CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Developed software can be deployed for any organization, Home applications or for secure confidential chatting system. The requirements can be modified as per need, i.e. user may want to chat or transfer a file over the system. Here we a developing the software as an application which can be used at the server or receiver end for encoding and retrieving the data. Here user is provided with a number of options that can be used.

Basic Needs of the Project are:

1. Users: Persons who will use this application:

    1.1 User1: who will encode and hide the data before transmission

    1.2 User2: who will decode the data and retrieve that from image.

2. Software:

    2.1 JDK tool kit: Java tools for the development of the project code.

    2.2 Java Swings: for generating a graphical user interface and make project more user friendly.

    2.3 AWT: Abstract window toolkit

3. Language:

    3.1 Java: to generate the code and make an application code that can run on itself like other softwares.

4. Operating System: Windows xp and windows vista

1.2 SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

1. **Introduction**: Our software will allow the user to transmit the data after hiding that into an image. The data to be transmitted will not just be hidden in image but also it will be encrypted by the password. This password will also act as key for encryption. All sort of security and various levels of compression can also be done with the user Requirements.

    1.1 **Purpose**: To make the entire System over the network secure. Security is provided with encryption and through obscurity. Encryption algorithm used is DES(Data Encryption Standards). For obscurity DATA CONCEALMENT is used.

    1.2 **Scope**: this project has no limitation of windows or software as the complied code provided can be used on any system. The system will just be loaded with the java tools. Interface designed will make the user more friendly with the system.

    1.3 **References**: In order to use various software required to make this project Internet or particular books on Java Swings and DATA CONCEALMENT must be used. The programmer must have a knowledge of Java Source coding, java Swings and Security features available in java. Various books related to these topics have been used to make the project

    1.4 **Overview**: The SRS contains the details of types of users, data that is to be encrypted, file that need to be encrypted and revealing all the data. Compression of data is also done that is to be understood by the programmer.

2. **Overall Description**: Project is not affected by the memory requirement because very Simple software used to make this project like Java. Java uses the command line to execute the code.

2.1 **Product Perspective**: It is a complete real time software. It can be used as a part of large system or it can also be used with in home or LAN network.

2.1.1 **System Interface**: Users will interact with the help of the Graphical user interface provided by the software. As Java is platform independent and based on just a Basic command line it can be loaded on any Operating System. Hence, no specific software requirement to use it.

2.1.2 **Interfaces**: Each page opened must have a link to cancel and change the selected option. No specific logins are provided so can be used anywhere.

2.1.3 **Hardware Interfaces**: there are no specific hardware requirements for the project.

2.1.4 **Software Interfaces**: In order to make this project various software are required. Jdk: create and execute java applications.

2.1.5 **Communication Interface**: No specific communication interface required for this project to execute because of platform independence of Java.

2.1.6 **Memory Constraints**: RAM memory should be large enough to execute the java code.

2.1.7 **Operations**: The project will work in four types of conditions:

2.1.7.1 Embed message.
2.1.7.2 Embed file
2.1.7.3 Retrieve message
2.1.7.4 Retrieve file

Apart from this Encryption will be allowed. Compression can be done at a scale of 0-9 so that high and low compression cold be permitted.

2.2 **Product Functions**: the software will perform following function:

2.2.1 Encryption of message

2.2.2 Encryption of file

2.2.3 Compression of message

2.2.4 Compression of file

2.2.5 Encryption and compression of message/ File

2.2.6 Embedding all above into an image.

2.3 **User Characteristics**: Any one can access the Software as no user authentication is being done. It is assumed that person who is using the system is authorized to use the software.

2.4 **Assumptions and Dependencies**: we have assumed that a person know how use a computer system. User may want to end at any time. Person sing software is already authorized if used in organisation. As key is same for encryption and decryption so key is exchanged through some secured channel

3. **Specific Requirements**:

3.1 **External Interface**: There are no specific external interface requirements to build/use this project just the basic needs of softwares and hardware are required.

3.2 **Functions**: Apart from the functions provided to users the developer must take care of the inputs provided by the user like appropriate length of the password.

3.3 **Performance requirement**: for the better performance of the system no specific requirements are there.

3.4 **Software System Attributes:** These are characteristics the system must possess, but that pervade the design. These requirements have to be testable just like the functional requirements**.**

3.4.1 **Reliability**: The inconsistency must be removed completely so that the outputs must be completely reliable. Also functioning should be detailed with the current details of the databases so that Output too are consistent.

3.4.2 **Availability**: Software should be available to everyone using it because it is based on their structure. No Specific modifications are required in the software.

3.4.3 **Maintainability**: Software maintenance will require testing of system after some intervals. These systems must be tested.

3.4.4 **Portability**: Software must be portable to all the systems because of the language used. Secondly no specific h/w or s/w requirement will make it more easily usable.

1.3 LANGUAGE USED (JAVA)

Java is a powerful, cross-platform, object-oriented programming language suitable for writing anything from a distributed application that runs on a corporate network to a database-driven Web site to host your personal photo gallery. To make it easier to learn, the Java language was designed to resemble some of the most popular programming languages in use today, most notably C/C++.

Advantages and Disadvantages of using JAVA:

Java has the following advantages for the development of software in an academic environment:

- Tailored for the Internet.
- Platform Independent
- Interactive Content
- Computation Aspects
- Java Applications Run Locally
- Security

Some disadvantages of using Java include the following:

- Programming Knowledge
- Performance Concerns
- Available Code

1.4 SECURITY IN JAVA

The security features provided by the Java Development Kit (JDK™) are intended for a variety of audiences:

- **Users running programs**:

  Built-in security functionality protects you from malevolent programs (including viruses), maintains the privacy of your files and information about you, and authenticates the identity of each code provider. You can subject applications and applets to security controls when you need to.

- **Developers**:

  You can use API methods to incorporate security functionality into your programs, including cryptography services and security checks. The API framework enables you to define and integrate your own permissions (controlling access to specific resources), cryptography service implementations, security manager implementations, and policy implementations. In addition, classes are provided for management of your public/private key pairs and public key certificates from people you trust.

- **Systems administrators, developers, and users**:

  JDK tools manage your keystore (database of keys and certificates); generate digital signatures for JAR files, and verify the authenticity of such signatures and the integrity of the signed contents; and create and modify the policy files that define your installation's security policy.

Security functions help you to protect your programs and data from harm, to keep data shielded and private, and to deploy new applications in security-aware runtime environments.

Java also provides several tools to help you to manage access to system resources; to create, store and maintain encrypted public and private passwords (key pairs) and certificates; and to create and sign jar files during the deployment process.

CHAPTER 2

CONCEPT MODULES USED

2.1 DATA CONCEALMENT

DATA CONCEALMENT is a familiar idea to readers of detective and spy novels. It involves the hiding of a secret message inside an innocent-looking package or container (often called a carrier). For example, a micro-dot hidden beneath a postage stamp, or a message written in milk on the back of a letter, or instructions tattooed under a person's hair.

The cracking of steganographic messages is called steganalysis, and comes in two main forms. The easiest type of cracking simply makes the hidden message unreadable by modifying the carrier. This can usually be achieved by cropping or blurring the image, or saving it in a different file format. A much harder task is the extraction of the hidden message, which typically starts with the identification of telltale regularities or patterns in the carrier, or spotting differences between the carrier and its original.

2.1.1 LSB ALGORITHM

The simplest form of digital DATA CONCEALMENT (and probably the most common) is the Least Significant Bit (LSB) method, where the binary representation of the data that's to be hidden is written into the LSB of the bytes of the carrier. The overall change to the image is so minor that it can't be seen by the human eye.

The least significant bits have the useful property of changing rapidly if the number changes even slightly. For example, if 1 (binary 00000001) is added to 3 (binary 00000011), the result will be 4 (binary 00000100) and three of the least significant bits will change (011 to 100). By contrast, the three most significant bits stay unchanged(000to000).Least significant bits are frequently employed in pseudorandom number generators, hash functions and checksums.

LSB, in all capitals, can also stand for "Least Significant Byte". The meaning is parallel to the above: it is the byte (or octet) in that position of a multi-byte number which has the least potential value in steganography a message might be hidden or encrypt with in an image by changing least significant bit to be the message bits then the image can be transmitted through network.LSB based steganography is perhaps the most simple and straight forward approach. in this will only affect each pixel by +\-1, if at all ,it is generally assumed with good reason that degradation caused by this embedding process would perceptually transparent. Hence there are a number of LSB based steganography techniques in the passive warden model as it difficult to differentiate cover-image from stego images ,given the small changes that have been made.

Input Text	**‘g’ ‘a’ ‘u’ ‘r’ ‘a’ ‘v’**
	0 1 1 0 0 1 1 1
Image	1 0 1 0 1 0 1 1 (Original Pixel)
Msg>>>bit	_ _ _ _ _ _ _ _0 + 1
New Image	1 0 1 0 1 0 1 0 (New Pixel)

Figure 1.0

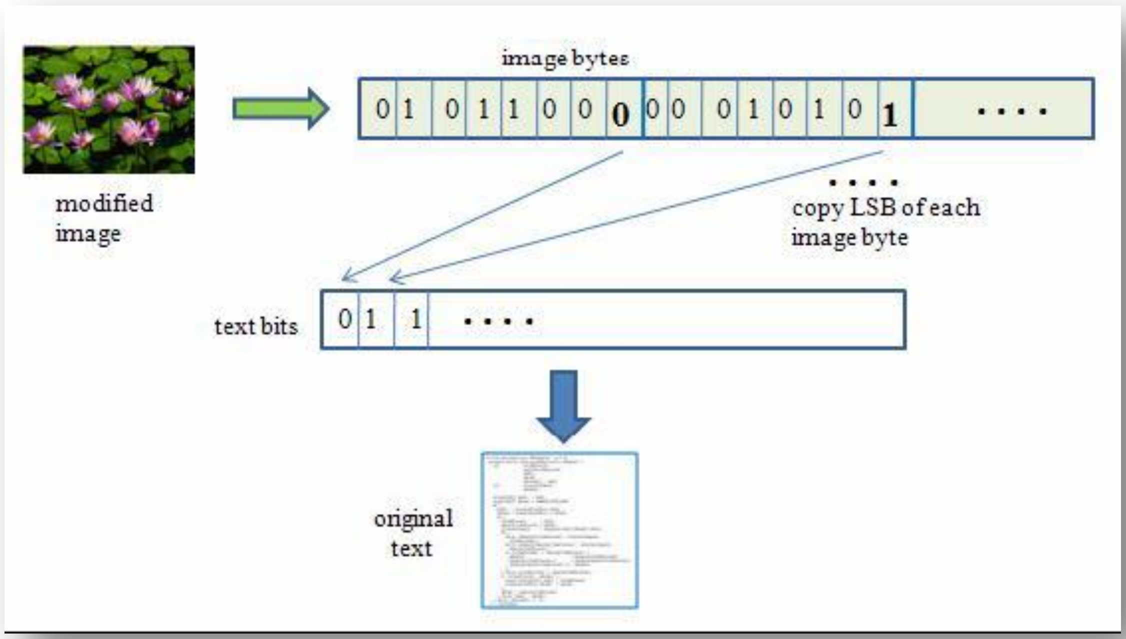


Figure 1.1

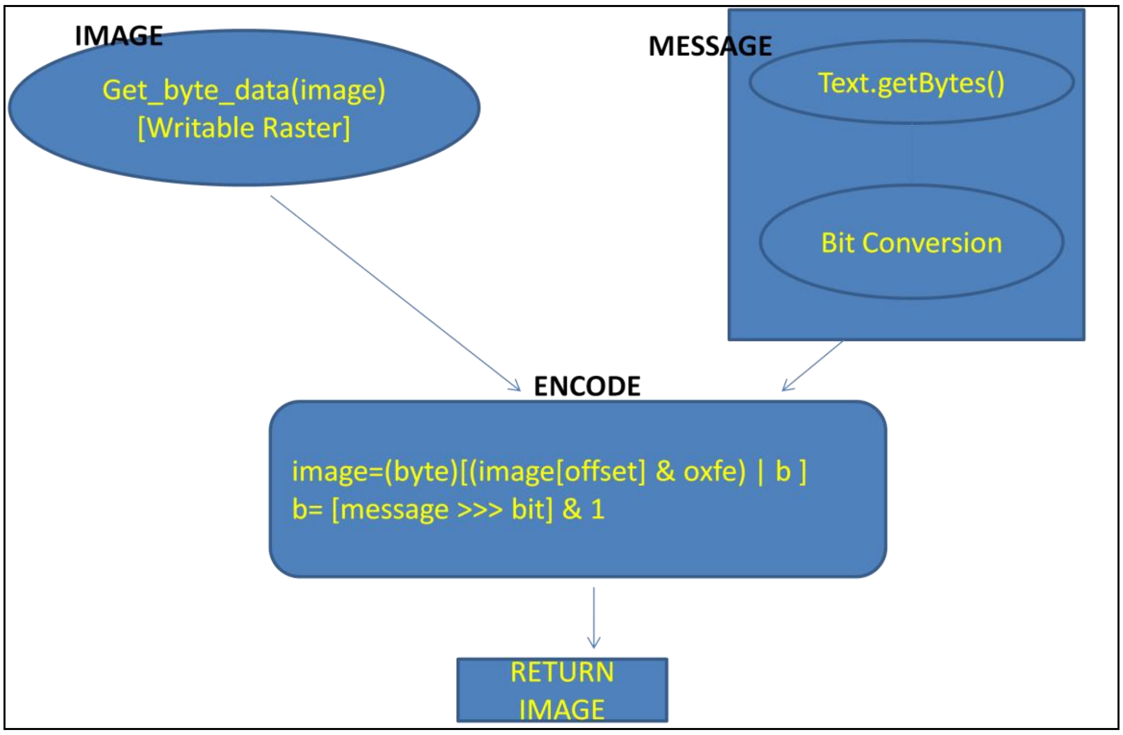


Figure 1.2

2.2 ENCRYPTION

Encryption is a way of scrambling and modifying our message so as to make it intelligible. Encryption can be simple or complex like Caeser cipher in which detection of message is easy to very complex like AES where the key used for encryption is so long that the cost of determining the data is very much.

DES: In 1972, National Institute of Standards and Technology decided that a strong encryption algorithm is required to protect non-classified information. That algorithm should be cheap, widely available and very secure. In 1974, IBM submitted Lucifer algorithm which meet most of the requirements. After that it was again modified and termed as DES.

DES encrypts and decrypts the data in64 bit blocks, using a 64-bit key. It takes 64 bit block of plain text as input and outputs 64 bit cipher text. Since it uses blocks of equal sizes and it uses both permutation and substitution in the algorithm, DES is both block cipher and product cipher.
DES has 16 rounds, meaning the main algorithm is repeated 16 times to produce the cipher text. It has been found that no. of rounds is exponentially proportional to the amount of time required to find a key using a brute force attack. So as the no. of rounds increases the security of algorithm increases. Although the key input is 64 bit but only 56 bits are used. The least significant bit in each byte is parity bit and should be set so that there will always be odd no. of 1's. these parity bits are ignored, so only the seven most significant bits f each byte is used, resulting in key length of 56 bits.

Steps to generate key for each round are:

1. Permutation of the input block.
2. Split it into 2 28 bits left and right.
3. Rotate left and right by the same no. of bits as specified.
4. Join left and right to get a new key.
5. Apply second permute to get final key.
6. Repeat the process to get 16 sub keys.

Various Steps in Encryption are:

1. Initial permutation is performed on the plain text.
2. Then using S-box substitution 32 bit data is extended to make it 48 bit.
3. 48 bit data is XORed with key and stored in a temporary buffer.
4. Again an Sbox substitution is performed on the data.
5. Permutation is performed on the data not obtained in the temporary buffer.
6. It is XORed with the left side 32 bits.

Actually DES is a fiestel Structure where the data is divided into left and right parts.

The left part remains as such and right hand side is modified and XORed with the left. After every Round the left and right hand side blocks are swapped. Xor operation, S boxes and Permutations are reversible hence original data can be recovered using the same key.

Key used is same for encryption and decryption. The security of this technique is based on the complexity to recover original data if key is not known. Time it takes for brute force attack is very large.

2.3 COMPRESSION

Data compression, **source coding** or **bit-rate reduction** is the process of encoding information using fewer bits than the original representation would use.

Compression is useful because it helps reduce the consumption of expensive resources, such as hard disk space or transmission bandwidth. On the downside, compressed data must be decompressed to be used, and this extra processing may be detrimental to some applications. For instance, a compression scheme for video may require expensive hardware for the video to be decompressed fast enough to be viewed as it is being decompressed (the option of decompressing the video in full before watching it may be inconvenient, and requires storage space for the decompressed video). The design of data compression schemes therefore involves trade-offs among various factors, including the degree of compression, the amount of distortion introduced (if using a lossy compression scheme), and the computational resources required to compress and uncompress the data.

Lossless compression algorithms usually exploit statistical redundancy in such a way as to represent the sender's data more concisely without error. Lossless compression is possible because most real-world data has statistical redundancy. For example, in English text, the letter 'e' is much more common than the letter 'z', and the probability that the letter 'q' will be followed by the letter 'z' is very small.

Run-length encoding (**RLE**) is a very simple form of data compression in which runs of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run. This is most useful on data that contains many such runs: for example, simple graphic images such as icons, line drawings, and animations. It is not useful with files that don't have many runs as it could greatly increase the file size.

RLE also refers to a little-used image format in Windows 3.x, with the extension .rle, which is a Run Length Encoded Bitmap, used to compress the Windows 3.x startup screen.

CHAPTER 3

DATA FLOW ANALYSIS

3.1 COMPRESSION ONLY

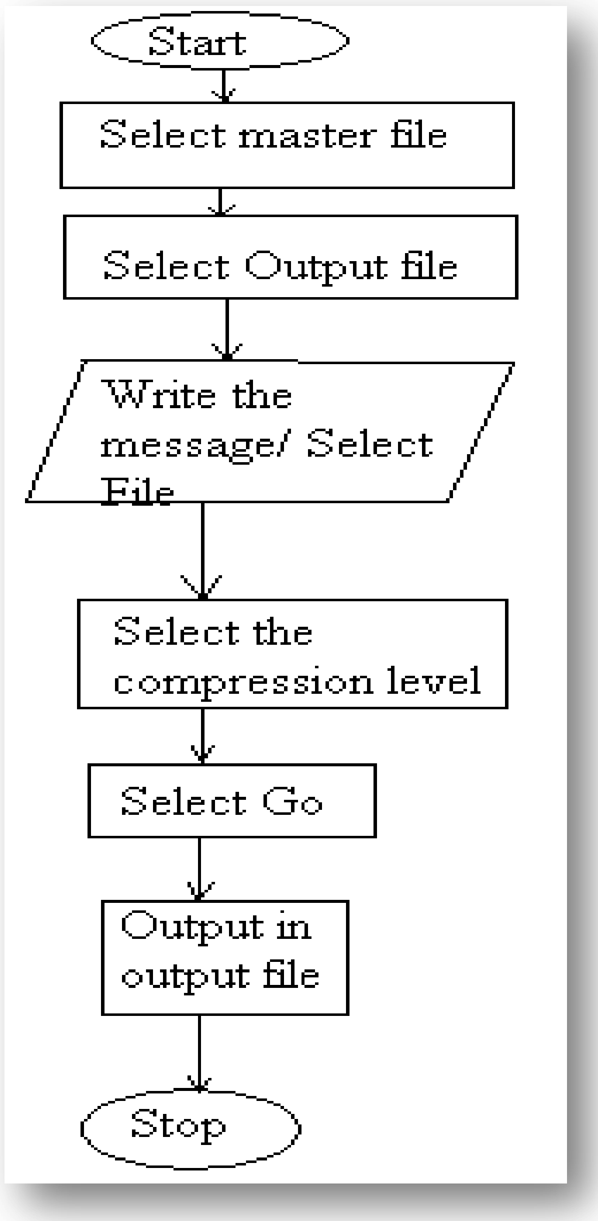


Figure 1.3

3.2 COMPRESSION WITH ENCRYPTION

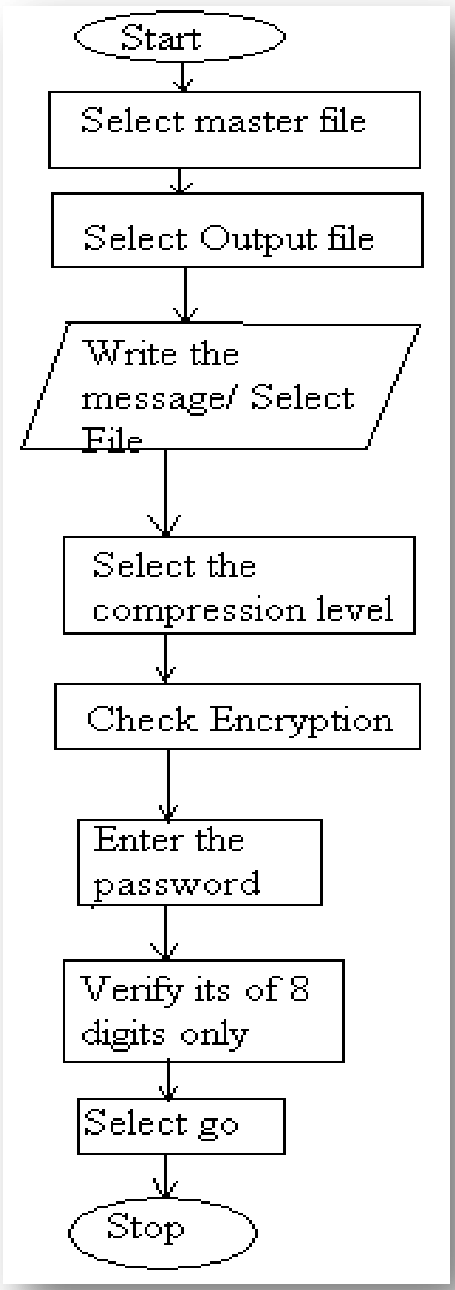


Figure 1.4

3.3 W/O COMPRESSION & W/O ENCRYPTION

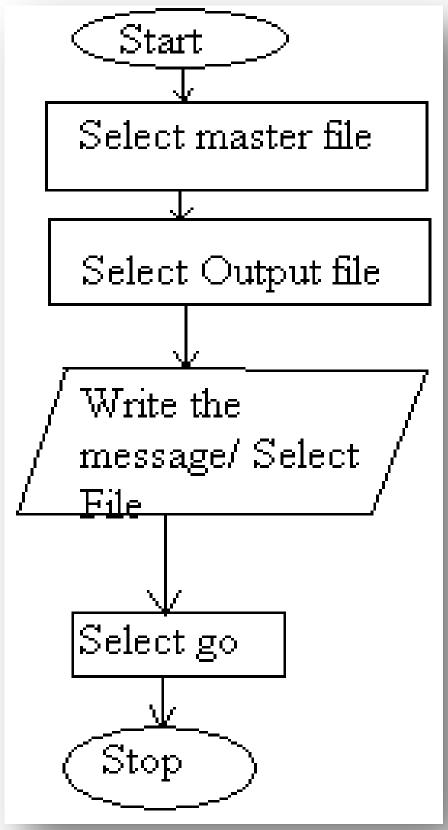


Figure 1.5

3.4 ENCRYPTION ONLY

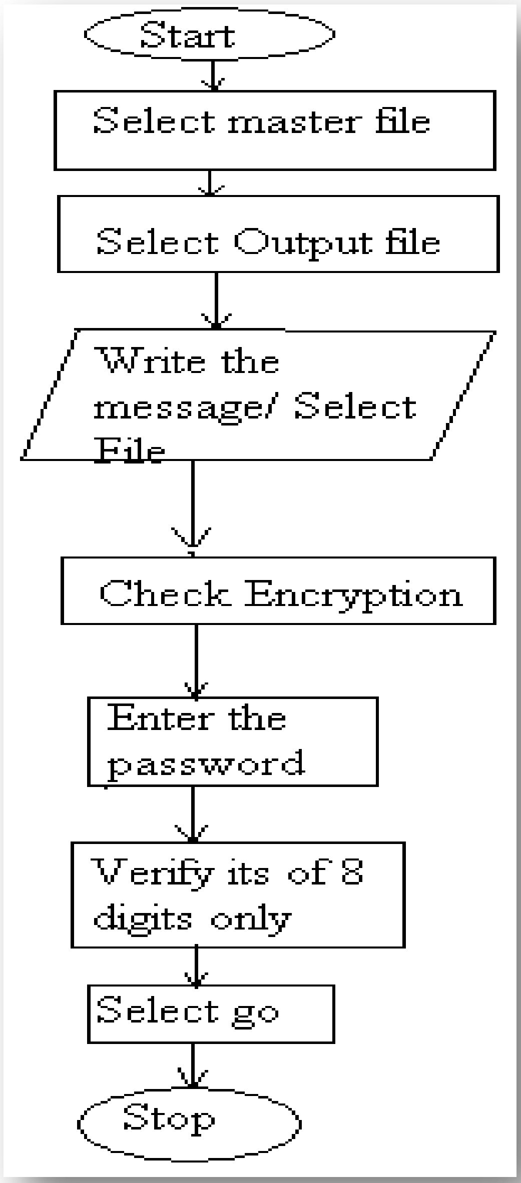


Figure 1.6

CHAPTER 4

WORKING – SNAPSHOTS

4.1 EMBEDDING MESSAGE

This is the front menu of the software. Embed message is selected:

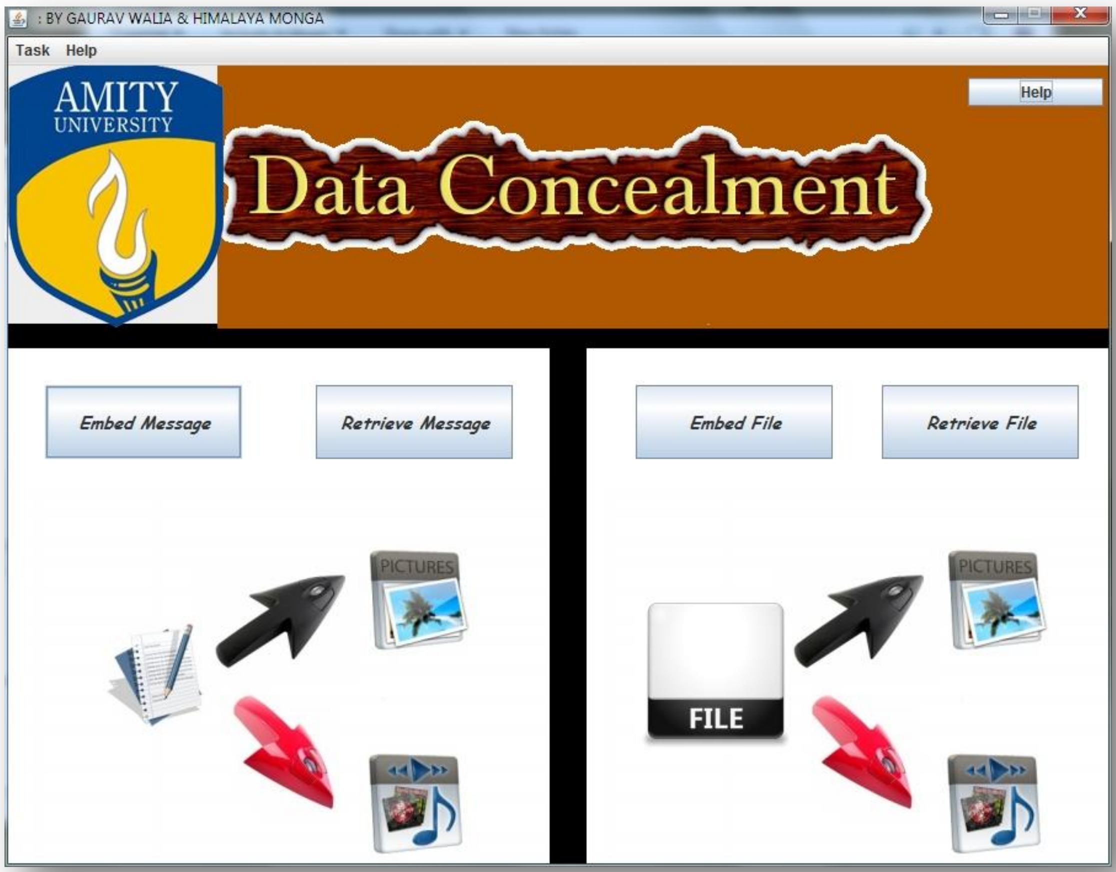
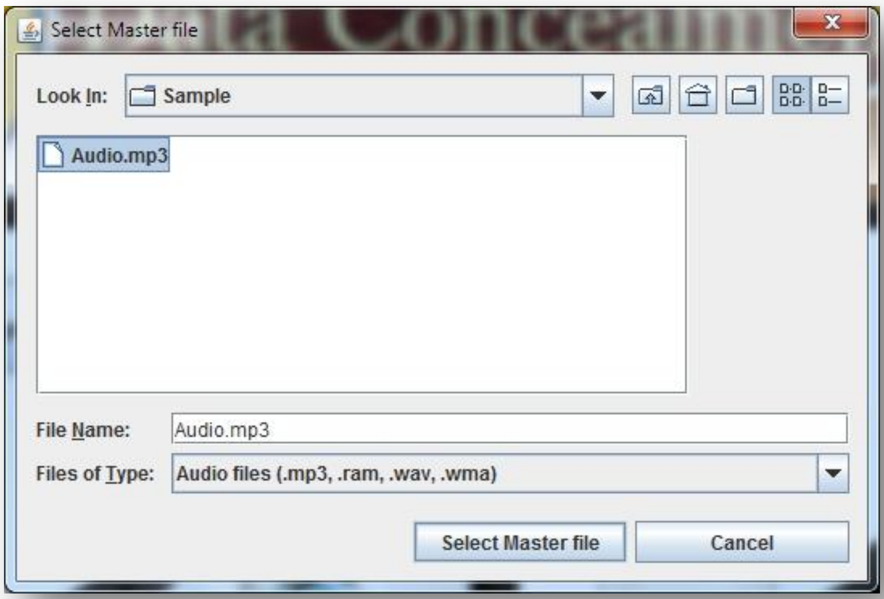


Fig. 1.7

Now selecting the Master file, means to select the type of file in which you want to embed the message into:

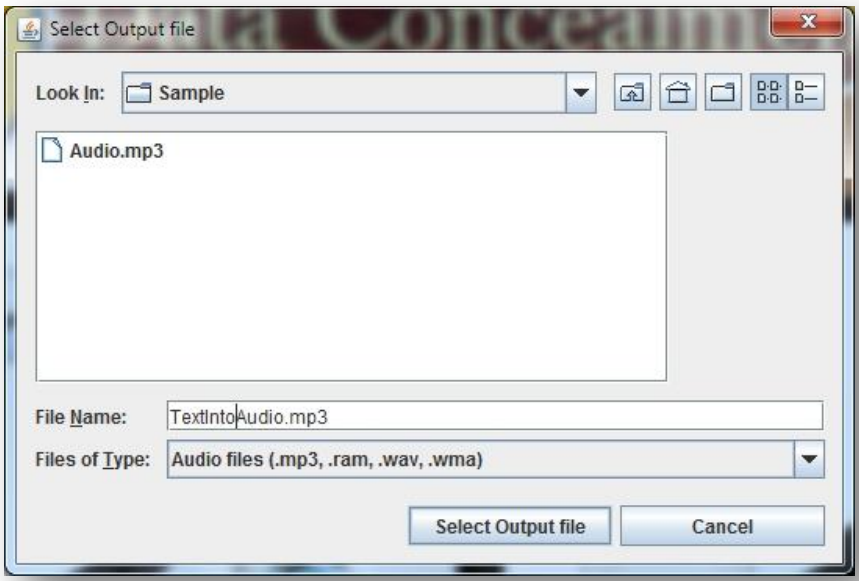


File Name: Audio.mp3

Files of Type: Audio files (.mp3, .ram, .wav, .wma)

Fig. 1.8

Now selecting the output file, means to select a new name for the file which contains the embedded message:

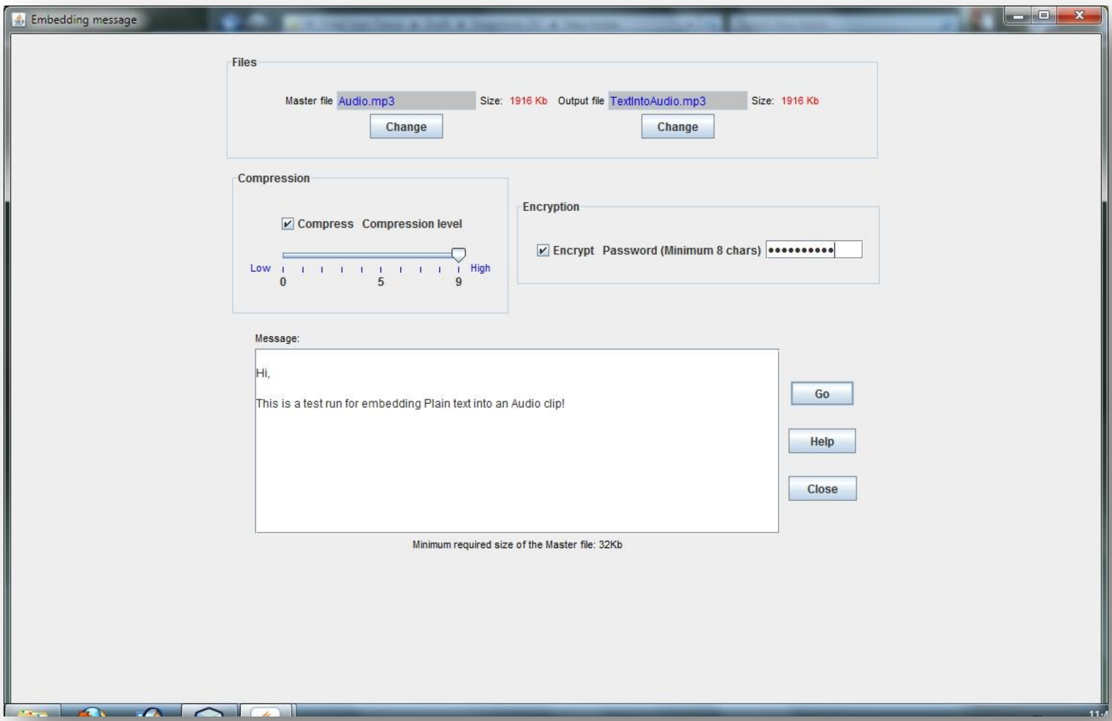


File Name: TextIntoAudio.mp3

Files of Type: Audio files (.mp3, .ram, .wav, .wma)

Fig. 1.9

This window allows us to write the plain text, set compression level and also password encrypt the file:


Embedding message

Files

Master file Audio.mp3 Size: 1916 Kb Output file TextIntoAudio.mp3 Size: 1916 Kb

Compression

✓ Compress Compression level

Encryption

✓ Encrypt Password (Minimum 8 chars) ••••••••••

Message:

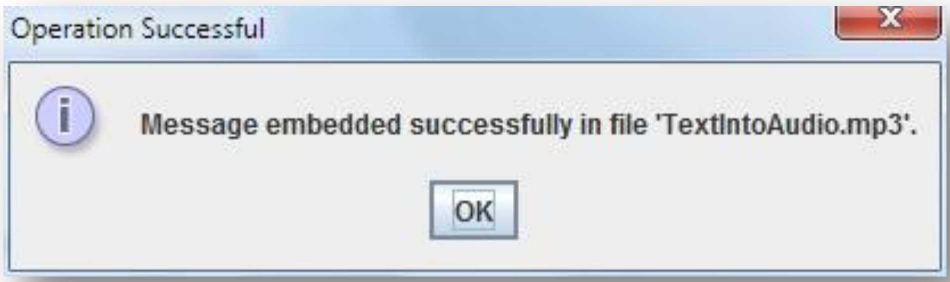
Hi,

This is a test run for embedding Plain text into an Audio clip!

Minimum required size of the Master file: 32Kb

Fig. 2.0

This message pops up on successful embedding of the text:


Operation Successful

Message embedded successfully in file 'TextIntoAudio.mp3'.

Fig. 2.1

This shows how the audio clips works perfect even with a plaintext hidden in it:

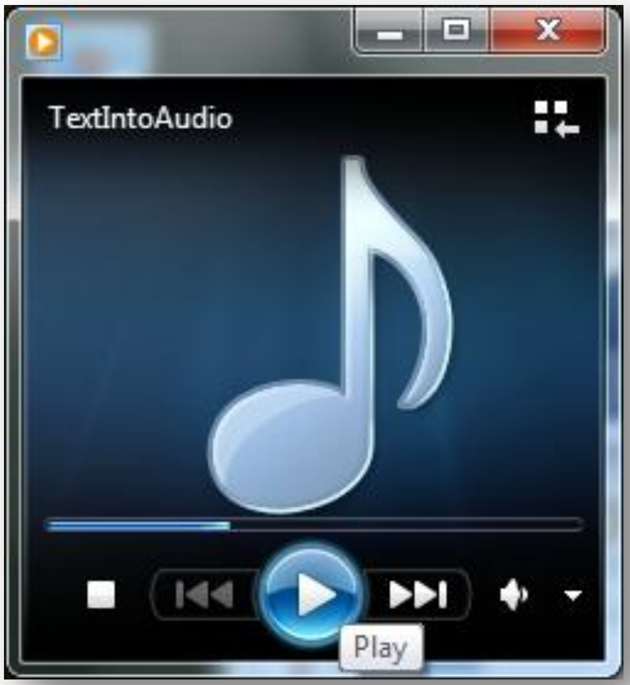


Fig. 2.2

4.2 RETRIEVE MESSAGE

The main menu. Retrieve message is selected now:

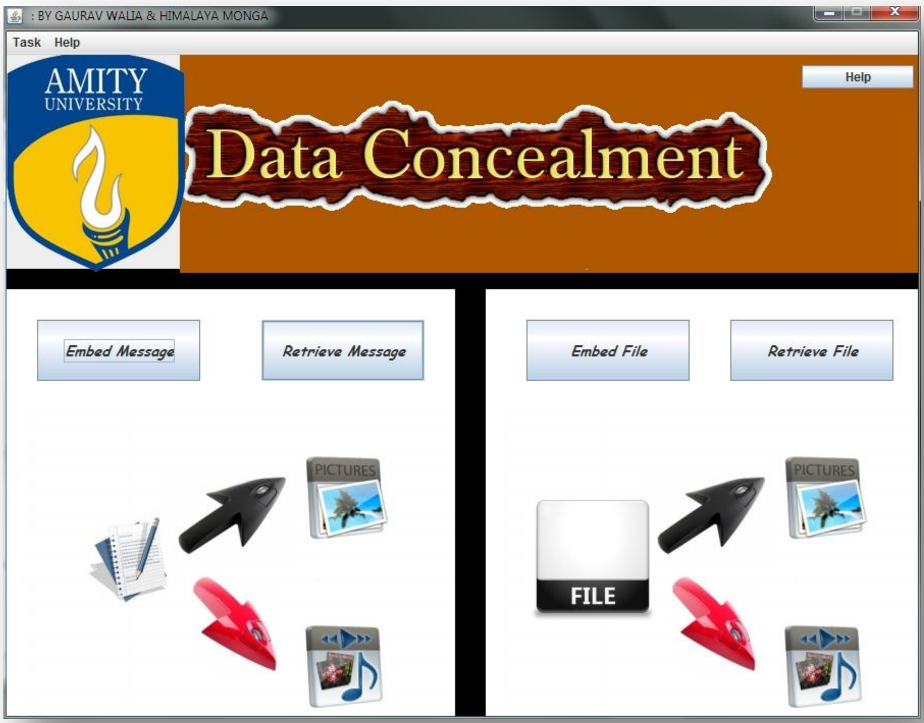
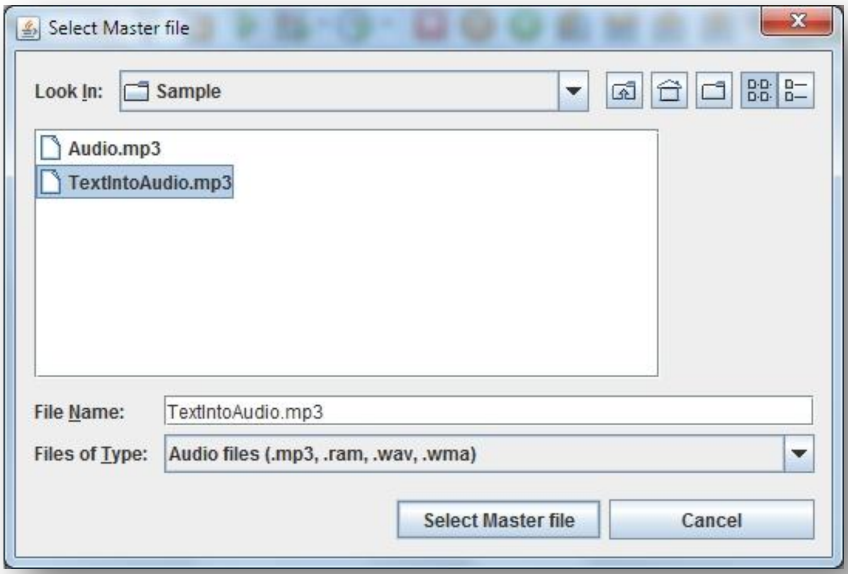


Fig. 2.3

The file previously embedded is selected:


Select Master file

Look In: Sample

Audio.mp3
TextIntoAudio.mp3

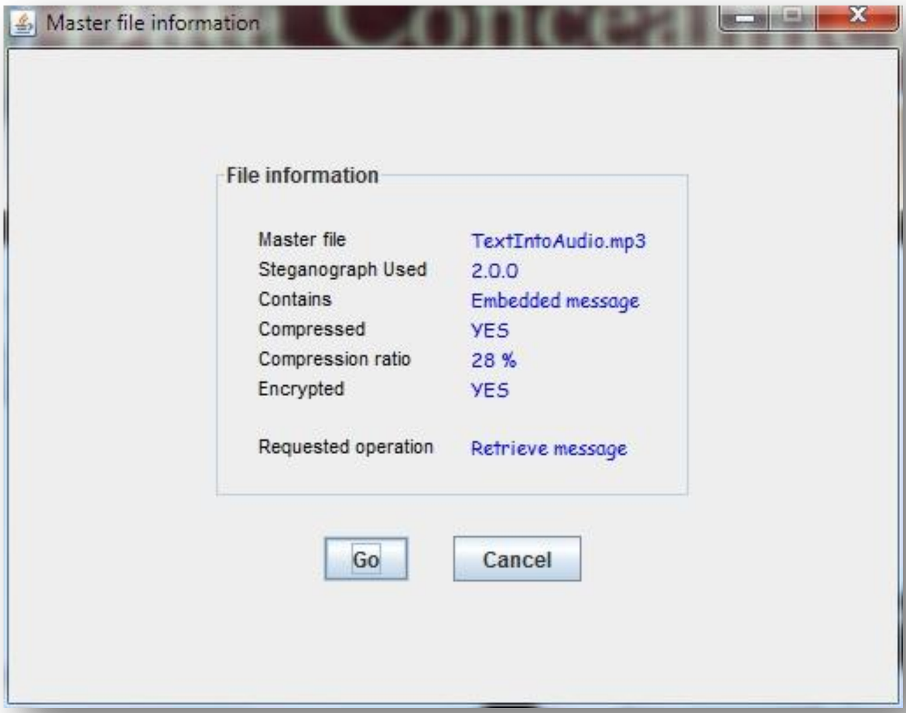
File Name: TextIntoAudio.mp3

Files of Type: Audio files (.mp3, .ram, .wav, .wma)

Select Master file | Cancel

Fig. 2.4

This shows the file info:



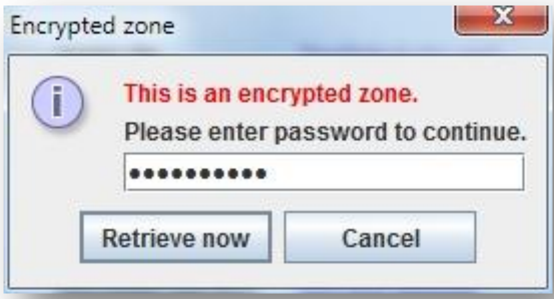
Master file information

File information

Master file	TextIntoAudio.mp3
Steganograph Used	2.0.0
Contains	Embedded message
Compressed	YES
Compression ratio	28 %
Encrypted	YES
Requested operation	Retrieve message

Fig. 2.5

Encryption box:



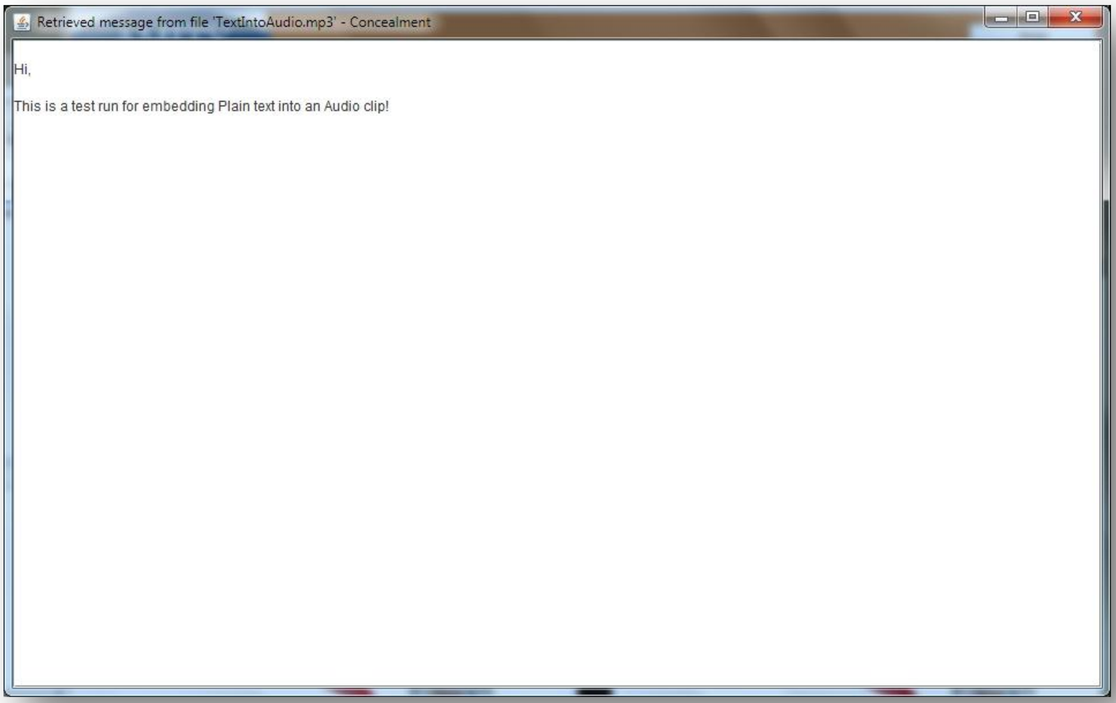
Encrypted zone

This is an encrypted zone.
Please enter password to continue.

●●●●●●●●●●●

Fig. 2.6

Decodes the message:


Retrieved message from file 'TextIntoAudio.mp3' - Concealment

Hi,

This is a test run for embedding Plain text into an Audio clip!

Fig. 2.7

4.3 EMBEDDING FILE

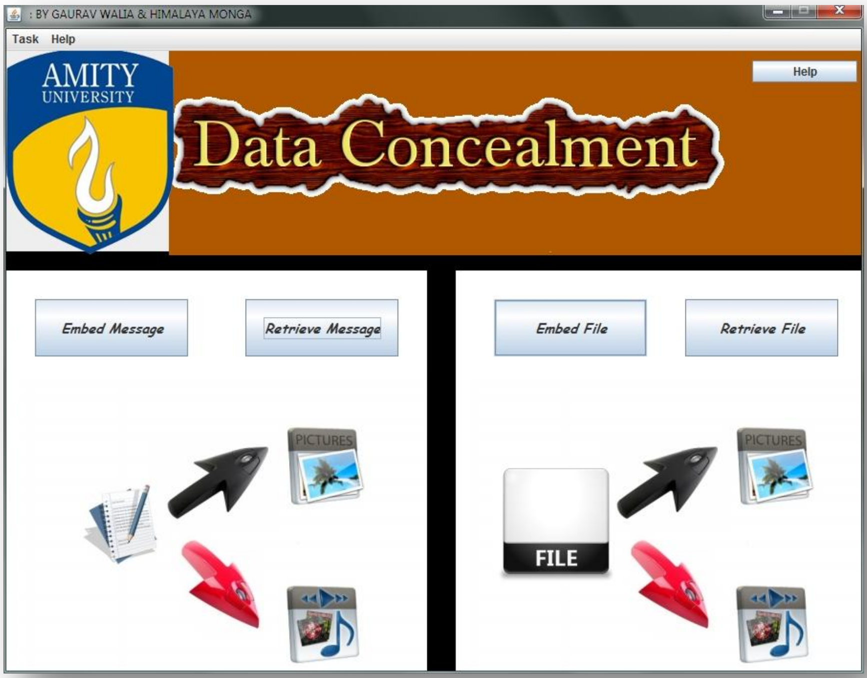


Fig. 2.8

Selecting master file means which file's identity to be retained after merger:

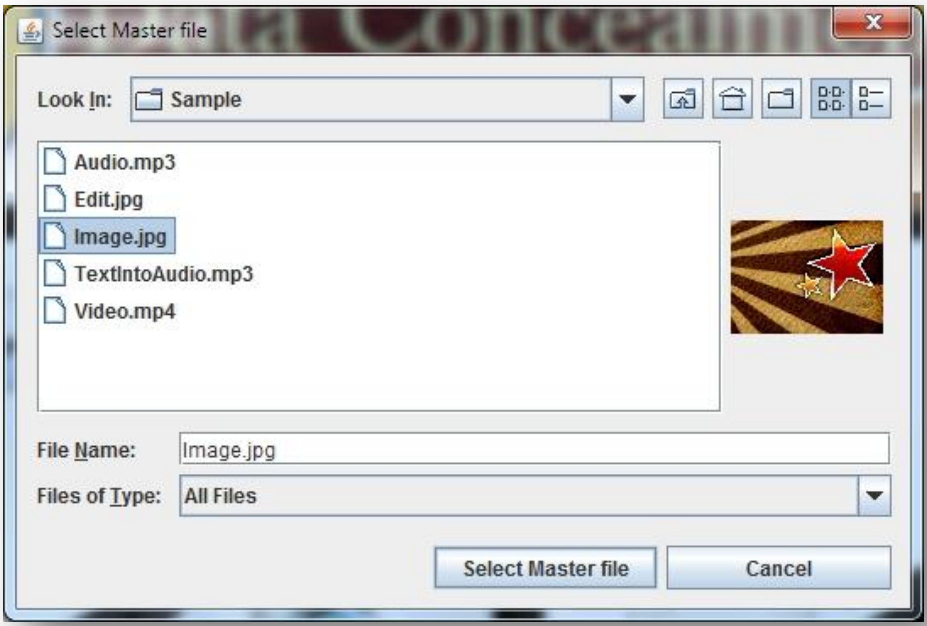


Fig. 2.9

Name of the new file:

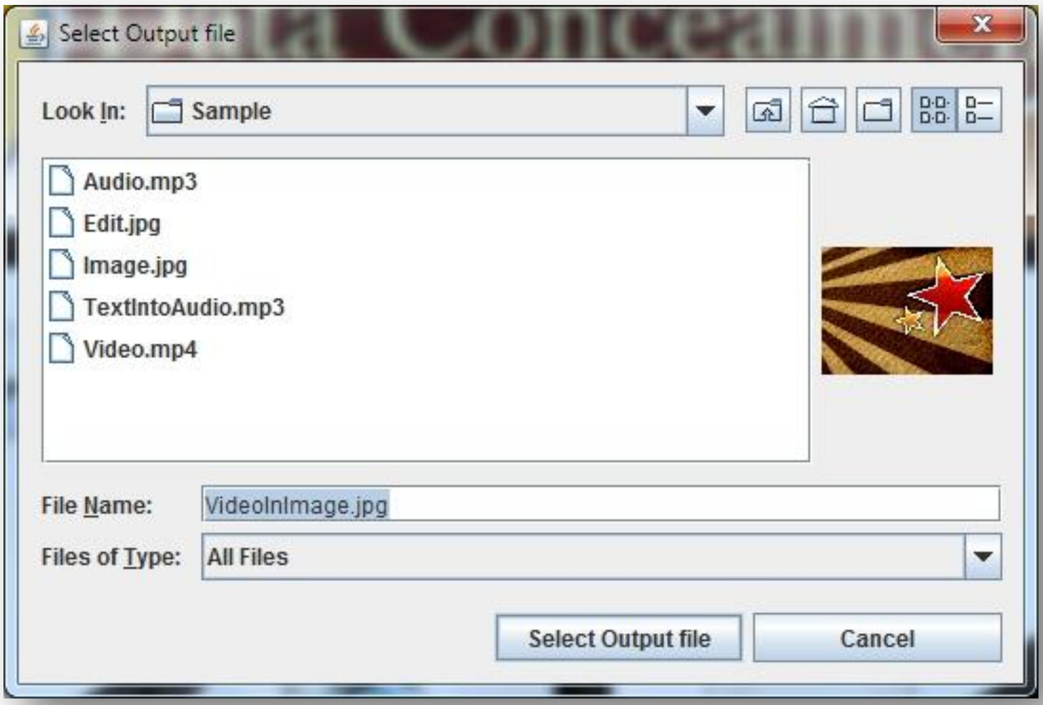


Fig. 3.0

Name of the input file:

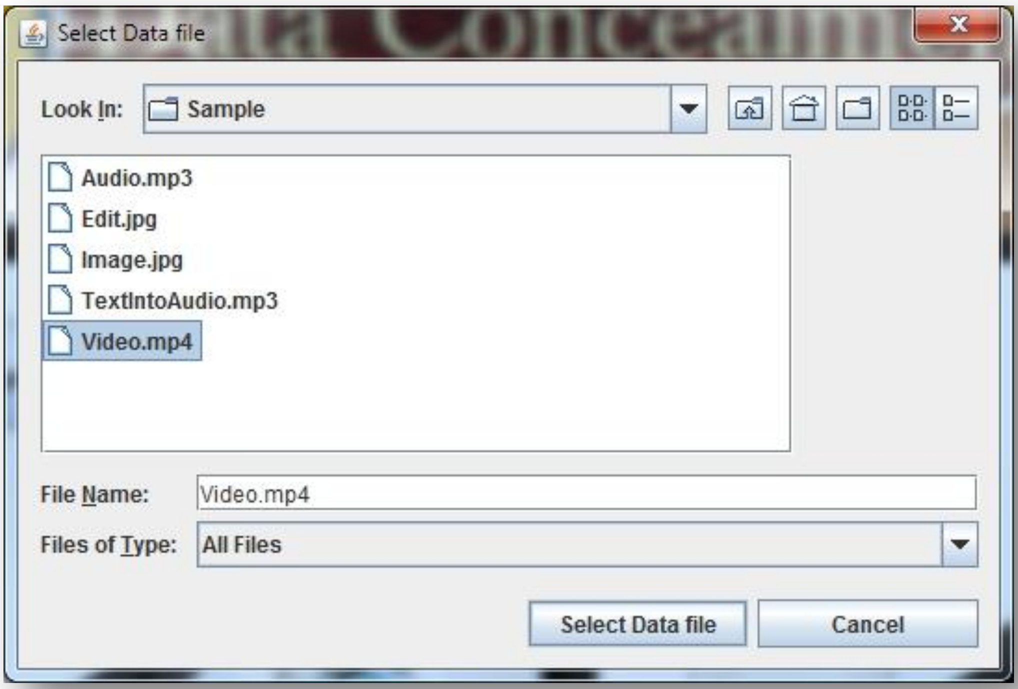
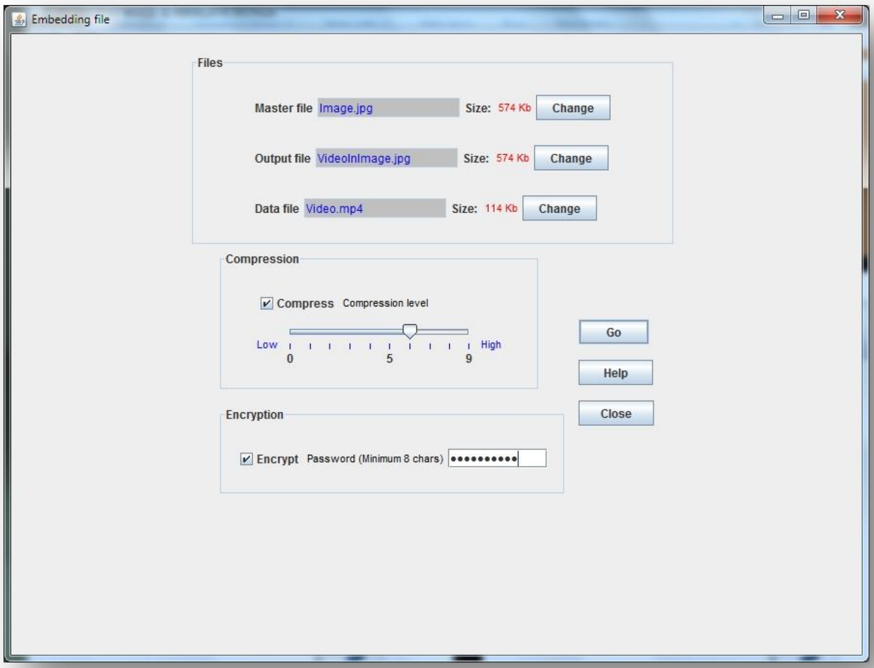


Fig. 3.1

File embed info requirement/check + Encryption:


Embedding file

Files

Master file Image.jpg Size: 574 Kb

Output file VideoInImage.jpg Size: 574 Kb

Data file Video.mp4 Size: 114 Kb

Compression

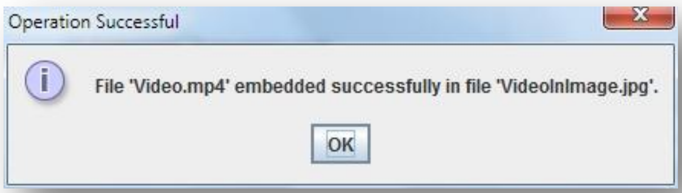
☑ Compress Compression level

Encryption

☑ Encrypt Password (Minimum 8 chars) ●●●●●●●●●●

Fig. 3.2

Success:


Operation Successful

File 'Video.mp4' embedded successfully in file 'VideoInImage.jpg'.

Fig. 3.3

Shows the image is still working even after a video already embedded:

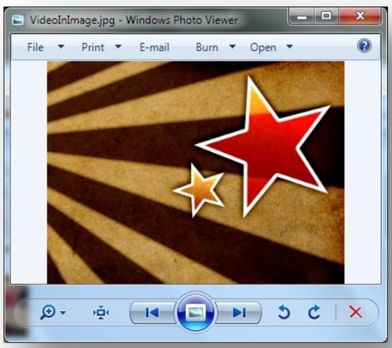


Fig. 3.4

4.4 RETRIEVING FILE

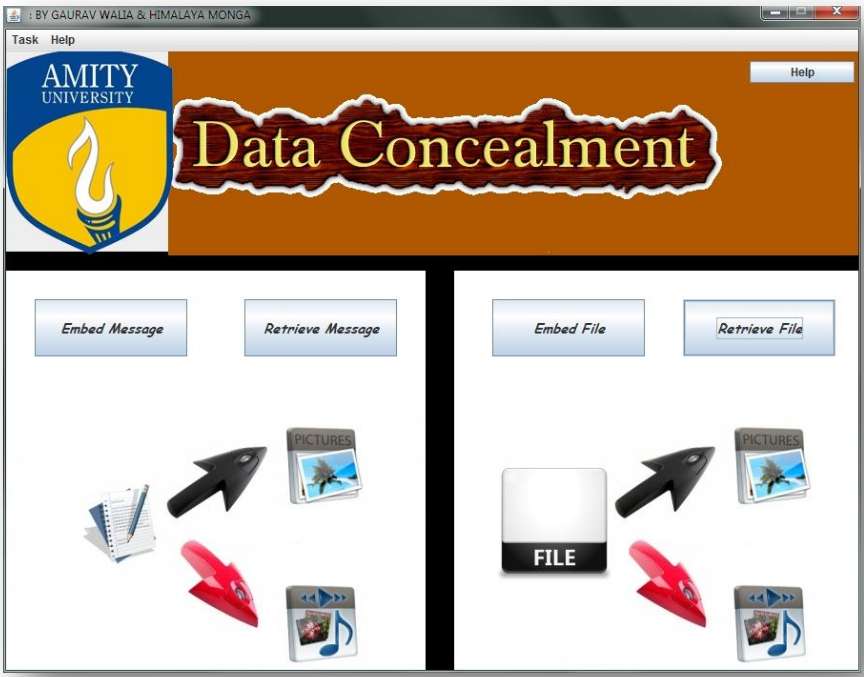
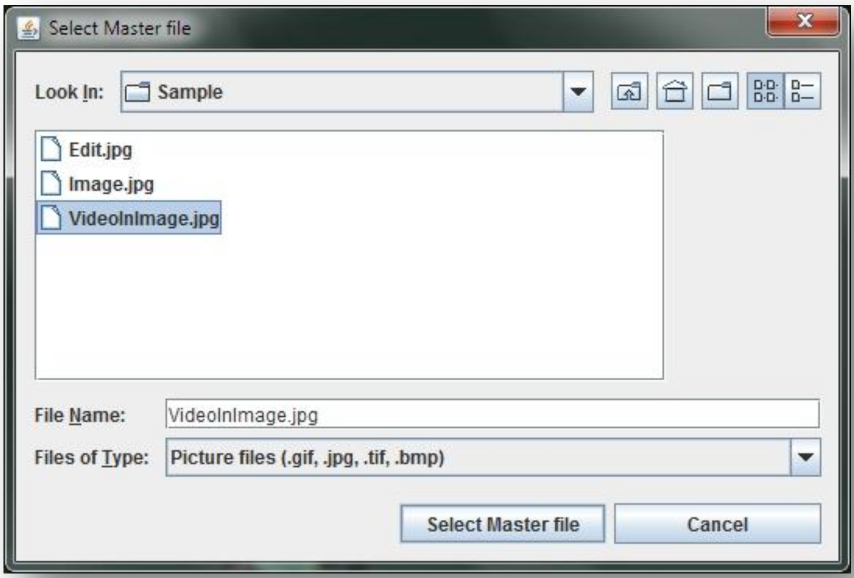


Fig. 3.5

Selecting the already embedded file:



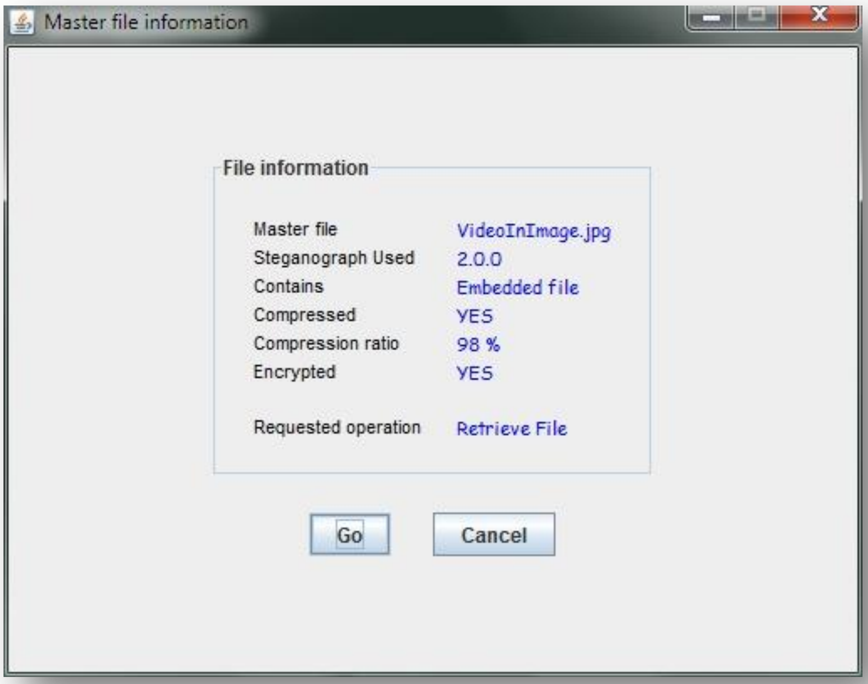
Edit.jpg
Image.jpg
VideoInImage.jpg

File Name: VideoInImage.jpg

Files of Type: Picture files (.gif, .jpg, .tif, .bmp)

Fig. 3.6

Displays file info:



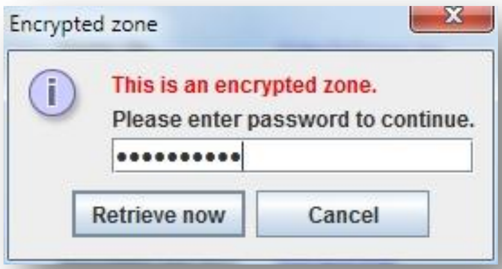
Master file information

File information

Master file	VideoInImage.jpg
Steganograph Used	2.0.0
Contains	Embedded file
Compressed	YES
Compression ratio	98 %
Encrypted	YES
Requested operation	Retrieve File

Fig. 3.7

Encryption box:

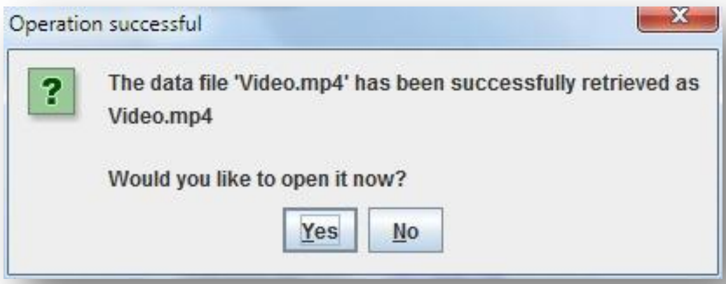


Encrypted zone

This is an encrypted zone.
Please enter password to continue.

Fig. 3.8

Asking to open the retrieved file:



Operation successful

The data file 'Video.mp4' has been successfully retrieved as Video.mp4

Would you like to open it now?

Fig. 3.9

The retrieved video play is shown:

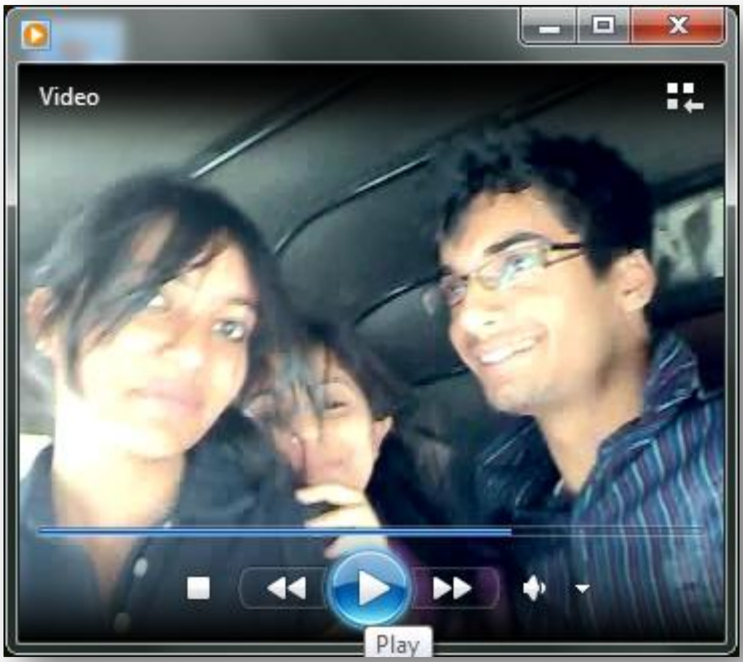


Fig. 4.0

CHAPTER 5

INDIVIDUAL CONTRIBUTION

In the successful completion of our project each of the team member contributed whole heartedly towards the development of the project. This project would not have been completed without the contribution of any of the team member. Among the fourteen functions that our project supports I was responsible to develop and execute the following seven functions:

1. **Concealing plain text in a video file:**
2. **Concealing image in an image file:**
3. **Concealing image in an audio file:**
4. **Concealing audio in an audio file:**
5. **Concealing audio in an video file:**
6. **Concealing video in an image file:**
7. **Dynamic file compression:**

After the successful concealment of plain text into image ad audio, the plain text is concealed in a video. The plain text is known as the input file, the video in which the text is going to be concealed is known as the mater file and the new file we will get after the concealment is named as the output file. The selection of input file (plain text), master file (audio file) and the name of the output file are chosen in the beginning.

Each file which is to be concealed named as the input files first stored in an array in the byte format similarly the master file is converted into the byte format and is being stored in an array. The bits of the input file are then replaced by the least significant bit (LBS) of every byte of the master file. Thus the data concealment is performed.

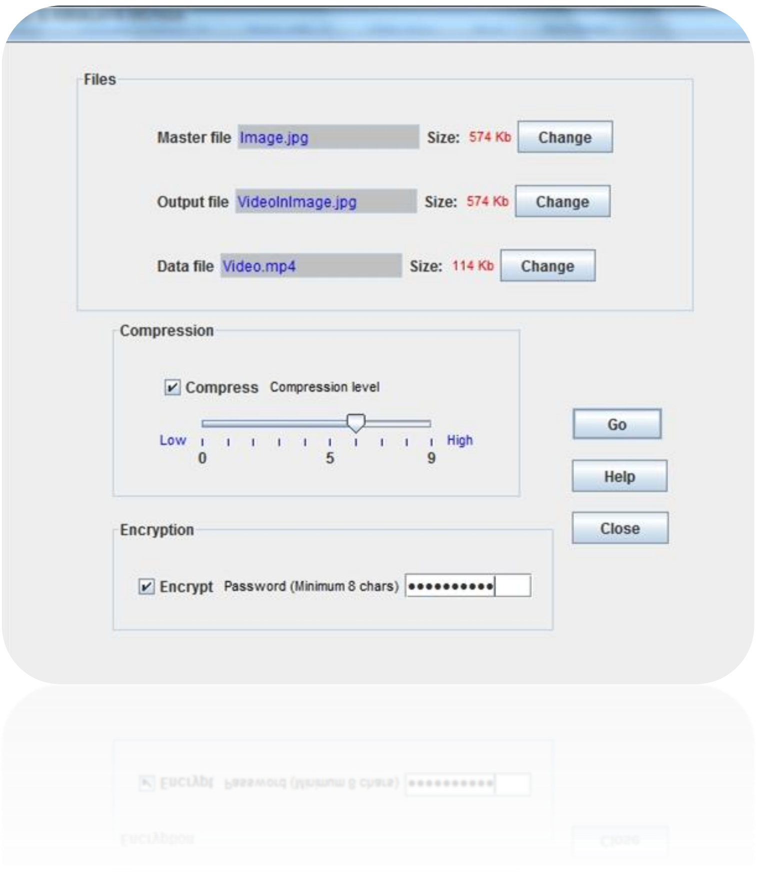
After the concealment of plain text in an image and audio, the next step was to conceal text in a video. The extraction of a video file in byte format was similar to extraction of image and audio in byte format.

Concealing an image in an image is known as steganography. The technique I used was LSB substitution in which the Least Significant Bits of the image is replaced by the bits of input file.

Apart from the substitution of least significant bits (LSBs) I performe*d* dynamic file compression in which user is asked to compress a file after the embedding is done.

There is a horizontal scroll bar fixed to decide the compression level.

- The key areas were to apply the compression algorithm and blending it with the encryption scheme.
- The module individually designed was very carefully merged with the other team member's module, to create a coherent system for data concealment.
- The compression part was also merged with a friendly GUI, here is how :



Files

Master file Image.jpg Size: 574 Kb

Output file VideoInImage.jpg Size: 574 Kb

Data file Video.mp4 Size: 114 Kb

Compression

[✓] Compress Compression level

Encryption

[✓] Encrypt Password (Minimum 8 chars) ●●●●●●●●●●

CHAPTER 6

CONCLUSION

Project has been completed and functioning properly with the provided properties. Encryption and Compression is independent of each other. Feature enables the users to filter the features as needed. Steganography implemented allows the user to create new file if main file is not to be modified. Security of the system has been done through password. Various levels of compression can be implemented. All features implemented are as per user requirements. GUI designed in Java is user friendly. Only a batch file is there that ca be executed to execute the source code which is not possible through any other software. Basic Software requirements are simple and don’t require much hardware. Project completed as per the features in all respects.

CHAPTER 7

FUTURE RESEARCH

- Other Encryption techniques can also be implemented.
- A combination of text and file can also be implemented.
- Authentication of user using this software can also be done using passwords for security in the organization.
- A more user friendly environment can also be created but that will hamper the system memory requirements.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Information Hiding: Steganography and Watermarking : Attacks and Countermeasures *(By Neil F. Johnson, Zoran Duric, Sushil Jajodia)*

- Digital Video Compression (By Peter D. Symes)

- Compression Algorithms for Real Programmers (By Peter Wayner)

WEB PORTALS:

- http://java.sun.com/developer/technicalArticles/Programming/compression/
- http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/image/WritableRaster.html