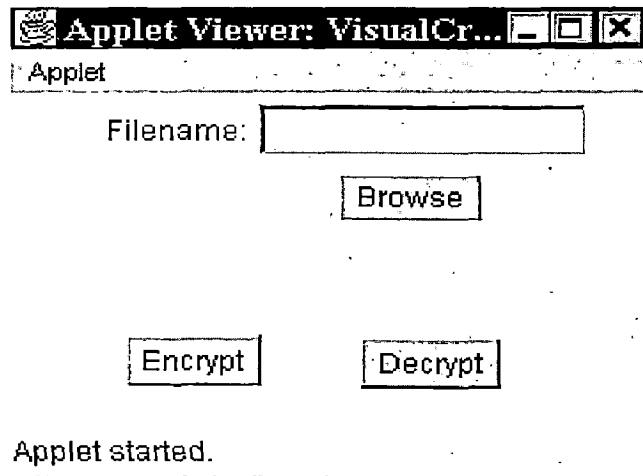


Figure 6 Screenshot of the user interface

These were the basic design considerations for the development of the project, to be kept in mind before the coding. Based on these design considerations the code is developed.

CODING AND TESTING

5.1 Coding

After the design phase comes the coding part, where the above-described design is implemented into the actual code. The main project, of which this project is a part, is being developed with JAVA [9] as the language. Hence, the code was written using JAVA. Java is an Object oriented programming language most suited for web-based applications. As the project of IPO is a web-based project hence Java was used for the development of the project. Java has got many features, which make it one of the most popular programming languages in use today. Some of them are stated below:

- Simple
- Robust
- Object Oriented
- Multi-threaded
- Architecture Neutral (hardware independent)
- Encapsulation
- Multilevel Inheritance
- Portable
- Dynamic

The Java programming language is unusual in that a program is both compiled and interpreted [9]. With the compiler, first you translate a program into an intermediate language called Java byte-codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte-code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed.

Java has various packages, which is a collection of related classes and interfaces providing access protection and namespace management. The classes and interfaces that

are part of the Java platform are members of various packages that bundle classes by function. The classes define the variables and the methods common to all objects of a certain kind. Some of the java packages used in this program are described below [11]:

- **java.io.*:** This package contains a collection of stream classes that support these algorithms for reading and writing by defining the input and output streams in the program. In java we have the byte stream (read 8 bit data) as well as the character stream (read 16 bit data). A program can send information to an external destination by opening a stream to a destination and writing the information out sequentially and then reading back the information by an input stream.
- **java.lang.*:** This package is one of the fundamental packages available in java. This package provides classes to that can access functions related to the basic data types like characters, strings, integers, etc. The basic data types can be treated as objects and then manipulated by applying functions.
- **javax.swing.*:** This package is used to design the graphical user interface for the program. This package comprises of components like frames, panels, buttons, etc. which help in designing a good user interface. As compared to the java.awt package the javax.swing package is more flexible any easy to use.

Now the source code for the problem can be easily written as the flowchart and all the design considerations that have to be kept in mind while writing the code are known.

The codes for the encryption as well as the decryption process are written. The program for encryption named encrypt, encrypts the TIFF image. Initially it prompts the user for the name of the file to be encrypted. When the filename has been entered, shares are created i.e. files are opened in read mode following the naming criteria. The header is copied to all the files. Then the tags are copied to the shares and the locations of the real data read. It has to be noted at this point that in a TIFF file the actual image data is stored at random in the file and is not in the sequential manner. Thus, a record of all the locations of the image data has to be kept. The image data is read byte by byte and the logical function applied to each byte, dividing the image into shares.

Now for the decryption process the program, named decrypt, performs the job. The user has to enter the name of the file to be decrypted. Once this has been entered the name

of the shares are determined and the corresponding files opened. A new file is opened and the corresponding header from the shares is written to the new file followed by the tags and their corresponding values. When the actual pixel data is read the reverse logical function is applied to the bytes read from the corresponding shares and the written to the new file. This new file is the decrypted file or the original image that was encrypted.

All the complexity of the code is hidden from the user and role of the user is just limited to entering the name of the file and specifying whether the file has to be encrypted or decrypted.

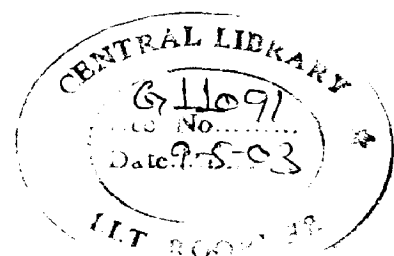
5.2 Testing

Once the coding has been done the next important step is to test the code for various set of inputs. Testing is a very important phase in the design of a program, as there is a need to check whether the code is giving the desired output or not. To test a given code a set of inputs is given to the program and then analyzing the output it is seen if the output is as expected or not. If it is the desired output then the code has been successfully designed, otherwise the code is not successful and changes have to be made to get the desired results.

Thus, for our code we carried out the tests giving different types of inputs. These inputs consisted of TIFF images of different formats. Three different types of files were given to the program as test data and the results recorded. The files were in different formats i.e.

- Black and White (Test 1)
- Gray scale files (Test 2)
- Coloured files (Test 3)

The next few pages shows a sample file and the results obtained after encrypting them and then decrypting the encrypted shares.



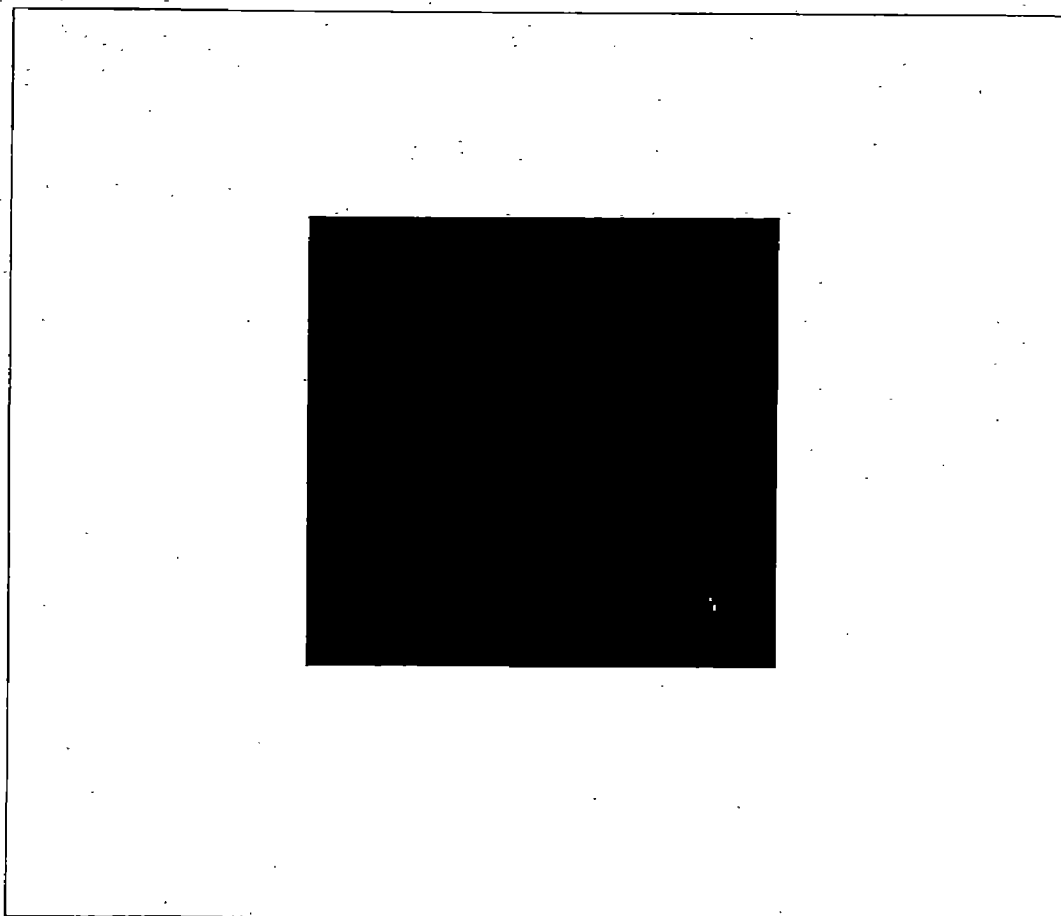


Figure 7(a) Test 1 - Sample encryption file (Black and White)

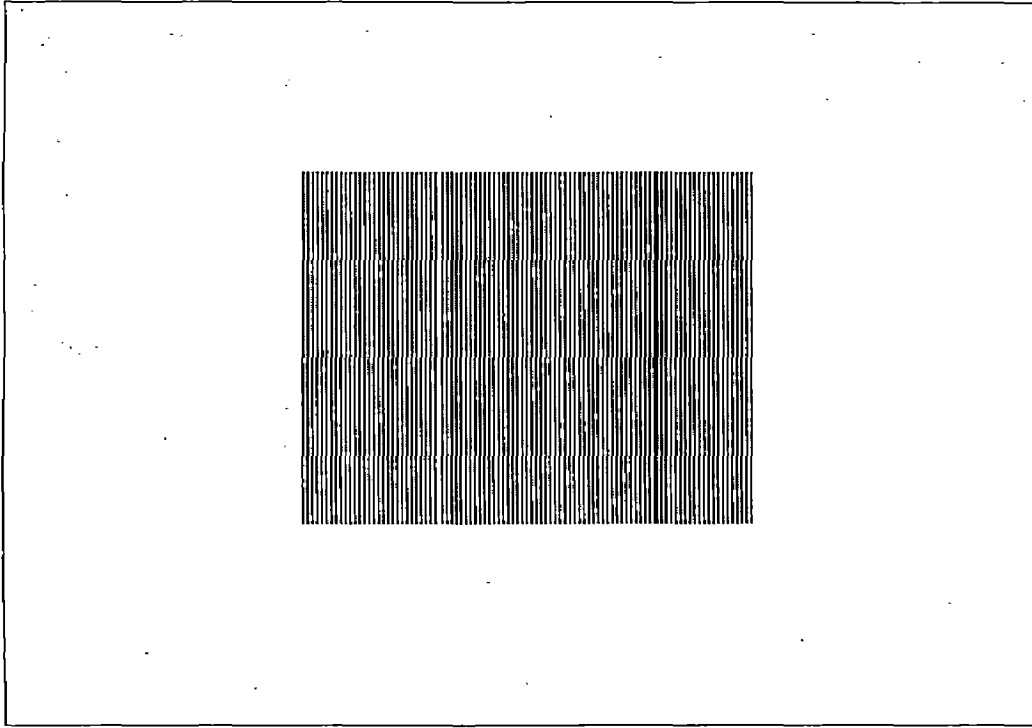


Figure 7(b) Test 1-After encrypting (1st out of 2 shares)

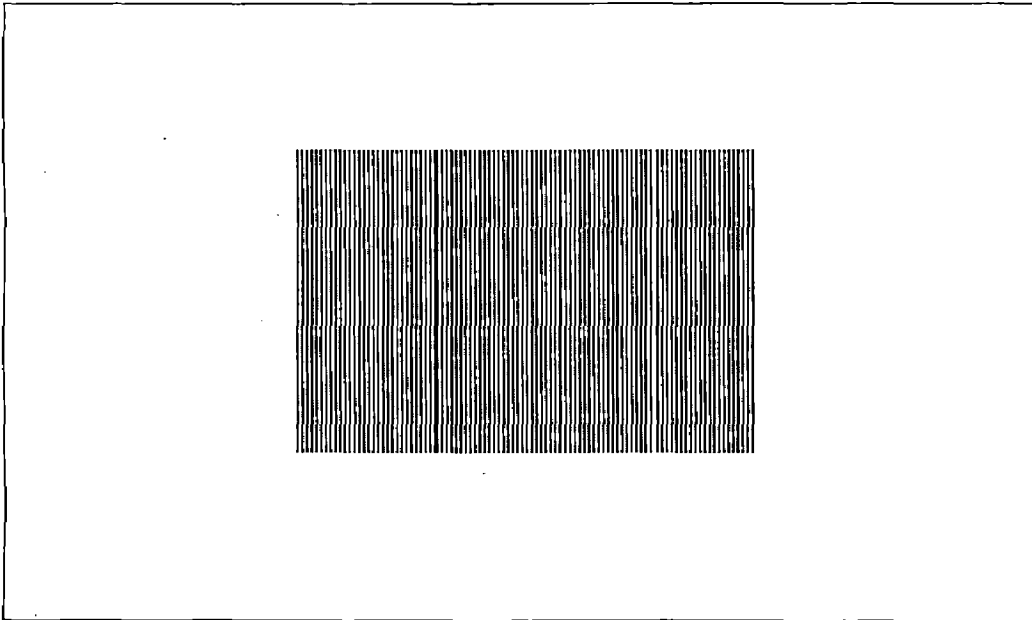


Figure 7(c) Test 1 - After encrypting (2nd out of 2 shares)

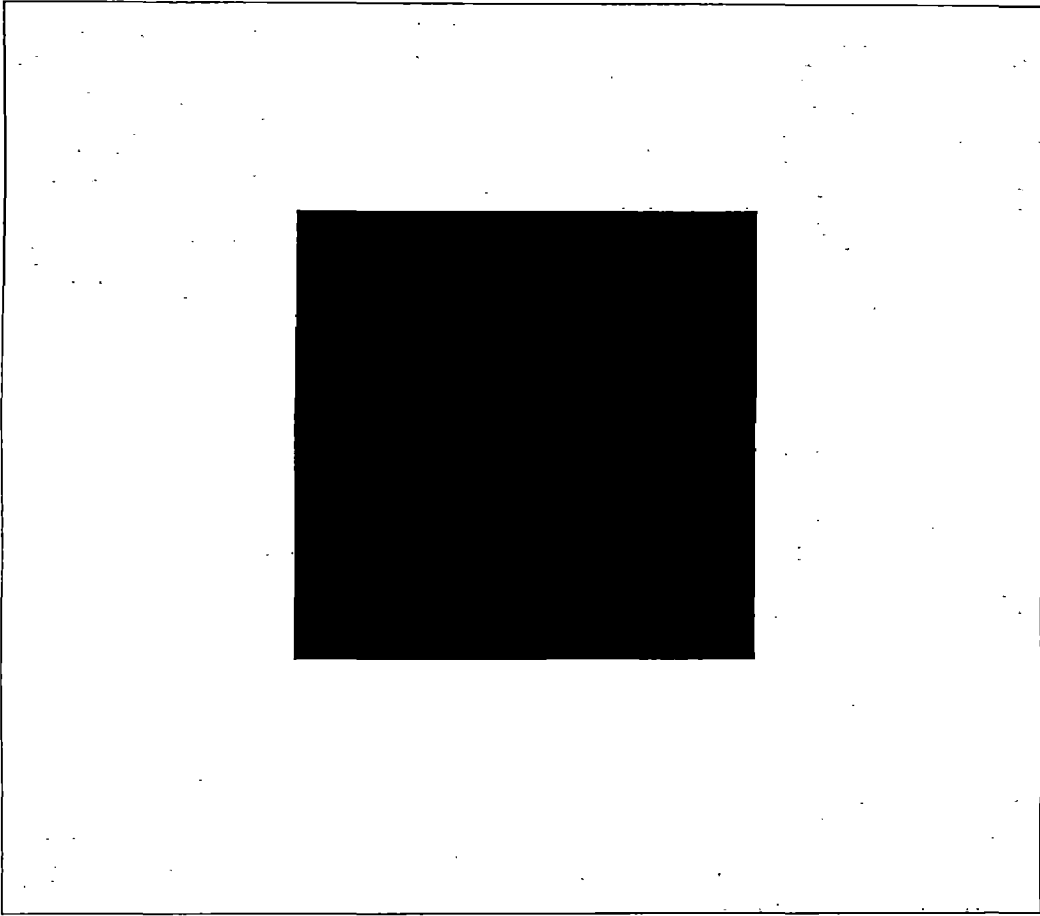


Figure 7(d) Test 1 - After decrypting from shares

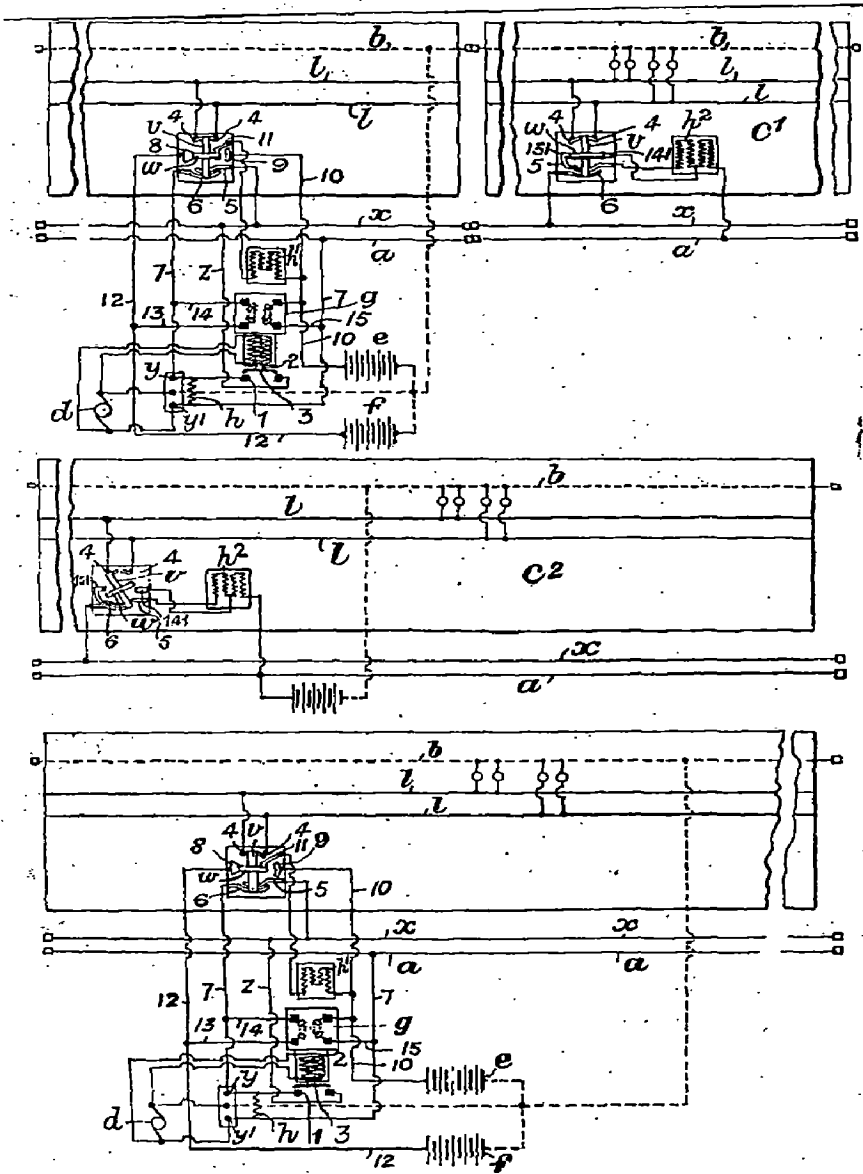


Figure 8(a) Test 2 - Sample encryption file (Grayscale)

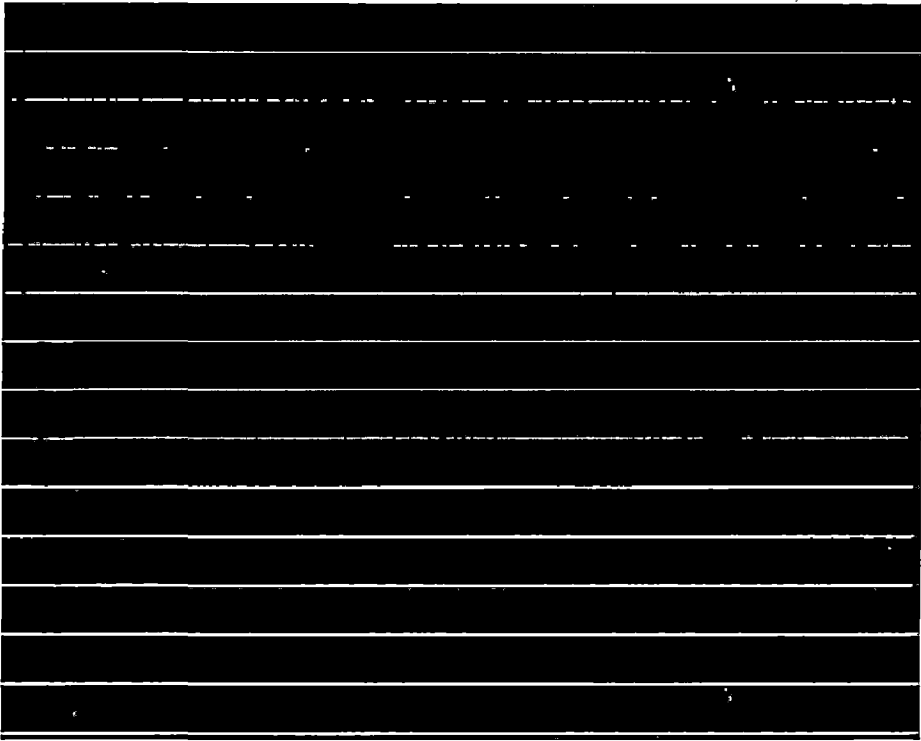


Figure 8(b) Test 2 - After encrypting (1st out of 2 shares)



Figure 8(c) Test 2 - After encrypting (2nd out of 2 shares)

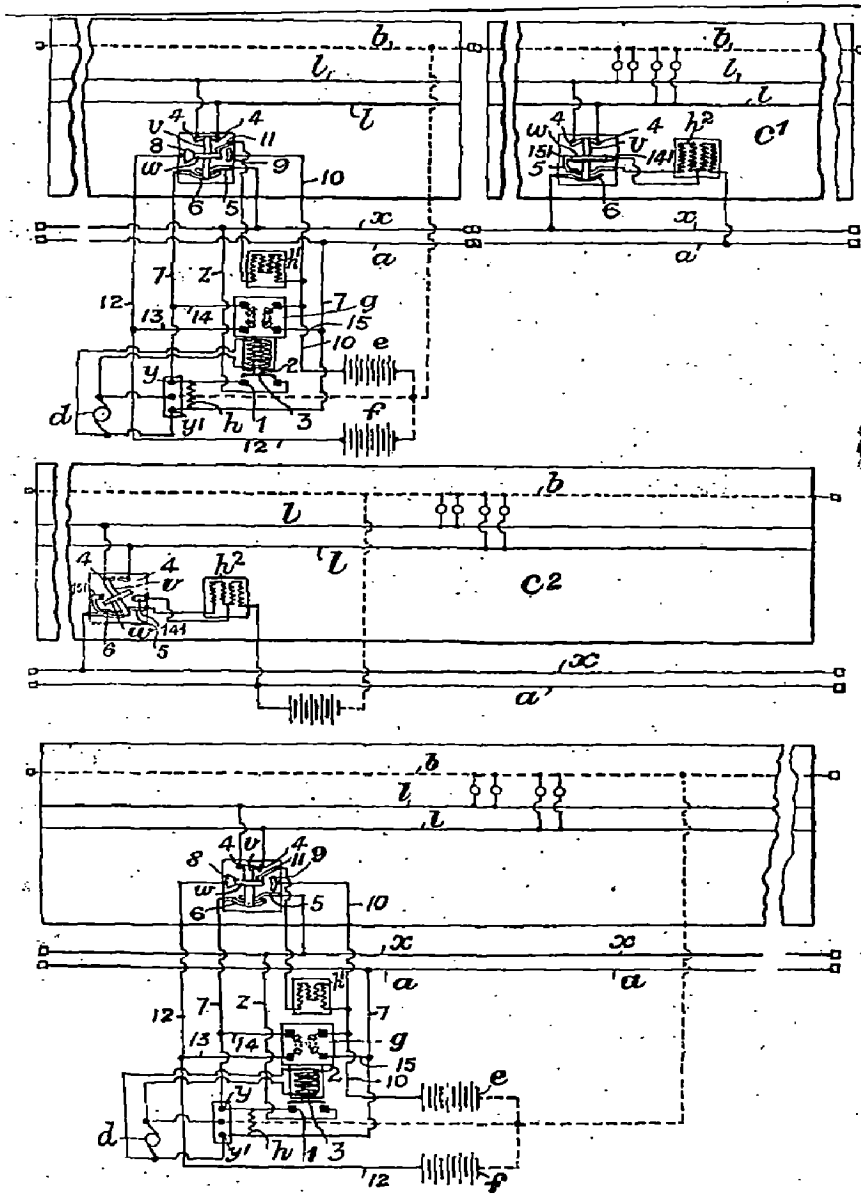


Figure 8(d) Test 2 - After decrypting from shares



Figure 9(a) Test 3 - Sample encryption file (Coloured)

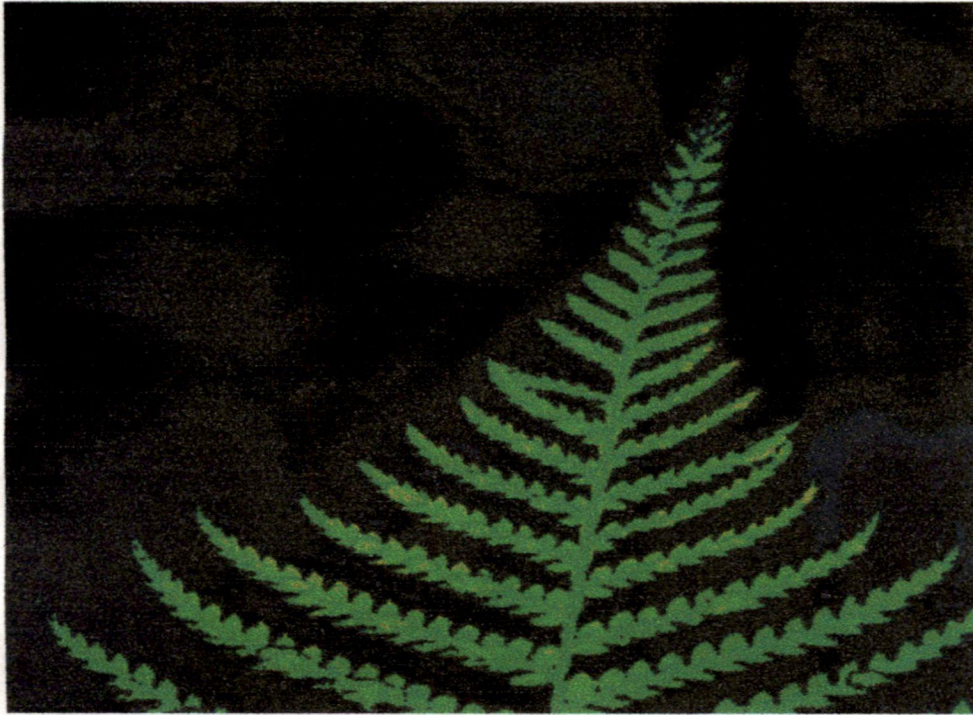


Figure 9(b) Test 3 - After encrypting (1st out of 2 shares)



Figure 9(c) Test 3 - After encrypting (2nd out of 2 shares)



Figure 9(d) Test 3 - After decrypting from shares

IMPLEMENTATION

After the development of the code, it has to be implemented to actually start the working. Implementation is one of the important steps in the development of software. Till now the works described can be called the development steps in the software life cycle. Once the software has been developed and thoroughly tested it has to be implemented in the actual environment in which it has to be used. In this case the project is a part of a larger project, which is still being implemented. This project aims at providing storage level security to the digitized images.

Once the whole project has been completed this module will be attached to the major project as a security feature in the project to safeguard the images, which are of highly confidential nature. This module will be attached at the storage level where the images are stored. Whenever a user requests for a file, the user will be authenticated and then the name of the file to be encrypted or decrypted will be asked. The program will perform the rest of the job of encryption and decryption. Thus, the stored files will be encrypted and then stored in the system so that even if someone has access to the database of these files he/she is not able to decipher the original image, as they will be in the encrypted form. Only the authenticated users will have access to the original files, as the decryption of the requested file will take place when the user has authenticated itself.

Type of File	Size of File	Time taken for encryption	Time taken for decryption	Result
Black & White (Test 1)	25KB	1.7 sec	1.5 sec	Successful
Grayscale (Test 2)	29KB	1.8 sec	1.6 sec	Successful
Coloured (Test 3)	31KB	2 sec	1.8 sec	Not Successful

Figure 10 Test results of Encryption and Decryption

RESULTS AND DISCUSSION

7.1 Results

The tests were carried out on different types of TIFF files ranging from black and white, grayscale to coloured. The visual cryptography scheme that has been implemented is the 2 out of 2 share. This means that the image is first encrypted into 2 shares and the decryption takes place with the help of these two shares. The original image cannot be formed until and unless both the shares are combined using the decryption algorithm. Thus, for each of the test case two shares were made when the image was encrypted and during the decryption process these two shares were combined to get back the original image.

The results that were obtained can be summarized as in the figure 10. It can be observed from these results that the 2 out of 2 share scheme for visual cryptography works best for the images, which require lesser number of bits per pixel. In the above tests the Black and White and Grayscale images were encrypted with maximum distortion whereas the Coloured image was decrypted with the least distortion. Hence, a different scheme has to be implemented for the coloured images.

7.2 Discussion

Based on the tests that were carried out for a number of test cases it can be said that the visual cryptography technique is a very efficient encryption and decryption mechanism, well suited for images. Also, as there are no mathematical functions to be calculated the speed of encryption and decryption is also very fast as compared to the conventional cryptography techniques.

It can also be said that the complexity of the technique depends on how the shares have been divided. The logical condition used has to be such that it divided the image into n distinct shares. Moreover the complexity of the technique also depends on the type of file being encrypted. The same division may be the best for one type of file (eg. Black & white)

and may not be at all good for the some other type of file (eg. colour file). Thus, to divide the shares, knowledge of the type of file associated and the bits being used to interpret the pixel value is required. For example, in case of black and white TIFF files only 1 bit per pixel is required, whereas in case of grayscale TIFF files 2 to 8 bits are required per pixel and up to 24 bits per pixel for coloured files (also known as the photometric interpretation).

As can be seen from the test results obtained, the logical function used in the code is best suited for black and white and gray scale files. This happens because for these two type of files the maximum number of bits required for each pixel ranges from 1-2 bits for black and white and 2-8 bits for grayscale files. Whereas in case of coloured files more bits per sample are needed and hence, the division of shares is not effective. The coloured files have to be divided into more number of shares thus, giving a tradeoff for the space as all the shares occupy the same space.

Therefore deciding the number of shares depends on the file type and specifically the number of bits per sample for each pixel in the image file. The time taken for encrypting the files is the same for all the files of same size and only depends on the size of the file and not the type, the larger the file the more the time taken. Moreover, there is also a tradeoff of space as the number of bits per sample increase. This is so because the resolution of the image requires more bits and to distort the image more shares will be needed to divide the information pertaining to each colour into small bits. This leads to more space requirement as each share will be of the same size and more number of shares means more space.

Thus, this technique is well suited for black and white and grayscale images and can also be applied for coloured images but with slight modifications.

CONCLUSION AND FUTURE WORK

In the end it can be concluded that the visual cryptography technique is a highly secure, fast, efficient and easy way to keep our images secure. As it is based on the power of the human eye to interpret the pixels, by distorting the image a very efficient means to hide the original image from unscrupulous people can be provided. By increasing the distortion the system can be made more secure. Moreover, the visual cryptography scheme is very fast and easy to implement, as it does not require any mathematical calculations to be done as can be seen in the case of conventional cryptography systems where mathematical computations take most of the time. Even if the user has hold of the shares he/she cannot decipher the original image from these, as he will need the decryption algorithm for this purpose.

As this field is a relatively new one there is tremendous scope of work in this field. The project can be extended to work equally well for all types of TIFF files. It can also be extended to work for other image file formats apart from TIFF. And this technique of visual cryptography can be used as security measure in any case dealing with images.

By adding certain compression techniques and other features to this system it can be made more efficient and secure and can also implement in the places where the image transfers take place by encrypting the image at the senders end and then decrypting it at the receivers end. This technique can be highly beneficial in the Internet when the transfer of highly confidential images is required. To send the image, it can be encrypted and the shares sent separately and at the receiving end the shares can be combined by using the decryption algorithm.

REFERENCES

1. IPO User Specification Document, C-DAC, Hauz Khas, New Delhi.
2. William Stallings, "Cryptography and Network Security", PHI Publication, 2000.
3. Moni Noar and Adi Shamir, "Visual Cryptography". Department of Applied Math and Computer Science, Weizmann Institute, Rchovot, Israel, 1995.
4. Moni Noar and Adi Shamir, Visual Cryptography II: Improving The Contrast Via The Cover Base, 1996.
5. Giuseppe Ateniese, Carlo Blundo, Alfredo De Santis, and Douglas R. Stinson, Visual Cryptography for General Access Structures, Department of Computer Science and Engineering and Center for Communication and Information Science University of Nebraska-Lincoln, Lincoln NE 68588, USA,1996.
6. Noar Shamir and Benny Pinkas, Visual Authentication and Identification, 1994.
7. TIFF Revision 6.0, Aldus Developers Desk, Aldus Corporation, Seattle, WA.
8. <http://www.cl.cam.ac.uk/~fms27/vck/>
9. <http://www.sun.java.com/>
10. M. Noar and B.Pinkas, "Visual Authentication and Identification", Advances in cryptology-CRYPTO-89 proceedings, 1989.
11. Patrick Naughton and Herbert Schildt. JAVA 2 The Complete Reference, Tata McGraw Hill Edition, 1999.

APPENDIX 1

Details of the Java packages and the methods used from those packages, is given below:

java.io package:

This package contains a collection of stream classes that support these algorithms for reading and writing by defining the input and output streams in the program. In java we have the byte stream (read 8 bit data) as well as the character stream (read 16 bit data). A program can send information to an external destination by opening a stream to a destination and writing the information out sequentially and then reading back the information by an input stream. The details of the methods of this package used are:

- **BufferedReader** extends Reader:

Read text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines.

Constructors:

- `public BufferedReader(Reader in, int sz)`

Create a buffering character-input stream that uses an input buffer of the specified size.

Parameters:

in - A Reader

sz - Input-buffer size

Throws:

IllegalArgumentException - If sz is ≤ 0

- `public BufferedReader(Reader in)`

Create a buffering character-input stream that uses a default-sized input buffer.

Parameters:

in - A Reader

Methods used:

- `public String readLine()`

throws `IOException`

Read a line of text. A line is considered to be terminated by any one of a line feed (`'\n'`), a carriage return (`'\r'`), or a carriage return followed immediately by a linefeed.

Returns:

A `String` containing the contents of the line, not including any line-termination characters, or null if the end of the stream has been reached.

Throws:

`IOException` - If an I/O error occurs

- `InputStreamReader` extends `Reader`

An `InputStreamReader` is a bridge from byte streams to character streams: It reads bytes and decodes them into characters using a specified charset. The charset that it uses may be specified by name or may be given explicitly, or the platform's default charset may be accepted.

Constructors:

- `public InputStreamReader(InputStream in)`

Create an `InputStreamReader` that uses the default charset.

Parameters:

`in` - An `InputStream`

- `File` extends `Object` implements `Serializable`, `Comparable`

An abstract representation of file and directory pathnames. User interfaces and operating systems use system-dependent pathname strings to name files and directories. This class presents an abstract, system-independent view of hierarchical pathnames.

Constructors:

- `public File(String pathname)`

Creates a new File instance by converting the given pathname string into an abstract pathname. If the given string is the empty string, then the result is the empty abstract pathname.

Parameters:

pathname - A pathname string

Throws:

NullPointerException - If the pathname argument is null.

Methods used:

- `public void close()`

throws IOException

Close the stream.

Specified by:

close in class Reader

Throws:

IOException - If an I/O error occurs

- **RandomAccessFile** extends Object implements DataOutput, DataInput

Instances of this class support both reading and writing to a random access file. A random access file behaves like a large array of bytes stored in the file system. There is a kind of cursor, or index into the implied array, called the *file pointer*; input operations read bytes starting at the file pointer and advance the file pointer past the bytes read. If the random access file is created in read/write mode, then output operations are also available; output operations write bytes starting at the file pointer and advance the file pointer past the bytes written. Output operations that write past the current end of the implied array cause the array to be

The file pointer can be read by the `getFilePointer` method and set by the `seek` method.

Constructors:

- `public RandomAccessFile(String name,String mode)`

throws `FileNotFoundException`

Creates a random access file stream to read from, and optionally to write to, a file with the specified name. A new `FileDescriptor` object is created to represent the connection to the file. The mode argument specifies the access mode with which the file is to be opened. The permitted values and their meanings are as specified for the `RandomAccessFile(File,String)` constructor. If there is a security manager, its `checkRead` method is called with the name argument as its argument to see if read access to the file is allowed. If the mode allows writing, the security manager's `checkWrite` method is also called with the name argument as its argument to see if write access to the file is allowed.

Parameters:

name - the system-dependent filename

mode - the access mode

Throws:

`IllegalArgumentException` - if the mode argument is not equal to one of "r", "rw", "rws", or "rwd"

`FileNotFoundException` - if the file exists but is a directory rather than a regular file, or cannot be opened or created for any other reason

`SecurityException` - if a security manager exists and its `checkRead` method denies read access to the file or the mode is "rw" and the security manager's `checkWrite` method denies write access to the file

Methods used:

- `public final int readUnsignedByte()`

throws `IOException`

Reads an unsigned eight-bit number from this file. This method reads a byte from this file, starting at the current file pointer, and returns that byte. This method blocks until the byte is read, the end of the stream is detected, or an exception is thrown.

Specified by:

`readUnsignedByte` in interface `DataInput`

Returns:

the next byte of this file, interpreted as an unsigned eight-bit number.

Throws:

`EOFException` - if this file has reached the end.

`IOException` - if an I/O error occurs.

- `public final void writeByte(int v)`

throws `IOException`

Writes a byte to the file as a one-byte value. The write starts at the current position of the file pointer.

Specified by:

`writeByte` in interface `DataOutput`

Parameters:

`v` - a byte value to be written.

Throws:

`IOException` - if an I/O error occurs.

- `public void seek(long pos)`

throws `IOException`

Sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs. The offset may be set

not change the file length. The file length will change only by writing after the offset has been set beyond the end of the file.

Parameters:

pos - the offset position, measured in bytes from the beginning of the file, at which to set the file pointer.

Throws:

IOException - if pos is less than 0 or if an I/O error occurs.

java.lang package:

This package is one of the fundamental packages available in java. This package provides classes to that can access functions related to the basic data types like characters, strings, integers, etc. We can treat the basic data types as objects and then manipulate those objects by applying functions to them.

- public final class String extends Object

implements Serializable, Comparable, CharSequence

The String class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class. Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. The class String includes methods for examining individual characters of the sequence, for comparing strings, for searching strings, for extracting substrings, and for creating a copy of a string with all characters translated to uppercase or to lowercase.

Constructors:

- public String()

Initializes a newly created String object so that it represents an empty character sequence. Note that use of this constructor is unnecessary since Strings are immutable.

- **public String(char[] value)**

Allocates a new String so that it represents the sequence of characters currently contained in the character array argument. The contents of the character array are copied; subsequent modification of the character array does not affect the newly created string.

Parameters:

value - the initial value of the string.

Throws:

NullPointerException - if value is null.

- **public String(char[] value, int offset, int count)**

Allocates a new String that contains characters from a subarray of the character array argument. The offset argument is the index of the first character of the subarray and the count argument specifies the length of the subarray. The contents of the subarray are copied; subsequent modification of the character array does not affect the newly created string.

Parameters:

value - array that is the source of characters.

offset - the initial offset.

count - the length.

Throws:

IndexOutOfBoundsException - if the offset and count arguments index characters outside the bounds of the value array.

NullPointerException - if value is null.

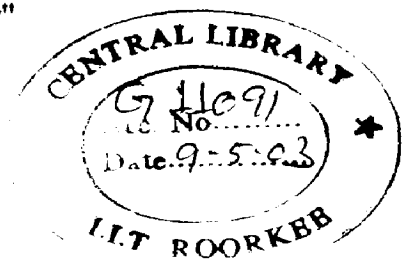
Methods used:

- **public String substring(int beginIndex)**

Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.

Examples:

"unhappy".substring(2) returns "happy"



Examples:

"unhappy".substring(2) returns "happy"

Parameters:

beginIndex - the beginning index, inclusive.

Returns:

the specified substring.

Throws:

IndexOutOfBoundsException - if beginIndex is negative or larger than the length of this String object.

○ public String substring(int beginIndex, int endIndex)

Returns a new string that is a substring of this string. The substring begins at the specified beginIndex and extends to the character at index endIndex - 1. Thus the length of the substring is endIndex - beginIndex.

Examples:

"hamburger".substring(4, 8) returns "urge"

"smiles".substring(1, 5) returns "mile"

Parameters:

beginIndex - the beginning index, inclusive.

endIndex - the ending index, exclusive.

Returns:

the specified substring.

Throws:

IndexOutOfBoundsException - if the beginIndex is negative, or endIndex is larger than the length of this String object, or beginIndex is larger than endIndex.

○ public String concat(String str)

Concatenates the specified string to the end of this string.

If the length of the argument string is 0, then this String object is returned. Otherwise, a new String object is created, representing a

sequence represented by this String object and the character sequence represented by the argument string.

Examples:

`"cares".concat("s")` returns `"caress"`

`"to".concat("get").concat("her")` returns `"together"`

Parameters:

`str` - the String that is concatenated to the end of this String.

Returns:

a string that represents the concatenation of this object's characters followed by the string argument's characters.

Throws:

`NullPointerException` - if `str` is null.