



P-3616



**SECURED COMPUTER ASSISTED INTELLIGENCE REPORT
EXCHANGE**

A PROJECT REPORT

Submitted by

KAARTHIKA S J 0710108302

RAMYA S 0710108040

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

**Autonomous institution Affiliated to Anna University of Technology,
Coimbatore.**

APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY: COIMBATORE-641 049

BONAFIDE CERTIFICATE

Certified that this project report entitled “Secured Computer Assisted Intelligence Report Exchange” is the bonafide work of Kaarthika S J, Ramya S who carried out the research under my supervision. Certified also, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.



SIGNATURE

Mrs.P.Devaki,M.E,(Ph.D)

HEAD OF THE DEPARTMENT

Department of Computer

Science and Engineering

Kumaraguru College of Technology

Coimbatore-641049



SIGNATURE

Mr.K.Sivan Arul Selvan, M.E,(Ph.D)

SUPERVISOR

ASSISTANT PROFESSOR (SRG)

Department of Computer

Science and Engineering

Kumaraguru College of Technology

Coimbatore-641049

The candidate with University Register Nos. 0710108302 and 0710108040 were examined by us in the project viva-voce examination held on 20/4


INTERNAL EXAMINER


EXTERNAL EXAMINER

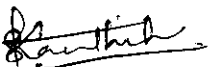
DECLARATION

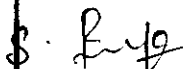
We hereby declare that the project entitled “ **Secured Computer Assisted Intelligence Report Exchange** “ is a record of original work done by us and to the best of our knowledge, a similar work has not been submitted to Anna University or any Institutions, for fulfillment of the requirement of the course study.

The report is submitted in partial fulfillment of the requirement for the award of the Degree of Bachelor of Computer Science and Engineering of Anna University, Coimbatore.

Place: Coimbatore

Date: 19.4.2011


(Kaarthika S J)


(Ramya S)

ACKNOWLEDGEMENT

We are intend to express our heartiest thanks to our chairman **Arutselvar Dr.N.Mahalingam ,B.sc., F.I.E** and the Co-Chairman **Dr.B.K.Krishnaraj Vanavarayar** for given us this opportunity to embark on this project.

We would like to make a special acknowledgement and thanks to our correspondent **M.Balasubramaniam, M.com., M.B.A.**, and our Director **Dr.J.Shanmugam** for his support and encouragement throughout the project.

We extend our sincere thanks to our Principal, **Dr. S.Ramachandran Ph.D.**, Kumaraguru College of Technology, Coimbatore, for being a constant source of inspiration and providing us with the necessary facility to work on this project.

We are indent to express my heartiest thanks to **Mrs.P.Devaki M.E,(Ph.D)**, *Project coordinator* ,Head of the Department of Computer Science &Engineering, for her valuable guidance and useful suggestions during the course of this project.

We express deep gratitude and gratefulness to our **Guide Mr.K.SivanArulSelvan M.E.,(Ph.D).**, *Assistant Professor(SRG)* Department of Computer Science & Engineering, for his supervision, enduring patience, active involvement and guidance.

We would like to convey our honest thanks to **all Faculty members** of the Department for their enthusiasm and wealth of experience from which we have greatly benefited.

We also thank our **friends and family** who helped us to complete this project fruitfully.

ABSTRACT

Owing to the advancement in network technology, information security is an increasingly important problem. The ability of network to transmit enormous amount of data leads us to acquire information through Multimedia Technology.

Steganography is a hiding technique used to ensure the security and secrecy of Original data. In our work, the message to be Steganographed (the plaintext) is first encrypted using Advanced Encryption Standard Algorithm, and then the Cover image is modified to contain the encrypted message (cipher text), resulting in stegoimage. To apply SLSB steganographic techniques cover files of image is used. The steganographic method that works in the spatial domain is the LSB (Least Significant Bit) method, which replaces the least significant bits of selected pixels to hide the information.

Here SLSB (Selected Least Significant Bit) method hiding information in only one of the three colors at each pixel of the cover image. The Image File of BMP format in which the text has to be hidden is taken as cover file. The text file is encrypted using AES algorithm and embedded into the cover file. To cover and uncover the text file an application is implemented using JAVA Programming language which provides authentication.

CHAPTER NO	TITLE	PAGE
	ABSTRACT	V
	LIST OF ABBREVIATIONS	IX
1	INTRODUCTION	1
	1.1 PROBLEM DEFINITION	2
	1.2 PURPOSE OF PROJECT	2
	1.3 EXISTING SYSTEM	2
	1.4 PROPOSED SYSTEM	3
2	LITERATURE REVIEW	3
	2.1 CRYPTOGRAPHY	3
	2.2 TYPES OF CRYPTOGRAPHY	4
	2.3 CRYPTOGRAPHY'S ROLE IN SECURITY	5
	2.4 STEGANOGRAPHY	5
	2.5 STEGANOGRAPHY PAST, PRESENT AND FUTURE	6
	2.6 APPLICATIONS OF STEGANOGRAPHY	8
	2.7 STEGANOGRAPHY'S ROLE IN SECURITY	10
	2.8 COMPARISON BETWEEN CRYPTOGRAPHY, STEGANOGRAPHY, WATERMARKING AND	10

	FINGERPRINTING	
	2.9 IMAGE STEGANOGRAPHY	11
	2.10 ADVANCED ENCRYPTION STANDARD ALGORITHM	12
	2.11 BITMAP FILE FORMAT	18
	2.12 LEAST SIGNIFICANT BIT ALGORITHM	23
	2.13 ADVANTAGES OF LSB ALGORITHM	29
	2.14 SELECTED LEAST SIGNIFICANT BIT ALGORITHM	29
	2.15 WHY JAVA?	31
3	SYSTEM REQUIREMENTS AND ANALYSIS	34
	3.1 FEASIBILITY	35
	3.1.1 FEASIBILITY STUDY	35
	3.2 SPECIFIC REQUIREMENTS	38
	3.2.1 FUNCTIONAL REQUIREMENTS	38
	SOFTWARE SPECIFICATION	38
	HARDWARE SPECIFICATION	38
	3.2.2 PERFORMANCE REQUIREMENTS	38
	INVISIBILITY	38
	SECURITY	38

	ROBUSTNESS	38
	INVERTIBILITY	39
4	SYSTEM STUDY	39
5	MODULES	40
	5.1 DATA ENCRYPTION	40
	5.2 DATA EMBEDDING	40
	5.3 DATA EXTRACTION	40
	5.4 DATA DECRYPTION	40
6	CONCLUSION	41
7	FUTURE ENHANCEMENTS	41
8	APPENDIX	42
	8.1 SAMPLE SOURCE CODE	42
	8.2 SAMPLE OUTPUT	63
9	REFERENCES	68

LIST OF ABBREVIATIONS

SNO	ABBREVIATION	EXPANSION
1	DES	Data Encryption Standard
2	AES	Advanced Encryption Standard
3	LSB	Least Significant Bit
4	SLSB	Selected Least Significant Bit
5	BMP	BitMap Picture
6	DIB	Device Independent Bitmap
7	GUI	Graphical User Interface
8	API	Application Program Interface
9	SKC	Secret Key Cryptography
10	PKC	Public Key Cryptography
11	DCT	Discrete Cosine Transform

SECURED COMPUTER ASSISTED INTELLIGENCE REPORT EXCHANGE

CHAPTER I

1. INTRODUCTION

Information security means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification or destruction. The need to protect information to ensure its confidentiality, integrity, and availability to those whom need it for making critical personal, business, or government decisions becomes more important. The advancement of technology, the Internet, and information sharing has had both positive and negative impacts. One of the negative impacts was the large increase in new “information” threats.

The following is a list of common threats to most information systems:

- Unauthorized access, alteration, or destruction of information.
- Misuse of authorized access to information.
- Accidental alteration or destruction of information.
- Malicious software programs (viruses/worms/trojans).
- Misconfigured or poorly designed information systems allowing too much
- access.
- Social engineering.
- System or communications disruptions (denial of service, hardware failure).

Since the rise of the Internet one of the most important factors of information technology and communication has been the security of information. Cryptography was created as a technique for securing the secrecy of communication and many different methods have been developed to encrypt and decrypt data in order to keep the message secret. Unfortunately it is sometimes not enough to keep the contents of a message secret, it may also be necessary to keep the existence of the message secret. The technique used to implement this, is called steganography.

1.1 PROBLEM DEFINITION

This application provides a mechanism for hiding and extracting the text into and from an image file. This can be adopted by using SLSB steganographic technique. The image file is in .bmp file format.

1.2PURPOSE OF PROJECT

The purpose of our project is to provide secure information transfer using a software for Intelligence agencies with the help of steganographic technique.

1.3 EXISTING SYSTEM:

Most of the algorithms that work in the spatial domain (altering the desired characteristics on the file itself) using a LSB method as the algorithm for information hiding, that is, hide one bit of information in the least significant bit of each color of a pixel. The problem stems from the fact that modifying the three colors of a pixel produces a major distortion in the resulting color. This distortion is not visible to the human eye, but detectable by statistical analysis.

1.4 PROPOSED SYSTEM:

In Our project we are implementing a new method Selected Least Significant Bit (SLSB), that improves the performance of the method LSB hiding information in only one of the three colors at each pixel of the cover image. This method would introduce more efficiency and less distortion.

CHAPTER II

2. LITERATURE REVIEW

2.1 CRYPTOGRAPHY

Cryptography is the science of writing in secret code and is an ancient art; the first documented use of cryptography in writing dates back to circa 1900 B.C. when an Egyptian scribe used non-standard hieroglyphs in an inscription. Some experts argue that cryptography appeared spontaneously sometime after writing was invented, with applications ranging from diplomatic missives to war-time battle plans. It is no surprise, then, that new forms of cryptography came soon after the widespread development of computer communications. In data and telecommunications, cryptography is necessary when communicating over any untrusted medium, which includes just about any network, particularly the Internet.

Within the context of any application-to-application communication, there are some specific security requirements, including:

- **Authentication:** The process of proving one's identity. (The primary forms of host-to-host authentication on the Internet today are name-based or address-based, both of which are notoriously weak.)

- Privacy/confidentiality: Ensuring that no one can read the message except the intended receiver.
- Integrity: Assuring the receiver that the received message has not been altered in any way from the original.
- Non-repudiation: A mechanism to prove that the sender really sent this message.

Cryptography, then, not only protects data from theft or alteration, but can also be used for user authentication. There are, in general, three types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography, and hash functions, each of which is described below. In all cases, the initial unencrypted data is referred to as plaintext. It is encrypted into ciphertext, which will in turn (usually) be decrypted into usable plaintext.

2.2 TYPES OF CRYPTOGRAPHIC ALGORITHMS

There are several ways of classifying cryptographic algorithms. The three types of algorithms are :

- Secret Key Cryptography (SKC): Uses a single key for both encryption and decryption
- Public Key Cryptography (PKC): Uses one key for encryption and another for decryption
- Hash Functions: Uses a mathematical transformation to irreversibly "encrypt" information.

2.3 CRYPTOGRAPHY'S ROLE IN SECURITY

Cryptography comes from the Greek words for "secret writing". Professionals make a distinction between ciphers and codes. A cipher is a character-for-character or a bit-for-bit transformation, without regard to the linguistic structure of the message. In contrast, a code replaces one word with another word or symbol.

Until the advent of computers, one of the main constraints on cryptography had been the ability of the code clerk to perform the necessary transformations, often on a battlefield with little equipment. Few years ago, cryptography was widely used to secure the data. But crackers always found a method to detect the key. Because of the increase in the intruding, the algorithms used kept changing. But each time, it has led to failures. This is because the message or audio was clearly visible to the cracker, which showed them an easy way to found the information.

2.4 STEGANOGRAPHY

Steganography is the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity. The word steganography is of Greek origin and means "concealed writing" from the Greek words steganos meaning "covered or protected", and graphein meaning "to write".

The advantage of steganography, over cryptography alone, is that messages do not attract attention to themselves. Therefore, whereas cryptography protects the contents of a message, steganography can be said to protect both messages and communicating parties.

Steganography includes the concealment of information within computer files. In digital steganography, electronic communications may include steganographic coding inside of a transport layer, such as a document file, image file, program or protocol. Media files are ideal for steganographic transmission because of their large size. As a simple example, a sender might start with an innocuous image file and adjust the color of every 100th pixel to correspond to a letter in the alphabet, a change so subtle that someone not specifically looking for it is unlikely to notice it.

2.5 STEGANOGRAPHY – PAST, PRESENT AND FUTURE

Past

Johannes Trithemius (1462-1516) was a German Abbot. His writing, “Steganography: the art through which writing is hidden requiring recovery by the minds of men.” The first two parts of his works are apparently some of the first books on cryptology describing methods to hide messages in writing. The third part of the trilogy is outwardly a book on occult astrology. The third book contains a number of tables containing numbers. In more recent history, several stenographic methods were used during World War II. Microdots developed by the Nazis are essentially microfilm chips created at high magnification (usually over 200X). These microfilm chips are the size of periods on a standard typewriter. These dots could contain pages of information, drawings, etc. The Nazis also employed invisible inks and null ciphers.

Present

Currently, the emphasis has been on various forms of digital steganography. Commonly there are a number of digital technologies that the community is concerned with, namely text files, still images, movie images, and audio. Steganographic methods includes two primary groups. “Image Domain tools

encompass bit-wise methods that apply least significant bit (LSB) insertion and noise manipulation. The transform domain group of tools include those that involve manipulation of algorithms and image transforms such as discrete cosine transformation (DCT) and wavelet transformation .” Beyond the concerns of hidden messages in images, there has been additional concern voiced regarding the television broadcast of bin Laden. Remembering that steganography is hardly the sole property of digital technology, there is the possibility that there could have been hidden messages in the audio portion of the broadcasts, or even in the background of the televised images.

Other interests who are making use of steganographic techniques are involved in the application of digital watermarks. Using a variety of techniques, images, music, movies can be imprinted with digital watermarks. Watermarks are available in several configurations: fragile vs. robust, visible vs. invisible, and private vs. public. Fragile watermarks are those that are easily destroyed by image manipulation, and find utilization in image authentication systems.

Future

According to Richard E. Smith (a data security expert), he doesn't "see many practical uses for steganography because it only works as long as nobody expects you to use it." The author respectfully takes exception to this statement. Initially after reading this statement, the myth that Charles H. Duell, Commissioner of Patents in 1899 had declared that the Patent Office should be closed because everything that could possibly be invented had already been invented came to mind. Perhaps the computer security community should give up on endless patches, security applications, etc because they only work if nobody expects that they are in use. To quote Dale Carnegie, "Most of the important things in the world

have been accomplished by people who have kept on trying when there seemed to be no hope at all.”

2.6 APPLICATIONS OF STEGANOGRAPHY

Steganography is applicable to, but not limited to, the following areas.

- 1) Confidential communication and secret data storing
- 2) Protection of data alteration
- 3) Access control system for digital content distribution
- 4) Media Database systems

The area differs in what feature of the steganography is utilized in each system.

1. Confidential communication and secret data storing

The "secrecy" of the embedded data is essential in this area. Historically, steganography have been approached in this area.

Steganography provides us with:

- (A) Potential capability to hide the existence of confidential data
- (B) Hardness of detecting the hidden (i.e., embedded) data
- (C) Strengthening of the secrecy of the encrypted data

2. Protection of data alteration

We take advantage of the fragility of the embedded data in this application area. Actually, embedded data are fragile in most steganography programs. However, this fragility opens a new direction toward an information-alteration protective system such as a "Digital Certificate Document System." The most novel point among others is that "no authentication bureau is needed." If it is implemented, people can send their "digital certificate data" to any place in the world through Internet. No one can forge, alter, nor tamper such certificate data. If forged, altered, or tampered, it is easily detected by the extraction program.

such software are not unified with the target pictures. Each annotation only has a link to the picture. Therefore, when we transfer the pictures to a different album software, all the annotation data are lost. This problem is technically referred to as "Metadata (e.g., annotation data) in a media database system (a photo album software) are separated from the media data (photo data) in the database managing system (DBMS)." This is a big problem.

Steganography can solve this problem because a steganography program unifies two types of data into one by way of embedding operation. So, metadata can easily be transferred from one system to another without hitch.

2.7 STEGANOGRAPHY'S ROLE IN SECURITY

Software used for implementing steganography is new and very effective. Such software enables information to be hidden in graphic, movie and apparently blank media. People who want to communicate secretly often try to hide the fact that any communication at all is taking place. The science of hiding messages is called "steganography", from the Greek words for "covered writing". It includes a vast array of methods of secret communications that conceal the very existence of the message. Among these methods are invisible inks, microdots, character arrangement (other than cryptographic methods of permutation and substitution), digital signatures, covert channels.

2.8 COMPARISION BETWEEN CRYPTOGRAPHY, STEGANOGRAPHY, WATERMARKING AND FINGERPRINTING

Steganography differs from cryptography in the sense that where cryptography focuses on keeping the contents of a message secret, steganography focuses on keeping the existence of a message secret . Steganography and cryptography are both ways to protect information from unwanted parties but neither technology

alone is perfect and can be compromised. Once the presence of hidden information is revealed or even suspected, the purpose of steganography is partly defeated . The strength of steganography can thus be amplified by combining it with cryptography.

Two other technologies that are closely related to steganography are watermarking and fingerprinting . These technologies are mainly concerned with the protection of intellectual property, thus the algorithms have different requirements than steganography.

In watermarking all of the instances of an object are “marked” in the same way. The kind of information hidden in objects when using watermarking is usually a signature to signify origin or ownership for the purpose of copyright protection .

With fingerprinting on the other hand, different, unique marks are embedded in distinct copies of the carrier object that are supplied to different customers. This enables the intellectual property owner to identify customers who break their licensing agreement by supplying the property to third parties. In watermarking and fingerprinting the fact that information is hidden inside the files may be public knowledge – sometimes it may even be visible – while in steganography the imperceptibility of the information is crucial. A successful attack on a steganographic system consists of an adversary observing that there is information hidden inside a file, while a successful attack on a watermarking or fingerprinting system would not be to detect the mark, but to remove it.

2.9 IMAGE STEGANOGRAPHY

Image Steganography has many applications, especially in today’s modern, high-tech world. Privacy and anonymity is a concern for most people on the internet. Image Steganography allows for two parties to communicate secretly and



covertly. It allows for some morally-conscious people to safely whistle blow on internal actions; it allows for copyright protection on digital files using the message as a digital watermark. One of the other main uses for Image Steganography is for the transportation of high-level or top-secret documents between international governments.

Steganography in images has truly come of age with the invention of fast, powerful computers. Software is readily available off the Internet for any user to hide data inside images. These softwares are designed to fight illegal distribution of image documents by stamping some recognisable feature into the image.

A message is embedded in a digital image (cover image) through an embedding algorithm, with the help of a secret key. The resulting stego image is transmitted over a channel to the receiver where it is processed by the extraction algorithm using the same key. During transmission the stego image, it can be monitored by unauthenticated viewers who will only notice the transmission of an image without discovering the existence of the hidden message. Capacity, security, and robustness are the three main aspects affecting steganography and its usefulness. Capacity refers to the amount of data bits.

While Image Steganography has many legitimate uses, it can also be quite nefarious. It can be used by hackers to send viruses and trojans to compromise machines, and also by terrorists and other organizations that rely on covert operations to communicate secretly and safely.

2.10 AES ENCRYPTION ALGORITHM

The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called

ciphertext; decrypting the ciphertext converts the data back into its original form, called plaintext.

The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits. The AES ciphers have been analyzed extensively and are now used worldwide, as was the case with its predecessor, the Data Encryption Standard (DES). AES was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001 after a 5-year standardization process in which fifteen competing designs were presented and evaluated before Rijndael was selected as the most suitable. It became effective as a Federal government standard on May 26, 2002 after approval by the Secretary of Commerce. It is available in many different encryption packages. AES is the first publicly accessible and open cipher approved by the NSA for top secret information. The Rijndael cipher was developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, and submitted by them to the AES selection process. Rijndael (Dutch pronunciation: is a wordplay based upon the names of the two inventors.

DESCRIPTION OF THE CIPHER

AES is based on a design principle known as a Substitution permutation network. It is fast in both software and hardware. Unlike its predecessor, DES, AES does not use a Feistel network. AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits, whereas Rijndael can be specified with block and key sizes in any multiple of 32 bits, with a minimum of 128 bits. The blocksize has a maximum of 256 bits, but the keysize has no theoretical maximum. AES operates on a 4×4 matrix of bytes, termed the state (versions of Rijndael with a larger block size have additional columns in the state). Most AES calculations are done in a special finite field. The AES cipher is specified as a number of repetitions of

transformation rounds that convert the input plaintext into the final output of ciphertext. Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key.

HIGH-LEVEL DESCRIPTION OF THE ALGORITHM

1. KeyExpansion—round keys are derived from the cipher key using Rijndael's key schedule
2. Initial Round
 1. AddRoundKey—each byte of the state is combined with the round key using bitwise xor
3. Rounds
 1. SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
 2. ShiftRows—a transposition step where each row of the state is shifted cyclically a certain number of steps.
 3. MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
 4. AddRoundKey
4. Final Round (no MixColumns)
 1. SubBytes
 2. ShiftRows
 3. AddRoundKey

The SubBytes step

In the SubBytes step, each byte in the matrix is updated using an 8-bit substitution box, the Rijndael S-box. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over $GF(2^8)$, known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points (and so is a derangement), and also any opposite fixed points.

The ShiftRows step

In the ShiftRows step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row. The ShiftRows step operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. For the block of size 128 bits and 192 bits the shifting pattern is the same. In this way, each column of the output state of the ShiftRows step is composed of bytes from each column of the input state. (Rijndael variants with a larger block size have slightly different offsets). In the case of the 256-bit block, the first row is unchanged and the shifting for second, third and fourth row is 1 byte, 3 bytes and 4 bytes respectively - this change only applies for the Rijndael cipher when used with a 256-bit block, as AES does not use 256-bit blocks. Here A_{ij} is from cipher text and B_{ij} is from key.

The MixColumns step

In the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with ShiftRows, MixColumns provides diffusion in the cipher.

The AddRoundKey step

In the AddRoundKey step, the subkey is combined with the state. For each round, a subkey is derived from the main key using Rijndael's key schedule; each subkey is the same size as the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR.

STEPS IN AES ALGORITHM

The algorithm is composed of three main parts: Cipher, Inverse Cipher and Key Expansion. Cipher converts data to an unintelligible form called ciphertext while Inverse Cipher converts data back into its original form called plaintext. Key Expansion generates a Key Schedule that is used in Cipher and Inverse Cipher procedure. Cipher and Inverse Cipher are composed of specific number of rounds. For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key length .

For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations: SubBytes, ShiftRows, MixColumns and AddRoundKey.

Inputs and outputs: The input and output for the AES algorithm each consists of sequences of 128 bits. The Cipher Key for the AES algorithm is a sequence of 128,

192 or 256 bits. The basic unit for processing in the AES algorithm is a byte (a sequence of eight bits), so the input bit sequence is first transformed into byte sequence. In the next step a two-dimensional array of bytes (called the State) is built. The State array consists of four rows of bytes, each containing N_b bytes, where N_b is the block size divided by 32 (number of words). All internal operations (Cipher and Inverse Cipher) of the AES algorithm's are then performed on the State array, after which its final value is copied to the output (State array is transformed back to the bit sequence).

Cipher: Using round function, which is composed of four different byte-oriented transformations, the Cipher converts input data (the input data is first copied to the State array) to an unintelligible form called ciphertext. After an initial Round Key addition, the State array is transformed by implementing a round function with the final round differing slightly from the first $N_r - 1$ rounds.

The round function is parameterized using a key schedule that consists of a one-dimensional array of four-byte words (Round Key) derived using the Key Expansion routine. All N_r rounds are identical with the exception of the final round, which does not include the MixColumns transformation.

Key Schedule: The AES algorithm takes the Cipher Key and performs a Key Expansion routine to generate a Key Schedule. The Key Expansion generates a total $N_b(N_r + 1)$ words ($N_r + 1$ Round Keys).

Inverse Cipher: At the start of the Inverse Cipher, the input (ciphertext) is copied to the State array. After Round Key addition (the last Round Key is added), the State array is transformed by implementing a round function, that is composed of three different inverse transformations and AddRoundKey transformation (Round Keys

are applied in the reverse order when decrypting), with the final round differing slightly from the first $Nr - 1$ rounds. So this procedure converts ciphertext back to its original form called plaintext.

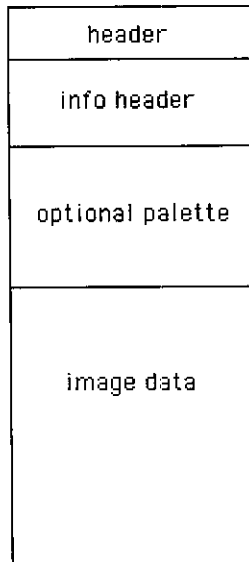
All Nr rounds are identical with the exception of the final round, which does not include the Inverse MixColumns transformation.

2.11 BMP FILE FORMAT

The BMP File Format, also known as Bitmap Image File or Device Independent Bitmap (DIB) file format or simply a Bitmap, is a Raster graphics image file format used to store bitmap digital images, independently of the display device (such as a graphics adapter), especially on Microsoft Windows and OS/2 operating systems.

The BMP File Format is capable of storing 2D digital images of arbitrary width, height, and resolution, both monochrome and color, in various color depths, and optionally with data compression, alpha channels, and color profiles.

A BMP file consists of either 3 or 4 parts as shown in the diagram . The first part is a header, this is followed by a information section, if the image is indexed color then the palette follows, and last of all is the pixel data. The position of the image data with respect to the start of the file is contained in the header. Information such as the image width and height, the type of compression, the number of colors is contained in the information header.



Device-independent bitmaps and BMP file format

Microsoft has defined a particular representation of color bitmaps of different color depths, as an aid to exchanging bitmaps between devices and applications with a variety of internal representations. They called these device-independent bitmaps or DIBs, and the file format for them is called DIB file format or BMP image file format.

According to Microsoft support:

A device-independent bitmap (DIB) is a format used to define device-independent bitmaps in various color resolutions. The main purpose of DIBs is to allow bitmaps to be moved from one device to another (hence, the device-independent part of the name). A DIB is an external format, in contrast to a device-dependent bitmap, which appears in the system as a bitmap object (created by an application...). A DIB is normally transported in metafiles (usually using the StretchDIBits() function), BMP files, and the Clipboard (CF_DIB data format).

Color Table

The number of entries in the palette is either 2^n or a smaller number specified in the header (in the OS/2 BITMAPCOREHEADER header format, only the full-size palette is supported). In most cases, each entry in the color table occupies 4 bytes, in the order Blue, Green, Red, 0x00 (see below for exceptions).

The Color Table is a block of bytes (a table) listing the colors used by the image. Each pixel in an indexed color image is described by a number of bits (1, 4, or 8) which is an index of a single color described by this table. The purpose of the color palette in indexed color bitmaps is to inform the application about the actual color that each of these index values corresponds to. The purpose of the Color Table in non-indexed (non-palettized) bitmaps is to list the colors used by the bitmap for the purposes of optimization on devices with limited color display capability and to facilitate future conversion to different pixel formats and paletization.

The colors in the Color Table are usually specified in the 4-byte per entry 8.8.8.0.8 format (in RGBAX notation). The Color Table used with the OS/2 BITMAPCOREHEADER uses the 3-byte per entry 8.8.8.0.0 format. For DIBs loaded in memory, the Color Table can optionally consist of 2-byte entries - these entries constitute indexes to the currently realized palette instead of explicit RGB color definitions.

Microsoft does not disallow the presence of a valid Alpha channel bit mask in BITMAPV4HEADER and BITMAPV5HEADER for 1bpp, 4bpp and 8bpp indexed color images, which indicates that the Color Table entries can also specify an Alpha component using the 8.8.8.[0-8].[0-8] format via the RGBQUAD.rgb

Reserved member. However, some versions of Microsoft's documentation disallow this feature by stating that the RGBQUAD.rgb Reserved member "must be zero".

As mentioned above, the Color Table is normally not used when the pixels are in the 16-bit per pixel (16bpp) format (and higher); there are normally no Color Table entries in those bitmap image files. However, the Microsoft documentation (on the MSDN web site as of Nov. 16, 2010) specifies that for 16bpp (and higher), the Color Table can be present to store a list of colors intended for optimization on devices with limited color display capability, while it also specifies, that in such cases, no indexed palette entries are present in this Color Table. This may seem like a contradiction if no distinction is made between the mandatory palette entries and the optional color list.

Pixel Format

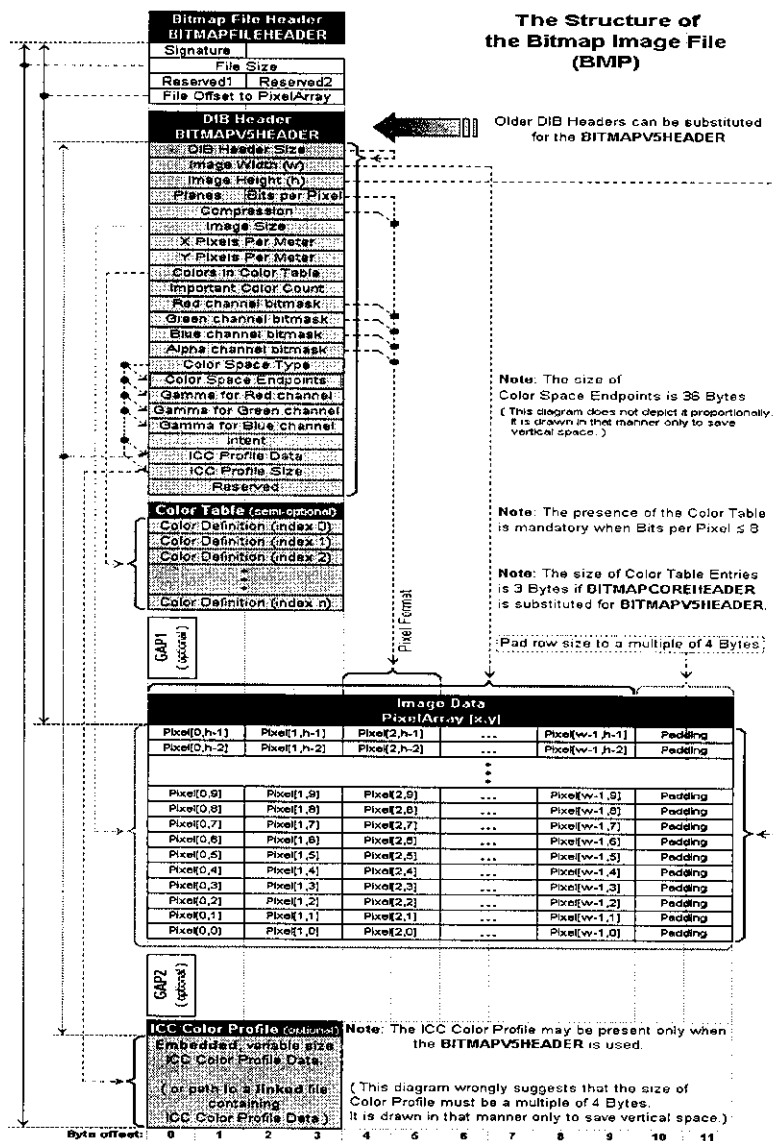
In bitmap image file on a disk or bitmap image in memory, the pixels can be defined by varying number of bits.

- The 1-bit per pixel (1bpp) format supports 2 distinct colors, (for example: black and white, or yellow and pink). The pixel values are stored in each bit, with the first (left-most) pixel in the most-significant bit of the first byte. Each bit is an index into a table of 2 colors. This Color Table is in 32bpp 8.8.8.0.8 RGBAX format. An unset bit will refer to the first color table entry, and a set bit will refer to the last (second) color table entry.
- The 2-bit per pixel (2bpp) format supports 4 distinct colors and stores 4 pixels per 1 byte, the left-most pixel being in the two most

significant bits. Each pixel value is a 2-bit index into a table of up to 4 colors. This Color Table is in 32bpp 8.8.8.0.8 RGBAX format.

- The 4-bit per pixel (4bpp) format supports 16 distinct colors and stores 2 pixels per 1 byte, the left-most pixel being in the more significant nibble. Each pixel value is a 4-bit index into a table of up to 16 colors. This Color Table is in 32bpp 8.8.8.0.8 RGBAX format.
- The 8-bit per pixel (8bpp) format supports 256 distinct colors and stores 1 pixel per 1 byte. Each byte is an index into a table of up to 256 colors. This Color Table is in 32bpp 8.8.8.0.8 RGBAX format.
- The 16-bit per pixel (16bpp) format supports 65536 distinct colors and stores 1 pixel per 2 byte WORD. Each WORD can define the Alpha, Red, Green and Blue samples of the pixel.
- The 24-bit pixel (24bpp) format supports 16,777,216 distinct colors and stores 1 pixel value per 3 bytes. Each pixel value defines the Red, Green and Blue samples of the pixel (8.8.8.0.0 in RGBAX notation). Specifically in the order (Blue, Green and Red, 8-bits per each sample).
- The 32-bit per pixel (32bpp) format supports 4,294,967,296 distinct colors and stores 1 pixel per 4 byte DWORD. Each DWORD can define the Alpha, Red, Green and Blue samples of the pixel.

Windows versions 3.0 and later support run-length encoded (RLE) formats for compressing bitmaps that use 4 bits per pixel and 8 bits per pixel. The default filename extension of a Windows DIB file is .BMP.



2.12 LEAST SIGNIFICANT BIT ALGORITHM

The least significant bit (LSB) algorithm is used in this stego machine to conceal the data in a video file. The main advantage of the LSB coding method is a very high watermark channel bit rate and a low computational complexity. The robustness of the watermark embedded using the LSB coding method, increases with increase of the LSB depth is used for data hiding. In this method, modifications are made to the least significant bits of the carrier file's individual

pixels, thereby encoding hidden data . Here each pixel has room for 3 bits of secret information, one in each RGB values. Using a 24-bit image, it is possible to hide three bits of data in each pixel's color value using a 1024x768 pixel image; also it is possible to hide up to 2,359,296 bits. The human eye cannot easily distinguish 21-bit color from 24-bit color. As a simple example of LSB substitution, imagine "hiding" the character 'A' across the following eight bytes of a carrier file:

(00100111 11101001 11001000)

(00100111 11001000 11101001)

(11001000 00100111 11101001)

Letter 'A' is represented in ASCII format as the binary string 10000011.

These eight bits can be "written" to the LSB of each of the eight carrier bytes as follows (the LSBs are italicized and bolded):

(00100111 1110100**0** 11001000)

(0010011**0** 11001000 1110100**0**)

(1100100**1** 00100111 11101001).

With such a small variation in the colors of the image it would be very difficult for the human eye to discern the difference thus providing high robustness to the system .

Technique basics

Today, when converting an analog image to digital format, we usually choose between three different ways of representing colors:

- 24-bit color: every pixel can have one in 2^{24} colors, and these are represented as different quantities of three basic colors: red (R), green (G), blue (B), given by 8 bits (256 values) each.
 - 8-bit color: every pixel can have one in 256 (2^8) colors, chosen from a palette, or a table of colors.
 - 8-bit gray-scale: every pixel can have one in 256 (2^8) shades of gray.
- LSB insertion modifies the LSBs of each color in 24-bit images, or the LSBs of the 8-bit value for 8-bit images.

Data Rate

The most basic of LSBs insertion for 24-bit pictures inserts 3 bits/pixel. Since every pixel is 24 bits, we can hide

$$3 \text{ hidden-bits/pixel} / 24 \text{ data-bits/pixel} = 1/8 \text{ hidden-bits/data-bits.}$$

So for this case we hide 1 bit of the embedded message for every 8 bits of the cover image. If we pushed the insertion to include the second LSBs, the formula would change to:

$$6 \text{ hidden-bits/pixel} / 24 \text{ data-bits/pixel} = 2/8 \text{ hidden-bits/data-bits}$$

And we would hide 2 bits of the embedded message for every 8 bits of the cover image.

Adding a third-bit insertion, we would get:

$$9 \text{ hidden-bits/pixel} / 24 \text{ data-bits/pixel} = 3/8 \text{ hidden-bits/data-bits}$$

Acquiring a data rate of 3 embedded bits every 8 bits of the image. The data rate for insertion in 8-bit images is analogous to the 1 LSB insertion in 24-bit images, or 1 embedded bit every 8 cover bits. We can see the problem

in another light, and ask how many cover bytes are needed to send an embedded byte.

For 1-LSB insertion in 24-bit images or in 8-bit images this value would be $8/1 \cdot 8 = 8$ Bytes. For 2-LSBs insertion in 24-bit pictures it would be $8/2 \cdot 8 = 4$ Bytes, for 3-LSBs insertion it would be $8/3 \cdot 8 = 21.33$ Bytes.

Robustness

LSB insertion is very vulnerable to a lot of transformations, even the most harmless and usual ones. Lossy compression, e.g. JPEG, is very likely to destroy it completely. The problem is that the "holes" in the Human Visual System that LSB insertion tries to exploit - little sensitivity to added noise - are the same that lossy compression algorithms rely on to be able to reduce the data rate of images. Geometrical transformations, moving the pixels around and especially displacing them from the original grid, are likely to destroy the embedded message, and the only one that could allow recovery is a simple translation. Any other kind of picture transformation, like blurring or other effects, usually will destroy the hidden data. All in all, LSB insertion is a very little robust technique for data hiding.

Ease of detection/extraction

There is no theoretical outstanding mark of LSB insertion, if not a little increase of background noise. It's very easy, instead, to extract LSBs even with simple programs, and to check them later to find if they mean something or not.

Suitability for steganography or watermarking

The two most important issues in this problem are:

- The choice of images
- The choice of the format (24-bit or 8-bit, compressed or not)

The cover image first of all must seem casual, so it must be chosen between a set of subjects that can have a reason to be exchanged between the source and the receiver. Then it must have quite varying colors, it must be "noisy", so that the added noise is going to be covered by the already present one. Wide solid-color areas magnify very much any little amount of noise added to them. Second, there is a problem with the file size, which involves the choice of the format. Unusually big files exchanged between two peers, in fact, are likely to arise suspicion.

To solve this problem, it has been studied a modification to the JPEG algorithm that inserts LSBs in some of the lossless stages or pilots the rounding of the coefficients of the DCT used to compress the image to encode the bits. Since we need to have small image file sizes, we should resort in using 8-bit images if we want to communicate using LSB insertion, because their size is more likely to be considered as normal.

The problem with 256 colors images is that they make use of an indexed palette, and changing a LSB means that we switch a pixel from a position to an adjacent one. If there are adjacent contrasting colors in the palette, it can happen that a pixel in the image changes its color abruptly and the hidden message becomes visible.

To solve this problem different methods have been studied, like rearranging the palette so that adjacent colors don't contrast so much, or even

reducing the palette to a smaller number of colors and replicating the same entry in the table in adjacent positions, so that the difference after the embedding of the message is not visible at all. Moreover for most images the reduction of colors from, for instance, 256 to 32 is hardly visible.

Most of the experts, anyway, advise to use 8-bit grayscale images, since their palette is much less varying than the color one, so LSB insertion is going to be very hard to detect by the human eye.

2.13 ADVANTAGES OF LSB INSERTION

Major advantage of the LSB algorithm is it is quick and easy.

- > There has also been steganography software developed which work around LSB color alterations via palette manipulation.
- > LSB insertion also works well with gray-scale images.
- > A slight variation of this technique allows for embedding the message in two or more of the least significant bits per byte. This increases the hidden information capacity.

2.15 SELECTED LEAST SIGNIFICANT BIT ALGORITHM

Selected Least Significant Bits (SLSB) algorithm works with the least significant bits of one of the pixel color components in the image and changes them according to the message's bits to hide. The rest of bits in the pixel color component selected are also changed in order get the nearest color to the original one in the scale of colors. In theory the three least significant bits of the pixel have changed, introducing a small distortion, but the difference between the old and new color represents a leap of 65793 colors in the scale of colors.

One method that would introduce more efficiency and less distortion would store the 3 bits of information to hide in the same color. Using the same example, the 3 bits of information will be introduced in the 3 LSB bits of green color (10101000-10101111-10101000).

Applying the method proposed here to the above example (in this case, decreasing the 4th least significant bit, which have been used 3 bits LSB to hide information) results in 11000111, with a distance of 1 from the original byte but with the same hidden information.

In this case the leap in the scale of colors is 1792 colors (in the case of changing the color green, if modify the blue color difference would be only 7 colors), that being the extreme case because it has been replaced last 3 bits with 0 value for 3 bits with a 1 value, that is, in most cases the distortion will be much lower.

In order to choose the color for the concealment, the SLSB algorithm performs a preliminary Sample Pairs analysis and select the color with higher ratio because it represents more diversity, leading to less noticeable changes. The choice of Sample Pairs analysis over other stegoanalitics methods is due to the results provided by the work of Ker , where this analysis shows that it is offering better results in terms of detecting hidden information. Thus, the chosen color will be the one that provides greater distortion and, therefore, the result of the withholding of information will be less detectable.

- It is based on the LSB method, but can hide the same information much more effectively using bits of just one color.

- Perform a Sample Pairs analysis prior to steganography, which allows you to select the best color of the three possible to hide information.
- Use a pixel selection filter to obtain the best areas to hide information.
- Implement the LSB Match method to reduce the difference between the original pixel and the steganographic pixel.
- It is immune to visual attacks. Changes are undetectable with the naked eye, and a filter of LSB bits doesn't present areas of random information that could indicate the presence of hidden information.
- It is immune to attacks by comparing histograms, as the frequency of appearance of colors in the steganographic image is very similar to that of the cover image.
- It is immune to statistical attacks, as two colors for each pixel are equal to those of the original image, and the final ratio of analysis is very close to the original image, which doesn't raise suspicion it contains hidden information. Even in some cases get better rates than those of the original image, creating confusion over which of two images would be the original.
- It yields well above that of most steganographic tools used today, both in RS and Sample Pairs analysis and in metric of distortion.

2.16 WHY JAVA

A programming language that allows the user to create programs which run well in a networked environment. Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, and dynamic language and software platform. Java is a network-oriented programming language that is specifically designed for writing programs

that can be safely downloaded to your computer through the Internet and immediately run without fear of viruses or other harm to your computer or files.

PACAKAGES USED FOR SCAIRE:

JAVA.IO

Provides for system input and output through data streams, serialization and the file system.

JAVA.IO.FILE

An abstract representation of file and directory pathnames.

JAVA.IO.BUFFEREDREADER

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

JAVA.IO.FILEINPUTSTREAM

A `FilterInputStream` contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality

JAVA.IO.FILEOUTPUTSTREAM

This class is the superclass of all classes that filter output streams.

JAVA.IO.IOEXCEPTION

Signals that an I/O exception of some sort has occurred.

JAVA.IO.INPUTSTREAM

This abstract class is the superclass of all classes representing an input stream of bytes.

JAVA.IO.INPUTSTREAMREADER

Abstract class for reading character streams.

JAVAX.IMAGEIO.IMAGEIO

A class containing static convenience methods for locating ImageReaders and ImageWriters, and performing simple encoding and decoding.

JAVA.AWT

Contains all of the classes for creating user interfaces and for painting graphics and images.

JAVA.AWT.IMAGE.PIXELGRABBER

The PixelGrabber class implements an ImageConsumer that can be attached to an Image or ImageProducer object to retrieve a subset of the pixels in that image.

PACKAGES USED FOR GUI DEVELOPMENT:

JAVAX.SWING

Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms.

JAVAX.SWING.FILECHOOSER.FILEFILTER

File choosers provide a GUI for navigating the file system, and then either choosing a file or directory from a list or entering the name of a file or directory. To display a file chooser, you usually use the JFileChooser API to show a modal dialogF:\LSB\dialog.html containing the file chooser. Another way to present a file chooser is to add an instance of JFileChooser to a container.

JAVA.AWT.EVENT.ACTIONEVENT

Provides interfaces and classes for dealing with different types of events fired by AWT components.

JAVA.AWT.EVENT.ACTIONLISTENER

The listener interface for receiving action events

JAVA.AWT.EVENT.WINDOWADAPTER

An abstract adapter class for receiving window events.

JAVA.AWT.EVENT.WINDOWEVENT

A low-level event that indicates that a window has changed its status.

JAVA.IO.FILENOTFOUNDEXCEPTION

Signals that an attempt to open the file denoted by a specified pathname has failed.

CHAPTER III

3. SYSTEM REQUIREMENTS AND ANALYSIS

A system requirements analysis takes the system design another step closer toward technical specification. System requirements describe, in a more technical way, the characteristics, components, and capacities of a system that will satisfy the functional requirements.

The purpose of System Requirements Analysis is to obtain a thorough and detailed understanding of the business need as defined in Project Origination and captured in the Business Case, and to break it down into discrete requirements, which are then clearly defined, reviewed and agreed upon with the Customer Decision-Makers. During System Requirements Analysis, the framework for the

application is developed, providing the foundation for all future design and development efforts.

System Requirements Analysis can be a challenging phase, because all of the major Customers and their interests are brought into the process of determining requirements. The quality of the final product is highly dependent on the effectiveness of the requirements identification process. Since the requirements form the basis for all future work on the project, from design and development to testing and documentation, it is of the utmost importance that the Project Team create a complete and accurate representation of all requirements that the system must accommodate.

3.1 FEASIBILITY

3.1.1 FEASIBILITY STUDY

Feasibility studies are preliminary investigations into the potential benefits associated with undertaking a specific activity or project. The main purpose of the feasibility study is to consider all factors associated with the project, and determine if the investment of time and other resources will yield a desirable result. While considered a preliminary study, it is not unusual for a feasibility study to be highly detailed.

When a business is considering a new operation or the launch of a new product, the feasibility study is a logical tool to employ before any resources are invested in the new project. One of the most important aspects of the study is to make sure that the total investment needed to successfully bring the project to completion is considered. Often, this will include addressing components such as cash reserves, labor, construction, production facilities, outsourcing, and the cost of

raw materials. Only when the feasibility study has addressed the total cost of completing the project can the study progress to the next level.

Technology and system feasibility

The assessment is based on an outline design of system requirements in terms of Input, Processes, Output, Fields, Programs, and Procedures. This can be quantified in terms of volumes of data, trends, frequency of updating, etc. in order to estimate whether the new system will perform adequately or not. Technological feasibility is carried out to determine whether the company has the capability, in terms of software, hardware, personnel and expertise, to handle the completion of the project when writing a feasibility report, the following should be taken to consideration:

- A brief description of the business
- The part of the business being examined
- The human and economic factor
- The possible solutions to the problems

At this level, the concern is whether the proposal is both technically and legally feasible (assuming moderate cost).

Economic feasibility

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the

decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action.

Cost-based study: It is important to identify cost and benefit factors, which can be categorized as follows: 1. Development costs; and 2. Operating costs. This is an analysis of the costs to be incurred in the system and the benefits derivable out of the system.

Time-based study: This is an analysis of the time required to achieve a return on investments. The future value of a project is also a factor.

Legal feasibility

Determines whether the proposed system conflicts with legal requirements, e.g. a data processing system must comply with the local Data Protection Acts.

Operational feasibility

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

Schedule feasibility

A project will fail if it takes too long to be completed before it is useful. Typically this means estimating how long the system will take to develop, and if it can be completed in a given time period using some methods like payback period. Schedule feasibility is a measure of how reasonable the project timetable is.

3.2 SPECIFIC REQUIREMENTS

3.2.1 FUNCTIONAL REQUIREMENTS

SOFTWARE SPECIFICATION

Operating System: Windows 7

Language: JAVA

HARDWARE SPECIFICATION

Processor: Intel core2 Duo processor

Hard Disk: 80 GB

System RAM: 3 GB

Processor Speed: 3.02 GHz

3.2.2 PERFORMANCE REQUIREMENTS

INVISIBILITY

The Text File is hidden in such a way that the quality is not degraded and attackers are prevented from finding the contents or distorting it.

SECURITY

The Text File is hidden in Image is previously encrypted using AES algorithm which provides an added Security.

ROBUSTNESS

The Use of Sounds, Images and Videos signals in Digital Form commonly involves many types of distortions, such as lossy compression, or in this case, changes in the contents of the Image file. Steganography tools must ensure that the embedded information is not removed by interception or any other attack.

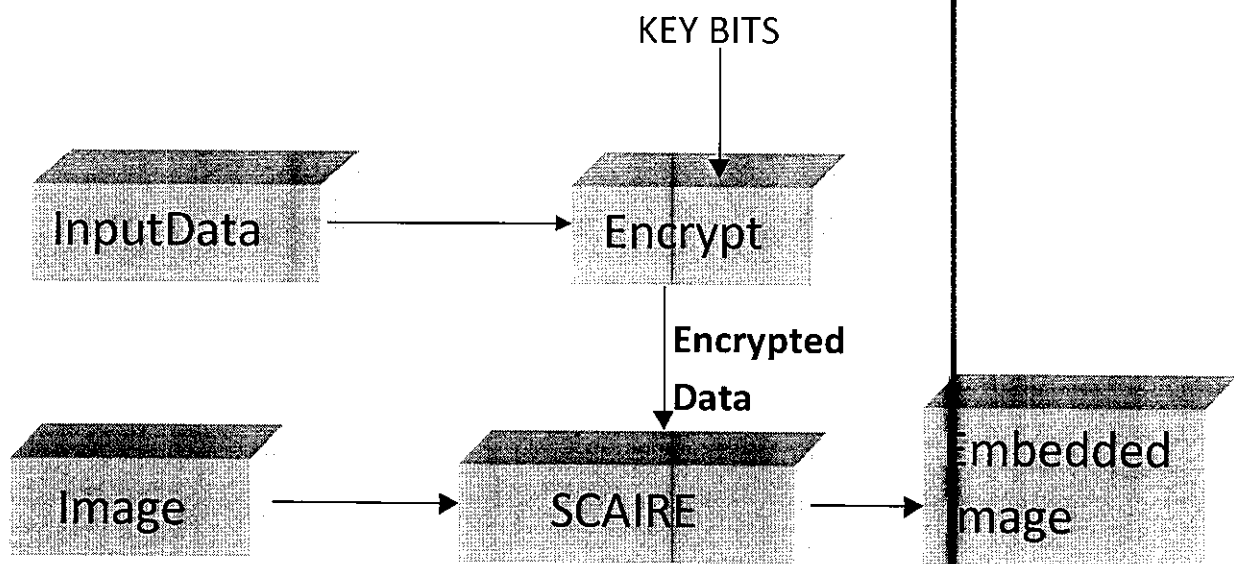
INVERTIBILITY

It is said to be invertible if authorized users can extract the Text file from the cover Image file.

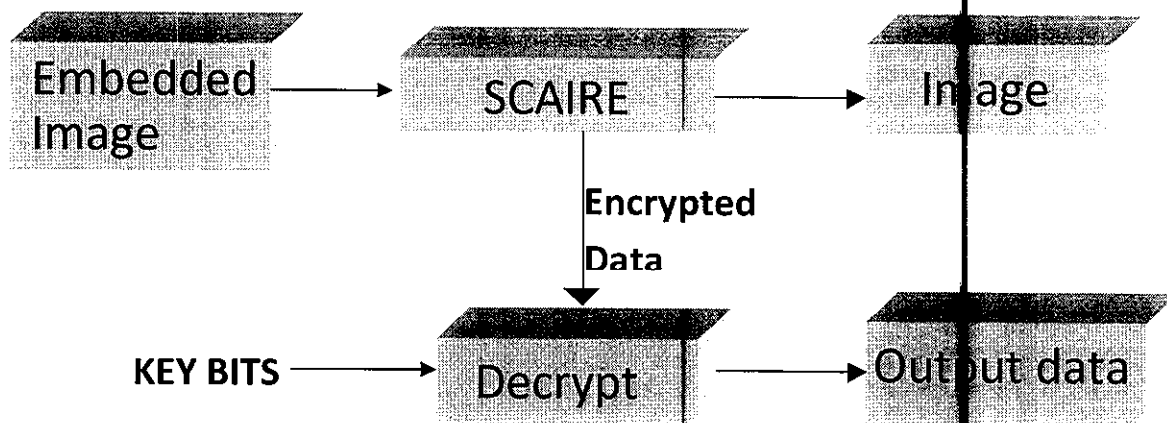
CHAPTER IV

5. SYSTEM STUDY

DATA ENCRYPTION AND HIDING



DATA EXTRACTION AND DECRYPTION



CHAPTER V

5. MODULES

- Data encryption
- Data embedding
- Data extraction
- Data decryption

DATA ENCRYPTION

To provide additional security we have developed an encryption algorithm based on Advanced Encryption Standard Algorithm for data encryption.

DATA EMBEDDING

Data embedding is implemented based on SLSB technique. In this, the data bits are hidden in the last 3 bits of any one color(RGB) of each pixel.

DATA EXTRACTION

Data extraction is the process of retrieving data from Image file and it is the reverse process of hiding. The process of data extraction is similar to that of hiding but instead of considering the carrier file as Image, the embedded data is considered for extraction.

DATA DECRYPTION

Data decryption is done at the receiver's side and displays the hidden data for the user. The Extracted file is then decrypted using AES Decryption Algorithm.

CHAPTER VI

6. CONCLUSION

This project ensures high security for secret message. Since it has combined the effort of cryptography and steganography, it can be called as double security provider.

The Advanced Encryption Standard(AES) is used to encrypt the secret data and the encrypted data is embedded within an image file (using SLSB technique) to ensure that it is not tampered with.

Cryptography itself ensures security to a certain extent. But steganography makes it very difficult for the intruders and crackers to even detect the presence of a secret message.

SCAIRE has combined both these qualities. Hence, it is very difficult to detect the secret data. Even if an intruder detects it, it is not easy to extract and decrypt the message. It is almost impossible to tamper with the secret message and hence security is ensured.

CHAPTER VII

7. FUTURE ENHANCEMENT

In the proposed system the messages to be hidden is data. We can also hide the image files, audio files and the video files inside the Image, Audio and Video. In the case there is voluminous data to be hidden in the cover image; the whole message can be compressed before hiding them in the cover image. Here we use an image as carrier, but we can also use the video files as the carrier.

CHAPTER VIII

8. APPENDIX

8.1 SAMPLE SOURCE CODE

AESP.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.JApplet;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.util.HashMap;
import javax.swing.Icon;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileView;
import java.io.*;
import javax.swing.*;
```

```

import java.net.*;
public class AESP2 extends Applet implements ActionListener, ItemListener
{
    private TextField plText = new TextField(30);
    private TextArea key = new TextArea(4,40);
    private TextArea cipherText = new TextArea(4,40);
    private TextArea decryptText = new TextArea(4,40);
    private Choice bit = new Choice();
    private Button encrypt = new Button("ENCRYPT");
    private Button decrypt = new Button("DECRYPT");
    private Button reset = new Button("RESET");
    private Button fileChooser = new Button("BROWSE");
    private Button fileSave = new Button("BROWSE");
    private Button fileSelect = new Button("BROWSE");
    private Panel centerPanel = new Panel();
    private Panel buttonPanel = new Panel();
    private Panel plPanel = new Panel();
    private Panel pl1Panel = new Panel();
    private Panel pl2Panel = new Panel();
    private Panel pl3Panel = new Panel();
    private Panel kkPanel = new Panel();
    private Panel cpPanel = new Panel();
    private Panel dpPanel = new Panel();
    private Label one = new Label("Plain Text: ");
    private Label two = new Label("Key Bits: ");
    private Label three = new Label("Cipher Text: ");

```

```

private Label four = new Label("Decrypt Text: ");
private Label note1 = new Label("Enter the Key bits below in 0's & 1's");
private String pltxt, kk, cptxt, instr, dptxt;
private int bits, dips;
public AES Ins;
public void init() {
    centerPanel.setLayout(new GridLayout (5, 1, 1, 1));
    initplPanel();
    centerPanel.add(plPanel);
    initkkPanel();
    centerPanel.add(kkPanel);
    initcpPanel();
    centerPanel.add(cpPanel);
    initdpPanel();
    centerPanel.add(dpPanel);
    initInnerPanel();
    centerPanel.add(buttonPanel);
    add(centerPanel);
    setBackground(Color.darkGray);
}
public void initInnerPanel() {
    bit.add("128");
    bit.add("192");
    bit.add("256");
    buttonPanel.add(bit);
    buttonPanel.add(encrypt);
}

```

```

        buttonPanel.add(decrypt);
        buttonPanel.add(reset);
        encrypt.addActionListener(this);
        decrypt.addActionListener(this);
        reset.addActionListener(this);
        bit.addItemListener(this);
    }
    public void initplPanel() {
        plPanel.setLayout(new GridLayout (2, 1, 1, 1));
        pl1Panel.add(one);
        pl1Panel.add(plText);
        pl1Panel.add(fileChooser);
        plText.setEditable(true);
        plText.addActionListener(this);
        fileChooser.addActionListener(this);
        plText.setBackground(Color.lightGray);
        pl2Panel.add(note1);
        plPanel.add(pl1Panel);
        plPanel.add(pl2Panel);
        key.setText("Enter the Key Bits in 0's & 1's only");
    } // initplPanel()
    public void initkkPanel() {
        kkPanel.add(two);
        kkPanel.add(key);
        key.setEditable(true);
        key.setBackground(Color.lightGray);
    }

```

```

} // initkkPanel()

public void initcpPanel() {
    cpPanel.add(three);
    cpPanel.add(cipherText);
    cipherText.setEditable(true);
    cipherText.setBackground(Color.lightGray);
    cpPanel.add(fileSave);
    fileSave.addActionListener(this);
} // initcpPanel()

public void initdpPanel() {
    dpPanel.add(four);
    dpPanel.add(decryptText);
    decryptText.setEditable(true);
    decryptText.setBackground(Color.lightGray);
    dpPanel.add(fileSelect);
    fileSelect.addActionListener(this);
} // initdpPanel()

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == encrypt) {
        pltxt = readFile(plText.getText());
        pltxt.replace((char) 13, '_');
        System.out.println("PLTXT:"+pltxt+"Length:"+pltxt.length());
        bits = Integer.parseInt(bit.getSelectedItemAt());
        kk = key.getText();
    }
}

```

```

if((dips = kk.length())<bits){
for(int i=0;i<bits-dips;i++)
kk = kk + "0";}
Ins = new AES(bits/32);
cptxt = Ins.encrypt(pltxt,kk,bits);
try {
String str = cipherText.getText();
if(str.equals("")){
JOptionPane.showMessageDialog(null,"Please enter the file name!");
}
else{
File f = new File(str);
if(f.exists()){
BufferedWriter out = new BufferedWriter(new FileWriter(f,true));
if(Ins.encrypt(pltxt,kk,bits).equals("")){
JOptionPane.showMessageDialog(null,"Please enter your text!");
}
else{
out.write(Ins.encrypt(pltxt,kk,bits));
if(f.canWrite()){
JOptionPane.showMessageDialog(null,"Text is written in "+str);
}
else{
JOptionPane.showMessageDialog(null,"Text isn't written in "+str);
}
out.close();
}
}
}

```

```

    }
    }
    else{
String text = Ins.encrypt(pltxt,kk,bits);
byte b[] = text.getBytes();
File outputFile = new File(str);
FileOutputStream out = new FileOutputStream(outputFile);
out.write(b);
out.close();
    }
    }
}
catch(Exception x){
    x.printStackTrace();
    }
} else if (e.getSource() == decrypt) {
    cptxt = readFile(decryptText.getText());
    kk = key.getText();
    bits = Integer.parseInt(bit.getSelectedText());
    if((dips = kk.length())<bits){
        for(int i=0;i<bits-dips;i++)
            kk = kk + "0";}
    Ins = new AES(bits/32);
    dptxt = Ins.decrypt(cptxt,kk,bits);
    decryptText.setText(dptxt);
} else if (e.getSource() == reset) {

```

```

        plText.setText("");
        cipherText.setText("");
        decryptText.setText("");
        key.setText("");
    } else if (e.getSource() == plText) {
        pltxt = plText.getText();
        pltxt.replace((char) 13, '_');
        bits = Integer.parseInt(bit.getSelectedItemAt());
        kk = key.getText();
        if((dips = kk.length()) < bits){
            for(int i=0; i < bits-dips; i++)
                kk = kk + "0";
        }
        Ins = new AES(bits/32);
        cptxt = Ins.encrypt(pltxt, kk, bits);
        cipherText.setText(cptxt);
    } else if (e.getSource() == fileChooser) {
        JFileChooser chooser = new JFileChooser(".");
        FileFilter type1 = new ExtensionFilter("Java source", ".java");
        FileFilter type2 = new ExtensionFilter("Image files", new
String[] { ".jpg", ".gif", ".jpeg", ".xbm" });
        FileFilter type3 = new ExtensionFilter("HTML files", new
String[] { ".htm", ".html" });
        FileFilter type4 = new ExtensionFilter("Text files", ".txt");
        chooser.addChoosableFileFilter(type1);
        chooser.addChoosableFileFilter(type2);
    }
}

```



```

chooser.addChoosableFileFilter(type3);
chooser.addChoosableFileFilter(type4);
chooser.setFileFilter(type4); // Initial filter setting
FileView view = new IconView();
chooser.setFileView(view);
int status = chooser.showOpenDialog(AESP2.this);
if (status == JFileChooser.APPROVE_OPTION) {
    File f = chooser.getSelectedFile();
    String ni = f.getPath();
    plText.setText(ni);
}
} else if (e.getSource() == fileSave) {

    JFileChooser chooser = new JFileChooser(".");
    FileFilter type1 = new ExtensionFilter("Java source", ".java");
    FileFilter type2 = new ExtensionFilter("Image files",new
String[] { ".jpg", ".gif", ".jpeg", ".xbm" });
    FileFilter type3 = new ExtensionFilter("HTML files",new
String[] { ".htm", ".html" });
    FileFilter type4 = new ExtensionFilter("Text files", ".txt");
    chooser.addChoosableFileFilter(type1);
    chooser.addChoosableFileFilter(type2);
    chooser.addChoosableFileFilter(type3);
    chooser.addChoosableFileFilter(type4);
    chooser.setFileFilter(type4); // Initial filter setting
    FileView view = new IconView();

```



P-3616

```
chooser.setFileView(view);
int status = chooser.showSaveDialog(AESP2.this);
if (status == JFileChooser.APPROVE_OPTION) {
    File f = chooser.getSelectedFile();
    String ni = f.getPath();
    cipherText.setText(ni);
}

} else if (e.getSource() == fileSelect) {
    JFileChooser chooser = new JFileChooser(".");
    FileFilter type1 = new ExtensionFilter("Java source", ".java");
    FileFilter type2 = new ExtensionFilter("Image files", new
String[] { ".jpg", ".gif", ".jpeg", ".xbm" });
    FileFilter type3 = new ExtensionFilter("HTML files", new
String[] { ".htm", ".html" });
    FileFilter type4 = new ExtensionFilter("Text files", ".txt");
    chooser.addChoosableFileFilter(type1);
    chooser.addChoosableFileFilter(type2);
    chooser.addChoosableFileFilter(type3);
    chooser.addChoosableFileFilter(type4);
    chooser.setFileFilter(type4); // Initial filter setting
    FileView view = new IconView();
    chooser.setFileView(view);
    int status = chooser.showOpenDialog(AESP2.this);
    if (status == JFileChooser.APPROVE_OPTION) {
        File f = chooser.getSelectedFile();
```

```

        String ni = f.getPath();
        decryptText.setText(ni);
    }
} // actionPerformed()
public void itemStateChanged(ItemEvent ie) {
    bits = Integer.parseInt(bit.getSelectedItem());
} // itemStateChanged()
class AnOvalIcon implements Icon {
    Color color;
    public AnOvalIcon(Color c) {
        color = c;
    }
    public void paintIcon(Component c, Graphics g, int x, int y) {
        g.setColor(color);
        g.fillOval(x, y, getIconWidth(), getIconHeight());
    }
    public int getIconWidth() {
        return 10;
    }
    public int getIconHeight() {
        return 15;
    }
}
public class IconView extends FileView {
    private HashMap hash = new HashMap();

```

```

public IconView() {
    hash.put("htm", new AnOvalIcon(Color.RED));
    hash.put("html", new AnOvalIcon(Color.GREEN));
    hash.put("java", new AnOvalIcon(Color.BLUE));
}

public String getName(File f) {
    String s = f.getName();
    if (s.length() == 0) {
        s = f.getAbsolutePath();}
    return s;
}

public String getDescription(File f) {
    return f.getName();
}

public String getTypeDescription(File f) {
    return f.getAbsolutePath();
}

public Icon getIcon(File f) {
    String path = f.getAbsolutePath();
    int pos = path.lastIndexOf('.');
    if ((pos >= 0) && (pos < (path.length() - 1))) {
        String ext = path.substring(pos + 1).toLowerCase();
        return (Icon) hash.get(ext);
    }
    return null;
}

```

```

public Boolean isTraversable(File file) {
    return (new Boolean(file.isDirectory()));
}
}

public class ExtensionFilter extends FileFilter {
    private String extensions[];
    private String description;
    public ExtensionFilter(String description, String extension) {
        this(description, new String[] { extension });
    }
    public ExtensionFilter(String description, String extensions[]) {
        this.description = description;
        this.extensions = (String[]) extensions.clone();
    }
    public boolean accept(File file) {
        if (file.isDirectory()) {
            return true;
        }
        int count = extensions.length;
        String path = file.getAbsolutePath();
        for (int i = 0; i < count; i++) {
            String ext = extensions[i];
            if (path.endsWith(ext)
                && (path.charAt(path.length() - ext.length()) == '.')) {
                return true;
            }
        }
    }
}

```

```

    }
    return false;
}
public String getDescription() {
    return (description == null ? extensions[0] : description);
}
}
public String readFile(String filename){
    StringBuffer sb = new StringBuffer();
    try{
        String lineSep = System.getProperty("line.separator");
        BufferedReader br = new BufferedReader(new
FileReader(filename));
        String nextLine = "";
        while ((nextLine = br.readLine()) != null) {
            sb.append(nextLine);
            sb.append(lineSep);
        }
    }
    catch(IOException e){}
    return sb.toString();
}
} // AESP

```

STEG.java

```
package newpackage;
import javax.swing.*;
import javax.swing.filechooser.FileFilter;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import javax.swing.WindowConstants.*;
public class steg extends javax.swing.JFrame {
    public int t1;
    int f=0;
    public steg(int choice) {
        initComponents(choice);
    }
    private void initComponents(int t) {
        t1=t;
        //jRadioButton1 = new javax.swing.JRadioButton();
        jRadioButton2 = new javax.swing.JRadioButton();
        jRadioButton3 = new javax.swing.JRadioButton();
        jRadioButton4 = new javax.swing.JRadioButton();
```

```

jRadioButton5 = new javax.swing.JRadioButton();
ButtonGroup buttonGroup = new ButtonGroup();
ButtonGroup buttonGroup1 = new ButtonGroup();
buttonGroup.add(jRadioButton1);
buttonGroup.add(jRadioButton3);
buttonGroup1.add(jRadioButton2);
buttonGroup1.add(jRadioButton4);
buttonGroup1.add(jRadioButton5);
    jRadioButton2.setEnabled(false);
    jRadioButton4.setEnabled(false);
    jRadioButton5.setEnabled(false);
jLabel1 = new javax.swing.JLabel();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
getContentPane().setLayout(null);
jRadioButton3.setFont(new java.awt.Font("Franklin Gothic Demi Cond", 1,
14));
jRadioButton3.setText("IMAGE");
jRadioButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton3ActionPerformed(evt);
    if(jRadioButton3.isSelected())
        {
            jRadioButton4.setEnabled(true);
            jRadioButton5.setEnabled(true);
            jRadioButton2.setEnabled(false);

```



```

        }
        else
        {
            jRadioButton4.setEnabled(false);
            jRadioButton5.setEnabled(false);
            jRadioButton2.setEnabled(true);
        }
    }
});
getContentPane().add(jRadioButton3);
jRadioButton3.setBounds(180, 170, 80, 25);

jRadioButton4.setFont(new java.awt.Font("Franklin Gothic Demi Cond", 1,
14));
jRadioButton4.setText("BMP");
jRadioButton4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton4ActionPerformed(evt);
    }
});
getContentPane().add(jRadioButton4);
jRadioButton4.setBounds(230, 220, 80, 25);
jLabel1.setFont(new java.awt.Font("Magneto", 1, 24));
jLabel1.setForeground(new java.awt.Color(0, 0, 255));
jLabel1.setText("Image Stegnography");
getContentPane().add(jLabel1);

```

```

jLabel1.setBounds(100, 10, 330, 30);
jButton1.setFont(new java.awt.Font("Arial", 0, 11));
jButton1.setText("OK");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});
if(jRadioButton3.isSelected())
{
if(jRadioButton2.isSelected())
    {
        f=1;
    }
    else if(jRadioButton4.isSelected())
    {
        f=2;
    }
    else if(jRadioButton5.isSelected())
    {
        f=3;
    }
}
if((jRadioButton2.isSelected())||(jRadioButton4.isSelected())||(jRadioButton5.isSelected()))
{
    if(t1==1)
    {
        StegoInput stegip = new StegoInput(f);
    }
}

```

```

        stegip.setVisible(true);
        setVisible(false);
    }
    else if(t1==2)
    {
        DestegoInput destegip = new DestegoInput(f);
        destegip.setVisible(true);
        setVisible(false);
    }
}
else{JOptionPane.showMessageDialog(null,"Select any one option","E R R O R
",1); }
}
else
{
JOptionPane.showMessageDialog(null,"Select any one option","E R R O R ",1);
}
});
getContentPane().add(jButton1);
jButton1.setBounds(110, 330, 80, 25);
jButton2.setFont(new java.awt.Font("Arial", 0, 11));
jButton2.setText("CANCEL");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});
setVisible(false);

```

```

Intf a=new Intf();
a.setVisible(true);
    }
});
getContentPane().add(jButton2);
jButton2.setBounds(260, 330, 90, 23);
    setBounds(120,50,500,500);
    setTitle("SCAIRE");
setResizable(false);
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
}
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

```

```

}
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new steg(1).setVisible(true);
        }
    });
}
// Variables declaration
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JLabel jLabel1;
private javax.swing.JRadioButton jButton1;
private javax.swing.JRadioButton jButton2;
private javax.swing.JRadioButton jButton3;
private javax.swing.JRadioButton jButton4;
private javax.swing.JRadioButton jButton5;
// End of variables declaration
}

```

8.2 SAMPLE OUTPUT

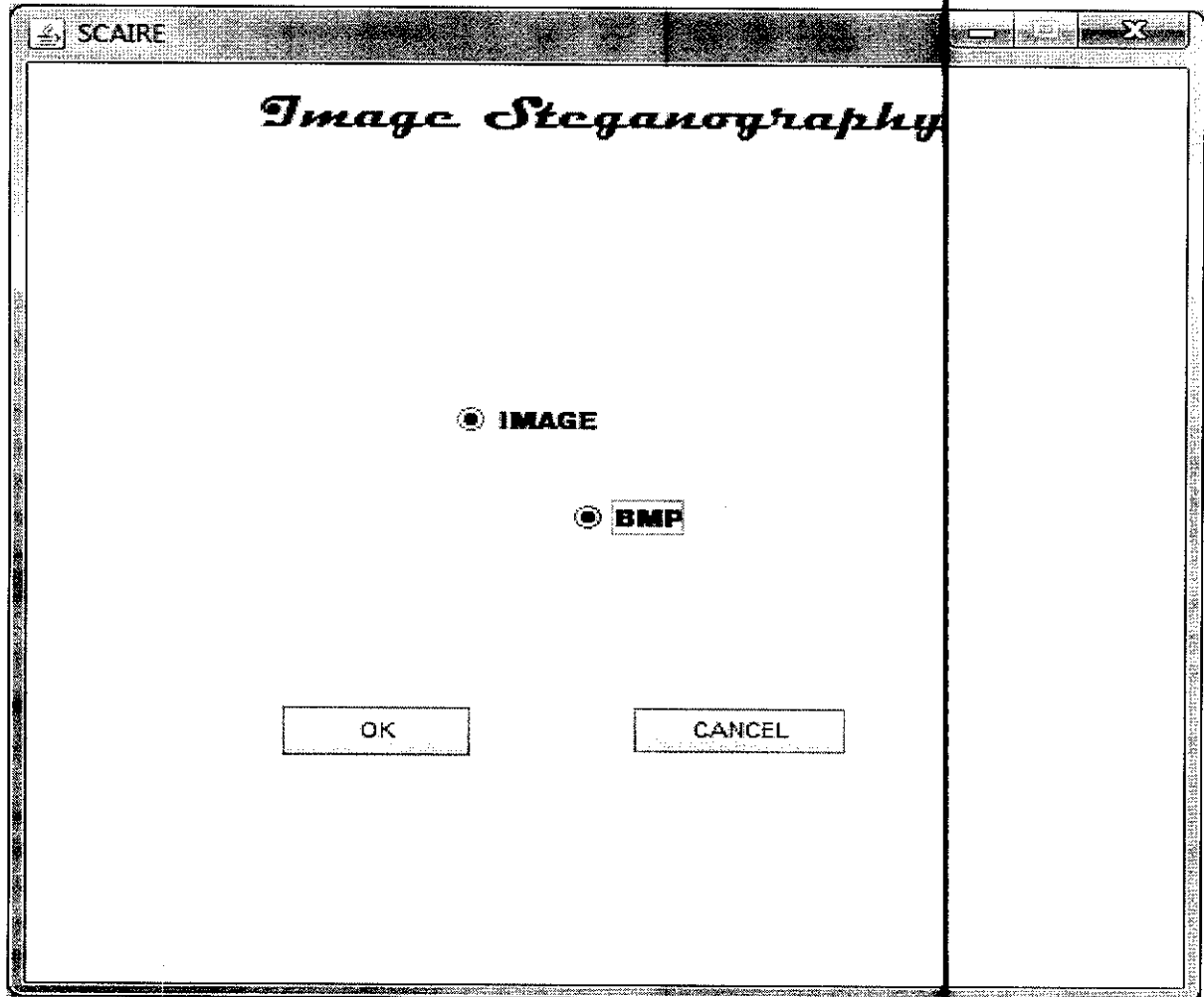
DATA ENCRYPTION AND DATA DECRYPTION

The screenshot shows a Java Applet Viewer window titled "Applet Viewer: enc/AESP2.jar". The applet interface includes the following elements:

- Plain Text:** A text field containing "D:\java\New folder (2)\content5.txt" with a "BROWSE" button to its right.
- Key Input:** A label "Enter the Key bits below in 0's & 1's" above a "Key Bits" text field containing "011111".
- Cipher Text:** A label "Cipher Text" next to a text field containing "D:\java\New folder (2)\Encrypt.txt" with a "BROWSE" button to its right.
- Decrypt Text:** A label "Decrypt Text" next to a text field containing "My Name is Swan And Sevan. I'm Working in Kumaraguru College of technology. My Call" with a "BROWSE" button to its right.
- Controls:** A dropdown menu showing "128", and three buttons: "ENCRYPT", "DECRYPT", and "RESET".

At the bottom left of the applet area, the text "Applet started." is visible. The Windows taskbar at the bottom shows the system tray with the time "22:54" and date "09-04-2011".

DATA HIDING



SCAIRE

Stego Input

Select the data file to be hidden:

Enter the password:

Select the file in which the data is to be hidden:

Select the location to save the file:

Data File size: 4867 Bytes
Total size needed: 38936 Bytes
Carrier File size: 3932214 Bytes
Steganography Possible

DATA EXTRACTION

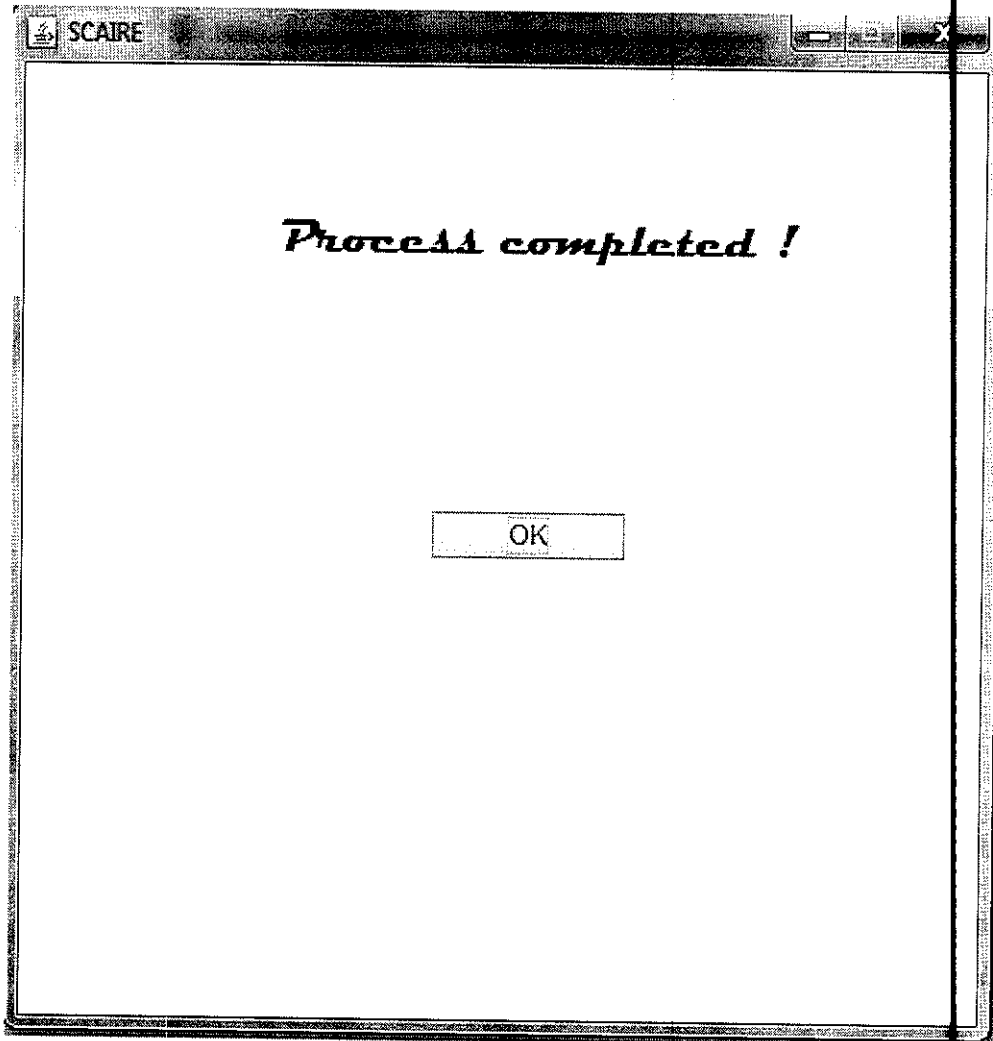
SCAIRE

Destego Input

Select the file to extract the data

Select the location to save the data

Enter the Password



REFERENCES

1. http://en.wikipedia.org/wiki/Advanced_Encryption_Standard
2. Moskowitz, I., Longdon G. and Chang, L.: A New Paradigm Hidden in Steganography.
3. Ker, A.: Improved detection of LSB steganography in grayscale images.
4. Pautric.Naughton, Herbert Schildt(1999) “ The Complete Reference JAVA 2 Third Edition ” , Tata McGraw-Hill Publishing Company Limited.
5. Deitel and Deitel(1999) ,“ JAVA How To Program” ,Pearson Education Inc.
6. Philip Heller and Simon Roberts (1997),”Java Developers HandBook”, BPB Publications.
7. Kurak, C. and McHugh, J.: A Cautionary Note on Image Downgrading. Proc. IEEE 8th Annual Computer Security Applications Conference. San Antonio, USA, Nov./Dec. 1992, pp. 153-155.
8. Juan José Roque, Jesús María Minguet: SLSB: Improving the Steganographic Algorithm LSB