5-2018

Security during Transmission of Data Using Web Steganography

Ahmed Uz Zaman
St. Cloud State University, ahmedbakhtiyar56@gmail.com

Security during Transmission of Data Using Web Steganography

by

Ahmed Bakhtiyar Uz Zaman

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science

in Information Assurance

May, 2018

Starred Paper Committee:
Susantha Herath, Chairperson
Dien D. Phan
Balasubramanian Kasi

Abstract

The project entitled "Steganography" is to give security to a content record. Since the security of the data over the internet has raised a concern to the people. There are many methods to protect the data from going into the access of unauthorized people. Steganography can be used along with the encryption technique to secure the data. Steganography is used to hide the data or a secret message whereas cryptography is used to encrypt the message and make it difficult the people to read. So, the proposed system is to combine both steganography and cryptography for the secret data transmission. The transmission can be done by using an image as a carrier of data. This paper uses high-performance BMP steganography along with a substitution encryption methodology. The approach that is used here is IDEA (International Data Encryption Algorithm) algorithm which is used for encryption. The IDEA algorithm works as follows, it will take the TEXT document and mystery key as the input and gives the encrypted and BMP picture as the output for the sender side. There can additionally be “Voice Recognition System” framework so that it can use voice to decrypt the message. This is the future expansion or scope of this paper.

Table of Contents

Chapter Page

Chapter Page

List of Tables

List of Figures

Figure Page

Chapter I: Introduction

Introduction

Data hiding techniques have been used widely for the transmission of data over the web for a long time. They are classified into two types: Watermarking and Steganography. Steganography comes from the Greek words “steganos” and “graptos” meaning covering and writing respectively. It is the art of embedding a message that is to be hidden in a medium, usually a picture, an audio file, or a video file, in such a way that no one apart from the sender and the intended recipient even realizes that there is a hidden message.

To hide the information using text, images, audio, and videos for carrying secret communication is the best way to use digital media. The major concern due to the increase of data communication over the computer network is the security of information (Laskar & Hemachandran, 2012a). Thus, data confidentiality and integrity are used to protect the data from unauthorized access.

Problem Statement

Data security is the main threat to the organizations worldwide. There is no such security system that combines two different security techniques for providing security to a file. There needs to be a system to ensure the data confidentiality and integrity while transferring data over the internet.

Nature and Significance of the Problem

The current encryption system used comprises of no security using encryption. In the present framework, the record information is being transmitted through the system without any encryption of that file.

According to Watters, Martin, and Stripf (2008), there is no privacy or security to the text file document that transmits the data. There are many threats while transferring a file over the internet, encrypting a document makes it secure. The security of a file like a text document is very necessary for any organization as there are many secret files that are to be sent over the internet to the clients or be it organizations. So, providing security to a document is very important in this scenario as the personal and confidential information of anyone can be compromised.

Data privacy issues can arise from a wide range of sources such as health care records, criminal justice investigations and proceedings, financial institutions and transactions, biological traits, residence and geographic records and ethnicity. Data security or data privacy has become increasingly important as more and more systems are connected to the Internet. There are information privacy laws that cover the protection of data or information on private individuals from intentional or unintentional disclosure or misuse. Thus, hiding the data in a kind of form such as within an image is vital to make sure that security or privacy of the important data is protected.

The encryption of a file is the secure way to do that, but it cannot completely rely on encryption as there is a slight probability that a file can be compromised when it uses encryption. Since the encryption has an encryption key that encrypts the file. When that key is known to the attacker along with the encryption technique used then he can easily access the files and read them. To protect the files from such a huge risk, there is another technique that can be used and i.e. steganography.

Steganographic systems embed hidden messages inside the least significant bit layers of color natural images. The presence of these messages can be difficult to detect by using statistical steganalysis.

However, visual steganalysis by humans may be more successful in natural image discrimination. Steganography as widely famous for its nature of hiding the information of a file inside another file can be used as an alternative. Steganography is not even easily recognizable to anyone. One should at first know that there is steganography used on that file to decode the file. Steganography can hide a text message into an image without changing the nature, quality, and specification of the image.

All digital files like images, videos, text, multimedia can be used in steganography to hide the data, but the format with a high amount of redundancy are more useful. “The redundant bits of an object are those bits that can be altered, but the alteration cannot be detected” (Khare, Singh, & Tiwari, 2011). As images have high amount or volume of redundant bits, they are best suitable for steganography. Li, He, Huang, and Shi (2011) said that using an image easy because an image is an array of pixels, so it contains an enormous amount of redundant information. An image has different light intensities at different areas of the image, as it is a collection of numbers. There are many ways to hide messages within images. The security of stego-images (steganographic images) depends completely on their capability to go unseen.

Digital images are too large to be transmitted over the internet. So, there are techniques to reduce the size of the image and to decrease the display time of the image. This kind of techniques is called compression and they use mathematical

formulas to reduce the image size and providing smaller size images. Choice of cover image is a key factor of steganographic techniques.

The image formats are divided into two types the lossy and lossless based upon their compression. They both save storage but different results.

Lossless compression reconstructs the original message exactly and thus it is preferred when the original information must remain intact (Kaur, Gupta, Sandhu, & Kaur, 2010, quoted in Khare et al., 2011). They also maintain the integrity of the image data. But they have low compression ratio when compared to lossy formats. Lossy compression saves the space, but they do not maintain the integrity of the image. Khare et al. (2011), and Watson (1994) also proved that the positive side of the lossy images is that they achieve extremely high compression when in JPEG/BMP format while maintaining the excellent quality.

According to Laskar and Hemachandran (2012b) using of BMP images in steganography was thought to be not possible as lossy compression previously as it would reduce the bits of the image and data may be lost. The major characteristic of steganography is that information is hidden in the redundant bits of an image and since the redundant bits are left out when using the JPEG/BMP it was feared that the hidden message would be destroyed (Walia, Jain & Navdeep, 2010). In 2008, Liu and Liao mention that the properties of the compression algorithm are re-designed to develop a steganographic algorithm for BMP.

According to Kharrazi, Sencar, and Memon (2006), it is not visible to a human eye that the image has been changed. In image-based steganography, the lossy compression is preferred as it achieves high compression when compared to lossless

compression, and thus is more secure than lossless. It also has fewer chances of detection than that of lossless.

Steganography is not just embedding the secret message into the digital image but also the intended receiver should also know the technique and method used to retrieve the message successfully. The receiver should be able to retrieve the message without drawing any attention from an unauthorized person or without anyone realizing that a secret communication is going on.

Objective of the Research

The main objective of this study is to ensure the data confidentiality and integrity along with the safe transmission of data over the web. By combining cryptography and steganography it can develop a system that can ensure both confidentiality and integrity of the data. Steganography can be used to hide the actual message without altering it and cryptography can be used to give the security layer to that message.

Study Questions/Hypotheses

1. Is Steganography the best and reliable technique to ensure the security of the secret message without any alterations to the message?
2. Can the combination of both steganography and cryptography be able to create a secure and efficient way to transfer data over the web?

Summary

The introduction gave a wide view of what the paper is all about. Using the techniques of steganography and cryptography combined to make a system for secure transmission of data. The next phase of this paper is about what is the media used for application of these technologies. Here the media used is a BMP image which is going

to be encrypted and steganography is going to be applied to the image by varying the pixel colors. In next chapter, there will a literature review on the background of the problem and the literature on the methodology that is going to be used in this paper.

Chapter II: Background and Review of Literature

Introduction

Data is the main part of computer communication and universal economy. In 2003, Conway stated that to ensure the security and integrity of the data, the data hiding concept using the steganography has attracted people used to come up with some creative ideas to protect the data from going into the wrong hands. In this report, the importance and method of application of steganography and encryption to a data can be seen. Anderson and Peticolas (1998) mentioned that digital data can be delivered over computer networks from one place to another without any errors and interference. The copy of digital data can be made without any loss in quality and content.

Thus, it poses a big problem for the security of data and protection of intellectual property rights of copyright owners (Petitcolas, Anderson, & Kuhn, 1999). The Internet is a widely-used method of communication to distribute information to the people. Because of spreading of the Internet around the world, the motivation of hiding a secret message in different multimedia and secure communication via the Internet is increased (Hosmer, 2006).

Background Related to the Problem

Steganography and cryptography are the two different techniques of hiding information which provide confidentiality and integrity of data (Raphael & Sundaram, 2011). Li et al. (2011) also stated that steganography technique aims to transmit a message on a channel, where some other kind of information is already being transmitted. The goal of steganography is to hide messages inside other digital media in

a way that does not allow any person to even detect the presence of secret message as written in “Secure data transmission using steganography and encryption” (Laskar & Hemachandran, 2012b).

Johnson and Jajodia (1998) suggest that the main goal of steganography is to communicate securely in such a way as to avoid drawing suspicion to the transmission of a hidden data. Cryptography hides the contents of a secret message from an unauthorized person, but the content of the message is visible (Laskar & Hemachandran, 2012b). In1994, Simmons proved that in cryptography, the structure of a message is scrambled in such a way as to make it meaningless and unintelligible manner. According to Anderson (1989), cryptography offers the ability to transmit information between persons in a way that it prevents a third party from reading it.

The Data Encryption Standard (DES) algorithm has been a popular secret key encryption algorithm and is used in many commercial and financial applications. Although introduced in 1976, it has proved resistant to all forms of cryptanalysis. However, its key size is too small by current standards and its entire 56-bit key space can be searched in approximately 22 hours (Sharma & Kumar, 2013). The Cryptographic research community is in a continuous process of developing new ways of securing information and publishing attacks showing the weaknesses of existing algorithms (Bozesan, Opritoiu & Vladutiu, 2013).

Some steganographic utilities use secret keys. In 1996, Pfitzmann compared that there are two kinds of keys, i.e., steganographic keys and cryptographic keys. A steganographic key controls the embedding and extracting process. For example, it can

scatter the message to be embedded over a subset of all suitable places in the carrier medium.

Without the key, this subset is unknown, and each sample used to detect embedding by a statistical attack is a mixture of used and unused places (i.e., of all potential places) which spoils the result. A cryptographic key is however used to encrypt the message before it is embedded. For both applications, the “secret”, which conceals the message, is detached from the actual algorithm in the form of a parameter the key.

Image steganography is the science of hiding secret messages inside of images. Think of it as 21st-century disappearing ink. The casual observer simply sees an ordinary image; only someone who knows how to look for it will notice or find the message.

A user will implement a simple, but very effective form of image steganography. The idea is to fiddle with each pixel's color in a way that is not perceivable to the human eye, but that the computer can interpret. Since the human eye is least sensitive to blue wavelengths, it'll slightly adjust the amount of blue in each pixel in a way that encodes a single bit of information.

As far as the computer is concerned, an image is just a 2-D array of pixels. The color of each pixel has encoded an integer between 0 and 16.8 million (2^{24} -1 to be precise), with 8 bits dedicated to each of the red, green, and blue primaries.

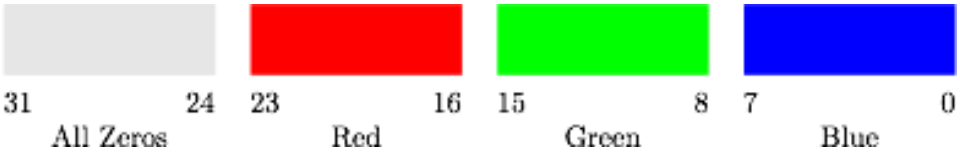

Figure 2.1: Color comparison of the different bits of an image.

Flipping the one's bit of this number (i.e., subtracting 1 from an odd number or adding 1 to an even number) changes the amount of blue in the color by a minute amount that is indistinguishable to the human eye.

Image steganography will represent the information to be hidden as a sequence of bits, and then hide that sequence of bits in the image. It will assume the information to be hidden is made up of (English) letters, digits, and punctuation marks, represented as bits according to the ASCII character set. Then, it will simply set the least significant bit (LSB) of each successive pixel to 0 or 1 to match the values of each bit in the message. This will cause the blue value of each pixel to change by at most one step, which won't be noticeable; but now each pixel in the image carries one bit of the message. Since ASCII requires 7 bits to represent a single character, each character of the message will be spread across 7 pixels in the image.

Someone else can then retrieve the hidden message by reading in the ones bits of each pixel. Every seven bits that are retrieved represents a single character in ASCII.

If the key is confidential, the steganographic algorithm can be public (Kirchhoff's' Principle). It is possible to decide whether the bits read are in fact an encoded message of a potential steganogram only if one has the appropriate decryption key. Encryption is also advisable in addition to steganographic utilities which do not implicitly encrypt. In 2011, Ibrahim and Kuan said that there are three famous techniques can be used in data hiding and they are watermarking, steganography and cryptography.

Research in steganography technique has been done back in ancient Greek were during that time the ancient Greek practice of tattooing a secret message on the shaved head of a messenger, and letting his hair grow back before sending him through

enemy territory where the latency of this communications system was measured in months.

The most famous method of traditional steganography technique around 440 B.C. is marking the document with invisible secret ink, like the juice of a lemon to hide information. Another method is to mark selected characters within a document by pinholes and to generate a pattern or signature (Schneier, 2011). However, in the year 2003, Cole proved that most of the development and use of computerized steganography only occurred in the year 2000. The main advantage of steganography algorithm is because of its simple security mechanism.

Because the steganographic message is integrated invisibly and covered inside other harmless sources, it is very difficult to detect the message without knowing the existence and the appropriate encoding scheme (Jahnke & Seitz, 2008). There are several steganography techniques used for hiding data such as batch steganography, permutation steganography, least significant bits (LSB), bit-plane complexity segmentation (BPCS).

Fridrich, Goljan, and Du (2001) have shown that cover-images stored in the BMP format are a very poor choice for steganographic methods that work in the spatial domain. This is because the quantization introduced by JPEG compression can serve as a "semi-fragile watermark" or a unique fingerprint that can be used for detection of very small modifications of the cover-image by inspecting the compatibility of the stego image with the JPEG format. Indeed, changes as small as flipping the least significant bit (LSB) of one pixel can be reliably detected.

Consequently, one should avoid using decompressed BMP images as covers for spatial steganographic methods, such as the LSB embedding or its variants.

Literature Related to the Problem

The current problem is the security of a file that is being transmitted over the system. This security can be given in much different kind of ways. But combining encryption with steganography makes the best system with utmost security. This system is using two different methods of securing the data and creating a single system using those methods. The existing system does not provide the security to the text files and they can be easily compromised.

There can be an attack on the data or a file that user is sending through the internet when it is in the transit phase. When it is the internet medium it can be accessible to the attacker and if there is a weak encryption used then there is a high risk of security being compromised.

The main drawbacks of the existing system are as follows

- There is no authentication to the text file; it’s not known whether the file received is authentic or altered.
- The hackers may attack the text file.
- The text file can be easily accessible and available to people which violates the integrity of the file. It should only be accessible to the assigned person.
- The text file takes more space while transmitting it to another system.

Steganography does not change the structure of the secret message, but instead, it hides the medium so that the message is not visible (Johnson & Jajodia,

1998). Conway (2003) also stated that steganography system relies on the secrecy of the data encoding system. When the encoding system is known, the steganography system is defeated and not useful anymore. Walia et al. (2010) also said that steganography is the invisible communication between the sender the receiver. According to Friedman (1967), steganography eliminates the unwanted attention coming towards the media in which the message is hidden. In Steganography, only the sender and the receiver know the existence of the message, but whereas in cryptography the existence of the encrypted message is visible to everyone (Ramesh, Athithan & Thiruvengadam, 1993).

The concepts used in this paper include the block cipher IDEA and the basics of GAs IDEA work on 64-bits of plaintext and a 128-bit symmetric key and involve 8 rounds. The algebraic operations used in IDEA include XOR (bitwise addition modulo 2), Addition modulo 2^{16} and multiplication modulo (2^{16} + 1). The key generation cycle differs for encryption and decryption in IDEA. It requires 52 keys that are 16-bits each, where 4 keys are used for last round of output transform and 6 keys in each of the previous 8 rounds.

GAs is based on the concept of ‘survival of the fittest’ and work to find the optimal or near optimal solution for the optimization problems. The idea behind GA is to model the natural selection process where some individuals are selected from the population and where genetic operators are applied to produce improved generation. The genetic operators involved are:

Selection operator: The motive behind this operator is to select the best parents, to transfer better characteristics to next generation. The goodness of everyone

in a generation depends upon its fitness, which may be calculated by an objective function or by a subjective judgment (Ghosh, 2012).

Crossover: The two best individuals are chosen using the selection operator and a crossover site is randomly chosen. Bits are exchanged in the bit strings up to the crossover point.

For example:

If S1 = 11111111 and S2 = 00000000 and the crossover point is 5, then S1′=11111000 and S2′= 00000111.

Crossover is likely to produce better offspring, as compared to the parent bit strings. Crossover can be categorized as One point or Single, two-point or Double, Uniform, Half-Uniform, Cut and Splice.

Single crossover includes the swapping of bits across a randomly selected crossover bit as shown in Figure below.

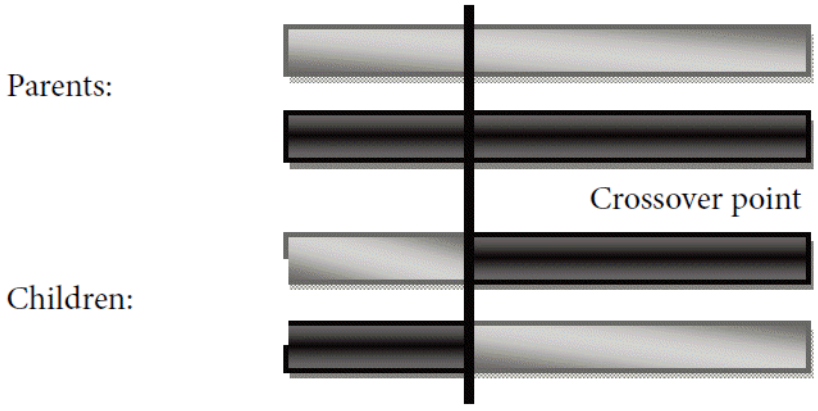

Figure 2.2: Single crossover point (Ghosh, 2012) modified improved text encryption approach inspired by genetic algorithm techniques using RSA algorithm.

Double crossover includes the selection of two random bits and swapping the bit strings across them, as represented in Figure below

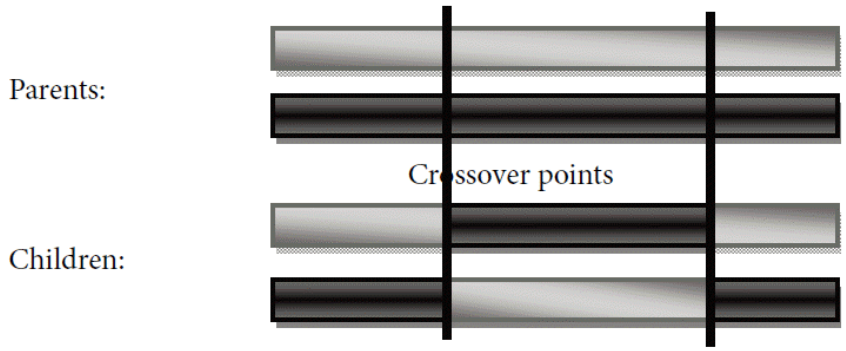

Figure 2.3: Double crossover point (Ghosh, 2012) modified improved text encryption approach inspired by genetic algorithm techniques using RSA algorithm.

There are a few other crossover techniques that are based on swapping of bits, but these are not discussed in detail in this paper.

Mutation: Some of the bits of the parent bit strings will be flipped or toggled to maintain diversity within the population, as illustrated in Figure below.

The figure above mutates the bit string around bit 4.

Bit position	0	1	2	3	4	5	6	7
Before	1	1	0	1	1	0	1	0
After	1	1	0	1	0	0	1	0

Figure 2.4: Mutation (Ghosh, 2012). A modified improved text encryption approach inspired by genetic algorithm techniques using RSA algorithm.

Steganography and Cryptography are different in their way of data hiding but they are in fact complementary techniques. Kahate (2008) summary states that no matter how strong the encryption algorithm may be if the secret message is discovered, it will be subject to cryptanalysis. Similarly, how well a message is hidden inside a digital media there is a possibility of the hidden message to be discovered by the third party. By combining Steganography and Cryptography there can be a better security by hiding the existence of an encrypted message (Song, Zhang, Liao, Du & Wen, 2011). The

resulting stego-object can be transmitted without revealing that secret information is being exchanged. In 2010, Fadhil proved that even if an attacker detected the message from the stego-object, he first needs to decode the message from digital media and then he would still require the cryptographic algorithm to decipher the encrypted message.

In the modern cryptography, there are three types of cryptographic algorithms used called Symmetric key cryptography, Public-key cryptography, and hash functions. Symmetric key cryptography involves encryption methods where both the sender and the receiver share the same key used to encrypt the data. In Public-key cryptography, two different but mathematically related keys are used. Hash functions do not use a key. Instead, they compute a fixed length hash value from the data. It is impossible to recover the length of the original plaintext from this hash value.

Research in hiding data inside the image using steganography technique has been done by many researchers. Warkentin, Bekkering , and Schmidt (2008) proposed an approach to hide data inside the audio-visual files. In their steganography algorithm, to hide data, the secret content must be hidden in a cover message. El-Emam (2007), on the other hand, proposed a steganography algorithm to hide a large amount of data with high security. His steganography algorithm is based on hiding a large amount of data (image, audio, text) file inside a color bitmap (BMP) image. Here the image will be filtered and segmented where bits replacement is used on the appropriate pixels.

The choice of cover-images is important because it significantly influences the design of the stego-system and its security. Images with a small number of colors, computer art, images with a unique semantic content, such as fonts, should be avoided. Aura (1996) recommends grayscale images as the best cover-images. He also

recommends uncompressed scans of photographs or an image obtained from a digital camera containing a high number of colors and considers them safest for steganography.

Modern Steganography methods are called Digital Steganography. These modern methods include hiding messages within noisy images, embedding a message within random data, embedding pictures with the message within video files, etc. Furthermore, Network Steganography is used in telecommunication networks. This includes techniques like Steganophony (hiding a message in Voice-over-IP conversations) and WLAN Steganography (methods for transmitting steganogram in Wireless Local Area Networks).

Table 2.1

Difference between Steganography and Cryptography

Steganography	**Cryptography**
Objective	Technique
Confidentiality	Cryptography
Integrity	Cryptography
Authentication and non-repudiation	Hashing Algorithms
Data hiding	Steganography

The choice of the image format also makes a very significant impact on the design of a secure steganographic system. Raw, uncompressed formats, such as BMP, provide the biggest space for secure steganography, but their obvious redundancy makes them very suspicious in the first place. Indeed, some researchers do not

consider those formats for steganography claiming that exchanging uncompressed images is equivalent to using cryptography (Eggers, Baeuml, & Girod, 2002). Never the less, most steganographic products available on the Internet work with uncompressed image formats or formats that compress data lossless (BMP, PCX, GIF, PGM, and TIFF).

Chen and Wu (2009) modified a method used in "A steganographic method for a digital image using side match" by Chang and Tseng (2004). They concentrated on hiding the data in the edge portions of the image. Wu and Tsai (2003) on the other hand, used pixel-value differencing by partitioning the original image into non-overlapping blocks of two consecutive pixels.

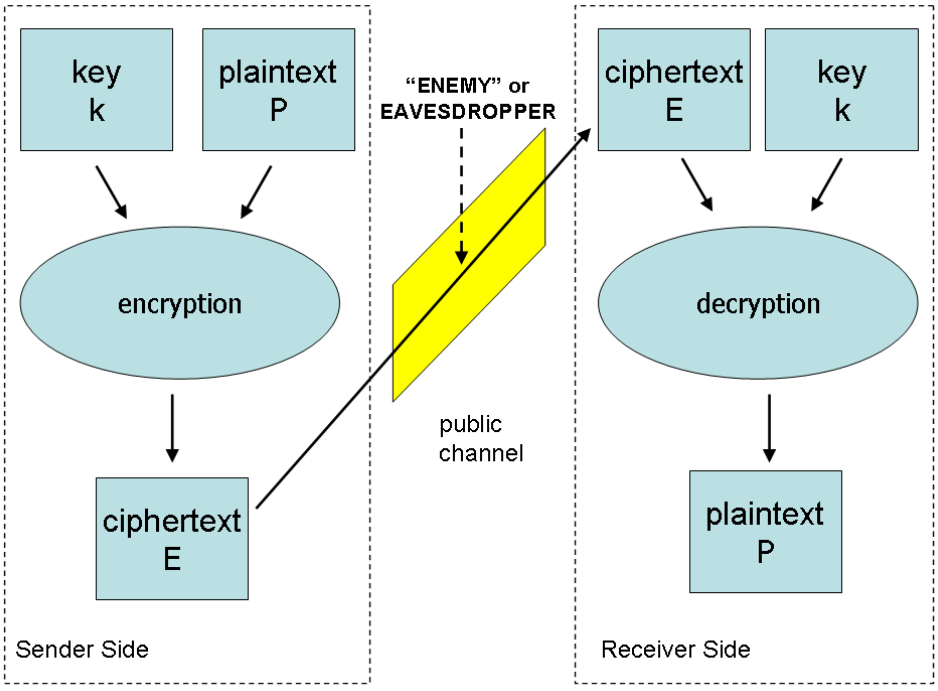

Figure 2.5: Symmetric-key cryptographic model. (Adapted from Bloisi & Iocchi, 2007).

The key encryption used for the symmetric key is shown in the figure above. There is a public channel that possesses the high-risk issue in this model. Instead of having the security of key ‘k' and using the symmetric key encryption to get the ciphertext, there is still a probability that the attacker may read the secret message before it reaches the receiver.

Literature Related to the Methodology

The advancement of the steganography depends on cryptographic components of the text document information. Steganography is a component that can be used as a part of any application to give the layer of security to the content document.

IDEA operates with 64-bit plaintext and ciphertext blocks and is controlled by a 128-bit key. IDEA was originally called IPES (Improved Proposed Encryption Standard). It avoids the substitution boxes and table lookups used in the block ciphers. The algorithm structure has been chosen such that when different key sub-blocks are used, the encryption process is identical to the decryption process.

International Data Encryption Algorithm (IDEA) is a block cipher designed by Xuejia Lai and James L. Massey (1990) of ETH-Zürich and was first described in 1991. It is a minor revision of an earlier cipher, PES (Proposed Encryption Standard); IDEA was originally called IPES (Improved PES). IDEA was used as the symmetric cipher in early versions of the Pretty Good Privacy cryptosystem.

There are numerous explorations that have been carried out so far with regards to IDEA and the Genetic Algorithm. Research has been executed in the field of the cryptanalysis of various rounds of IDEA. In 1993, Hawkes extended the study of Daemen, Govaerts, and Vandewalle, which identified certain weak classes for IDEA. In

1998, Hawkes presented a related-key differential attack on 4 rounds of IDEA. Moreover, he identified large weak key classes for 4.5–6.5 rounds and 8 rounds of the algorithm.

Biryukov, Nakahara, Preneel, and Vandewalle in 2002, identified a few classes of weak keys in 2002. In 2010, Zhu further put forward an attack on 5.5 rounds of IDEA. Also, in 2010, Abed proved that the GA gives optimal keys and enhances the security of the key. Abed, 2010 used the RC4 algorithm and applied it to images. In 2011, Bhowmik and Acharyya worked on images and combined GA with Blowfish. In 2012, Goyat proved that the GA maintains the strength of the asymmetric key. She presented the idea that random numbers that are generated should be secure enough against attacks. Most of the research that has been done on IDEA, as well as using the genetic approach, involves conducting cryptanalysis on various rounds of IDEA and on various ciphers, respectively. This research paper provides an optimal solution for the problem of weak keys.

IDEA was to develop a strong encryption algorithm, which would replace the DES procedure developed in the U.S.A. in the seventies. It is also interesting in that it entirely avoids the use of any lookup tables or S-boxes. When the famous PGP email and file encryption product was designed by Zimmermann (McLaughlin, 2006), the developers were looking for maximum security. IDEA was their first choice for data encryption based on its proven design and its great reputation. Since IDEA is the improved version for replacing the previous DES (Data Encryption Standard). It is the revised version of Proposes Encryption Standard (PES).

The IDEA Encryption Algorithm:

- Provides high-level security not based on keeping the algorithm a secret, but rather upon ignorance of the secret key.
- Is fully specified and easily understood.
- Is available to everybody.
- Is suitable for use in a wide range of applications.
- Can be economically implemented in electronic components (VLSI Chip).
- Can be used efficiently.
- May be exported worldwide.
- Is patent protected to prevent fraud and piracy.

International Data Encryption Algorithm (IDEA) is a popular and secure cryptography algorithm, suitable for hardware implementation. IDEA comprises of modulo 2^{16} additions, bitwise exclusive-OR operations and modulo 2^{16+1} multiplications of 16-bit words (Pekmestzi, Efstathiou, Moschopoulos, & Tsoumanis, 2013). The IDEA algorithm has been implemented in software on Intel Pentium II 445 MHz with encryption rate of 23.53 Mb/Sec (Cheung, Tsoi, Leong, & M. Leong, 2001).

According to Schneier (2007), there are two cryptographic techniques symmetric key cryptography and asymmetric key cryptography both of which are implemented by various algorithms that are defined in this field. In this research paper, it considers International Data Encryption Algorithm (IDEA), which is a symmetric key algorithm, for encryption and decryption.

A genetic algorithm (GA) is a search heuristic that is used to portray the process of natural evolution. This algorithm provides solutions to optimization and search problems (Goldberg, 1989).

The proposed method in this paper uses a GA for the generation of a 128-bit symmetric key that is used in IDEA. It inhibits the creation of any weak key for this cryptographic technique.

The remainder of this paper is laid out as follows: In Chapter II it describes the basics of IDEA and sheds some light on the fundamentals of the GA and its operators. Explains some of the related work done till now on this issue. In Chapter III, it has the methodology that has been adapted to generate a symmetric key genetically. In Chapter IV, the results from using this methodology are illustrated.

Steganography is the method of hiding confidential messages and information into digital media in such a way that no one apart from the sender and intended receiver ever realizes that there is a hidden message inside the media (Conway, 2003). Digital rights, information security and conceal secrets are addressed by using the steganography techniques. Johnson and Jajodia (1998) proposed that the steganographic systems in the present days mostly use images as cover media because digital media is mostly transmitted over Internet communication. Digital images contain a high amount of redundant data and noise in them. This gives space to embed the data into them and this modification is not even visible to the human eye. Li et al., (2011) stated that steganalysis is a term closely related to steganography which is a method for detecting hidden messages in a digital medium.

Today steganography advances are followed by the advances in steganalysis. The steganographic methods that make changes to the image that are not visible to the human eye. Kharrazi et al. (2006) review say that this feature is not enough because statistical methods can detect the changes in the image even if it is not visible. Watson's (1994) Image compression using the discrete cosine transform summary's states that compression also plays a vital role in the image-based steganography as the result of the steganographic technique depends on the compression scheme used. There is more research being done by the stenographers as they are trying to find out a more efficient method of hiding the message in a digital life, only to avoid being defeated by the techniques derived by steganalysis.

The existing cryptographic system can give privacy and confidentiality, but they don't have a section to hide cryptographic communication. Steganography can be used under certain circumstances for data security as it makes communication invisible, but steganography like cryptography can be detected (Li et al., 2011). Thus, the approach here is to transfer some of the features of steganography to cryptography, rather using steganography by itself.

Raphael and Sundaram (2011) compared that neither cryptography nor steganography alone is an effective way to implement information security, but by combining both it can make a very good method of data security. The secret message that is to be transmitted is first encrypted using the IDEA algorithm and then embedded into the image using steganography. The security of the data is doubled and increased. Both steganography and cryptography work differently on how to hide the data. The aim

of this paper is to make a model that has the features of both cryptography and steganography.

This research uses a similar concept introduced by El-Emam (2007). A bitmap (BMP) image will be used to hide the data. Data will be embedded inside the image using the pixels. Then the pixels of stego image can then be accessed back to retrieve back the hidden data inside the image. Two stages are involved. The first stage is to use the IDEA algorithm to hide the data inside the image and the second stage is to use decryption algorithm using IDEA algorithm as well to retrieve the hidden data that is hidden within the stego image.

Song et al. (2011) state this model can be successful when it can integrate the cryptography and steganography through image processing. The outcome of the combination model will be a steganographic one that will perform cryptographic functionality and, preserving the steganographic nature of the image (Fadhil, 2010).

Summary

This chapter was all about the literature that was related to the problem and methodology both. The literature also reviewed what the problems are in the existing system and how the system proposed manages to overcome those drawbacks. The next chapter while mainly focus on what methodology is going to be used in this paper to solve the problem. The next chapter will elaborate on working with the IDEA algorithm and how the desired output can be achieved and the advantages of the proposed system.

Chapter III: Methodology

Introduction

In this paper, it is proposing a technique of combining both steganography and cryptography to solve the problem of unauthorized data access and security of the data using the IDEA algorithm. Steganography can also be implemented to cryptographic data to increase the security of the data (Raphael & Sundaram, 2011, quoted in Song et al., 2011). It is going have how to learn about the outline of how IDEA algorithm works and its advantages.

Design of the Study

The proposed framework depends on the cryptography components of the text document. The text file to be transmitted is converted into BMP picture file using the algorithm. The proposed framework takes the text file and encryption key as input and gives a BMP document as output using the IDEA algorithm. This framework gives the security and changes the text file into the image file.

The receiver of the image should use the same algorithm used for encryption to decrypt the file. By using the encryption key and the image received it can decrypt the message and get the text document as the output.

The quantitative approach is going to be used in this proposed paper. As the quantitative approach best suites this kind of study where it needs to deal with numerical data and compare it with the previous data. So, the quantitative approach provides best result outcome of this study.

The image below is a simple way to illustrate the process of the proposed system of how both steganography and cryptography can be combined for the secure and efficient transfer of data over the internet.

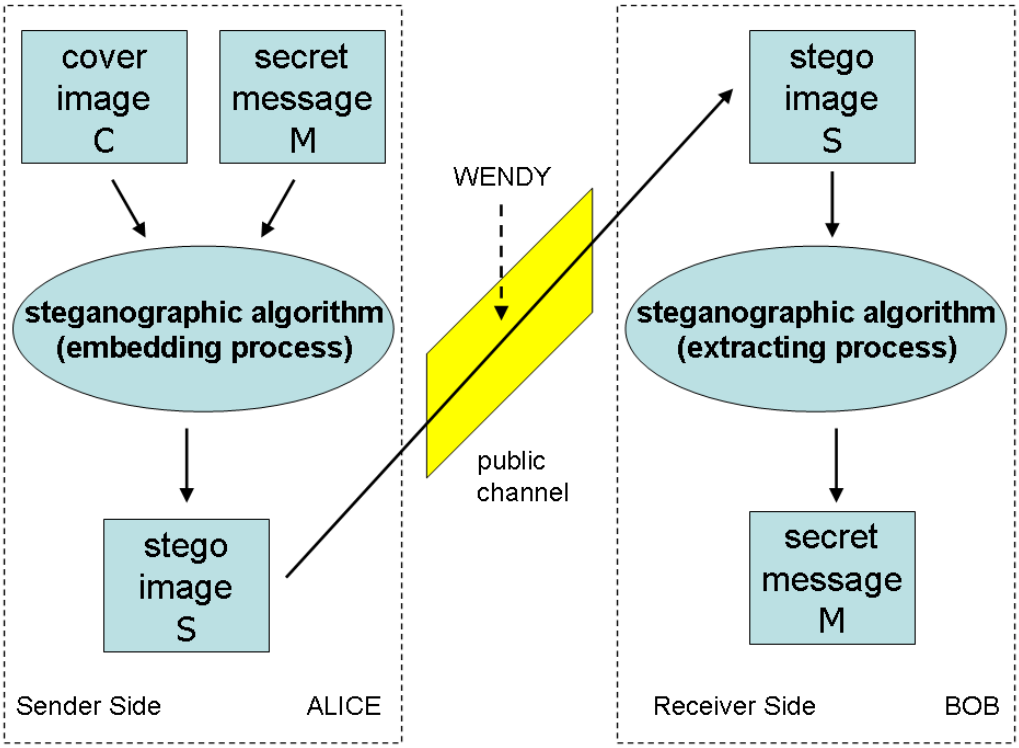

Figure 3.1: Steganographic model. (Adapted from Bloisi & Iocchi, 2007).

The image shown above describes the proposed system of using an image to embed a message into it by the means of IDEA algorithm in place of a steganographic algorithm. It has Alice the sender who is wishing to send a secret message 'M' to Bob who is the receiver here to do this, Alice chooses a cover image ‘C’. The steganographic algorithm identifies C’s redundant bits, i.e., those that can be modified without arising Wendy’s suspicion, then the embedding process creates a stego image ‘S’ by replacing these redundant bits with data from ‘M’.

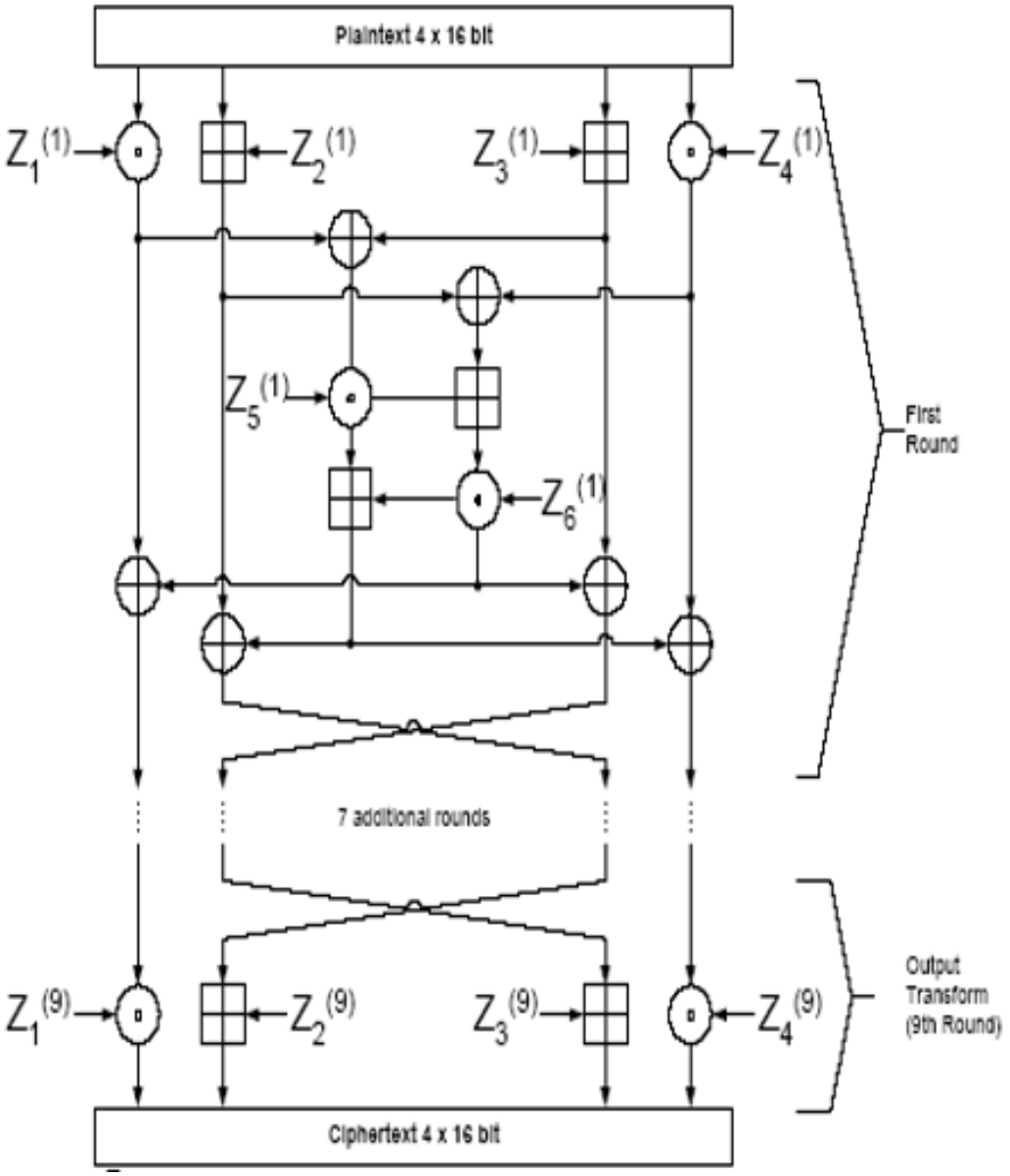

Figure 3.2: Block diagram of IDEA algorithm. (Adapted from Leong, Cheung, Tsoi, & Leong, 2000).

Then that stego image ‘S' is transferred over the internet using the cryptography algorithm. It hides the stego image using the IDEA algorithm for encryption and makes the data both encrypted and stego image. By doing this the third person Wendy could not anything here as the data is first encrypted and then stego image. Knowing both the

hiding techniques is quite difficult for the attacker. The encryption key plays a vital role here as it holds the way to decrypt the message.

The block cipher IDEA operates with 64-bit plaintext and ciphertext blocks and is controlled by a 128-bit key. The fundamental innovation in the design of this algorithm is the use of operations from three different algebraic groups. The substitution boxes and the associated table lookups used in the block ciphers available to-date have been completely avoided. The algorithm structure has been chosen such that, with the exception that different key sub-blocks are used, the encryption process is identical to the decryption process.

Encryption Process

The functional representation of the encryption process is shown in Figure 3.2. The process consists of eight identical encryption steps (known as encryption rounds) followed by an output transformation. The structure of the first round is shown in detail in Figure 3.2.

In the first encryption round, the first four 16-bit key sub-blocks are combined with two of the 16-bit plaintext blocks using addition modulo 2^{16}, and with the other two plaintext blocks using multiplication modulo $2^{16} + 1$. The results are then processed further as shown in Figure 3.3, whereby two more 16-bit key sub-blocks enter the calculation and the third algebraic group operator, the bit-by-bit exclusive OR, is used. At the end of the first encryption round four 16-bit values are produced which are used as input to the second encryption round in a partially changed order.

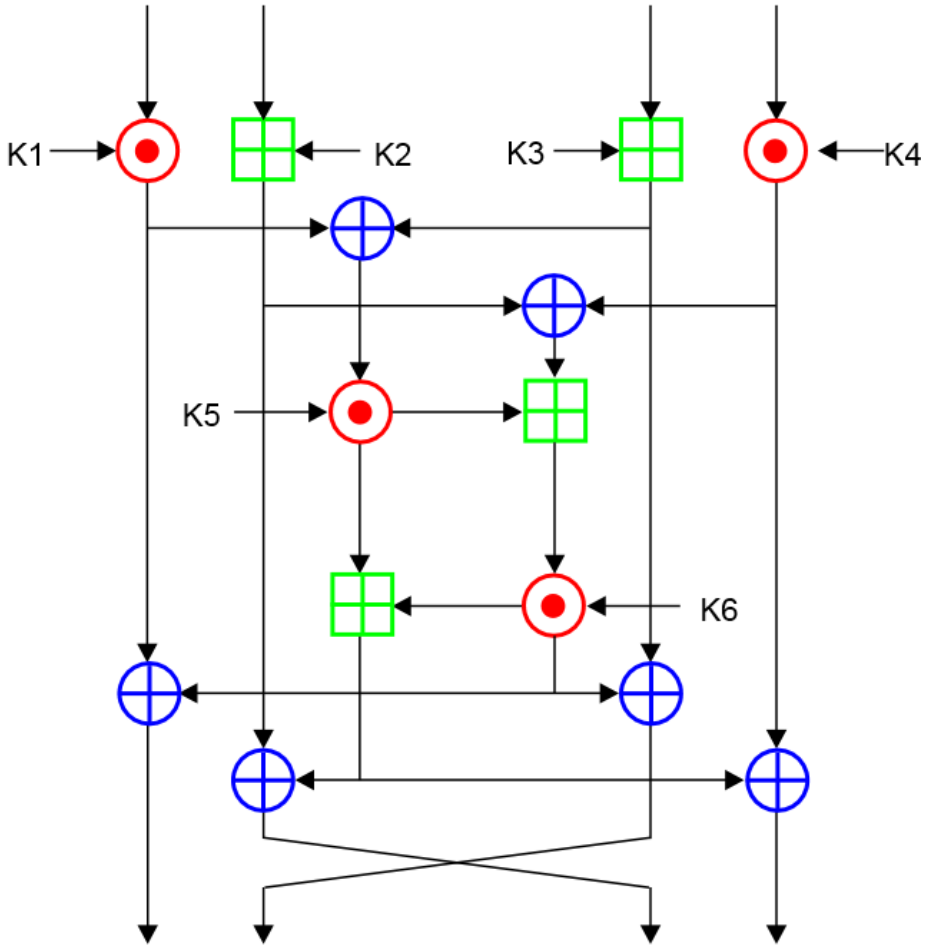

Figure 3.3: A single encryption round of IDEA. (Reprinted from International Data Encryption Algorithm, n.d.).

The process described above for round one is repeated in each of the subsequent 7 encryption rounds using different 16-bit key sub-blocks for each combination. During the subsequent output transformation, the four 16-bit values produced at the end of the 8th encryption round are combined with the last four of the 52 key sub-blocks using addition modulo 2^{16} and multiplication modulo $2^{16} + 1$ to form the resulting four 16-bit ciphertext blocks.

- The 64-bit plaintext block is partitioned into four 16-bit sub-blocks.
- Six 16-bit key is generated from the 128-bit key. Since a further four 16-bit key-sub-blocks are required for the subsequent output transformation, a total of (8 x 6 + 4 = 52) different 16-bit sub-blocks must be generated from the 128-bit key.

The operators used in Figure 3.4 and their uses are described below along with simple graphical representation.

- Bitwise exclusive OR (denoted with a blue circled plus ⊕).
- Addition modulo 216 (denoted with a green boxed plus ⊞).
- Multiplication modulo 216+1, where the all-zero word (0x0000) in inputs is interpreted as 216 and 216 in output is interpreted as the all-zero word (0x0000) (denoted by a red circled dot ⊙).

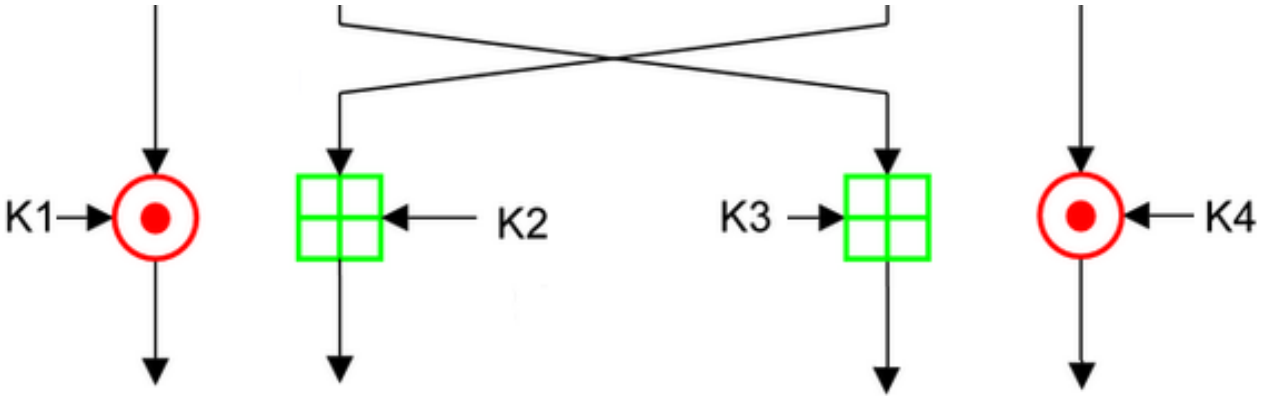

Figure 3.4: Describing the use of operators in IDEA algorithm. (Reprinted from International Data Encryption Algorithm, n.d.).

Key Generation

The 64-bit plaintext block is partitioned into four 16-bit sub-blocks since all the algebraic operations used in the encryption process operate on 16-bit numbers. Another

process produces for each of the encryption rounds, six 16-bit key sub-blocks from the 128-bit key. Since a further four 16-bit key-sub-blocks are required for the subsequent output transformation, a total of 52 (= 8 x 6 + 4) different 16-bit sub-blocks must be generated from the 128-bit key.

Table 3.1

Key Encryption

Round 1	$Z_1^{[1]}$ $Z_2^{[1]}$ $Z_3^{[1]}$ $Z_4^{[1]}$ $Z_5^{[1]}$ $Z_6^{[1]}$
Round 2	$Z_1^{[2]}$ $Z_2^{[2]}$ $Z_3^{[2]}$ $Z_4^{[2]}$ $Z_5^{[2]}$ $Z_6^{[2]}$
Round 3	$Z_1^{[3]}$ $Z_2^{[3]}$ $Z_3^{[3]}$ $Z_4^{[3]}$ $Z_5^{[3]}$ $Z_6^{[3]}$
Round 4	$Z_1^{[4]}$ $Z_2^{[4]}$ $Z_3^{[4]}$ $Z_4^{[4]}$ $Z_5^{[4]}$ $Z_6^{[4]}$
Round 5	$Z_1^{[5]}$ $Z_2^{[5]}$ $Z_3^{[5]}$ $Z_4^{[5]}$ $Z_5^{[5]}$ $Z_6^{[5]}$
Round 6	$Z_1^{[6]}$ $Z_2^{[6]}$ $Z_3^{[6]}$ $Z_4^{[6]}$ $Z_5^{[6]}$ $Z_6^{[6]}$
Round 7	$Z_1^{[7]}$ $Z_2^{[7]}$ $Z_3^{[7]}$ $Z_4^{[7]}$ $Z_5^{[7]}$ $Z_6^{[7]}$
Round 8	$Z_1^{[8]}$ $Z_2^{[8]}$ $Z_3^{[8]}$ $Z_4^{[8]}$ $Z_5^{[8]}$ $Z_6^{[8]}$
Output Transform	$Z_1^{[9]}$ $Z_2^{[9]}$ $Z_3^{[9]}$ $Z_4^{[9]}$

The key sub-blocks used for the encryption and the decryption in the individual rounds are shown in Table 3.1.

The 52 16-bit key sub-blocks which are generated from the 128-bit key are produced as follows:

- First, the 128-bit key is partitioned into eight 16-bit sub-blocks which are then directly used as the first eight key sub-blocks.
- The 128-bit key is then cyclically shifted to the left by 25 positions, after which the resulting 128-bit block is again partitioned into eight 16-bit sub-blocks to be directly used as the next eight key sub-blocks.

- The cyclic shift procedure described above is repeated until all the required 52 16-bit key sub-blocks have been generated.

Decryption

The computational process used for decryption of the ciphertext is essentially the same as that used for encryption of the plaintext. The only difference compared with encryption is that during decryption, different 16-bit key sub-blocks are generated. More precisely, each of the 52 16-bit key sub-blocks used for decryption is the inverse of the key sub-block used during encryption in respect of the applied algebraic group operation. Additionally, the key sub-blocks must be used in the reverse order during decryption to reverse the encryption process as shown in Table 3.2.

Table 3.2

Key Decryption

Round 1	$Z_1^{[9]-1}\ Z_2^{[9]}\ Z_3^{[9]}\ Z_4^{[9]-1}\ Z_5^{[8]}\ Z_6^{[8]}$
Round 2	$Z_1^{[8]-1}\ Z_3^{[8]}\ Z_2^{[8]}\ Z_4^{[8]-1}\ Z_5^{[7]}\ Z_6^{[7]}$
Round 3	$Z_1^{[7]-1}\ Z_3^{[7]}\ Z_2^{[7]}\ Z_4^{[7]-1}\ Z_5^{[6]}\ Z_6^{[6]}$
Round 4	$Z_1^{[6]-1}\ Z_3^{[6]}\ Z_2^{[6]}\ Z_4^{[6]-1}\ Z_5^{[5]}\ Z_6^{[5]}$
Round 5	$Z_1^{[5]-1}\ Z_3^{[5]}\ Z_2^{[5]}\ Z_4^{[5]-1}\ Z_5^{[4]}\ Z_6^{[4]}$
Round 6	$Z_1^{[4]-1}\ Z_3^{[4]}\ Z_2^{[4]}\ Z_4^{[4]-1}\ Z_5^{[3]}\ Z_6^{[3]}$
Round 7	$Z_1^{[3]-1}\ Z_2^{[3]}\ Z_2^{[3]}\ Z_4^{[3]-1}\ Z_5^{[2]}\ Z_6^{[2]}$
Round 8	$Z_1^{[2]-1}\ Z_3^{[2]}\ Z_2^{[2]}\ Z_4^{[2]-1}\ Z_5^{[1]}\ Z_6^{[1]}$
Output Transform	$Z_1^{[1]-1}\ Z_2^{[1]}\ Z_3^{[1]}\ Z_4^{[1]-1}$

Advantages of the Proposed System

- This framework provides the basic security benefit of confidentiality, as the data received is not accessible to anyone except the right person who has the encryption key and the image.
- The combination of these two methods will enhance the security of the data embedded and will satisfy the requirements such as capacity, security, and robustness.
- This framework provides the reliability of a text file.
- This framework keeps up the same configuration following to giving security highlights.
- The extra layer of security using the encryption technique provides the user to send more confidential files through this system.
- If an attacker were to defeat the steganographic technique to detect the message from the stego-object, he would still require the cryptographic method to decipher the encrypted message.

The electronic data transmission is increasing day by day, so there is a need for security of the data. When the PGP (pretty good privacy) cryptosystem was designed IDEA was the choice of data encryption technique used. The important aspect of the development of IDEA was to provide security to all military requirements and easy software and hardware implementation.

Data Analysis

The tools and techniques that are going to be used to analyze the data are steganography, cryptography, encryption, decryption, compression and data transfer.

Hardware and software requirements:

- System: Intel core i5.
- Operating system: Linux.
- Software version: EditPlus 2.
- JPEG/BMP compressor.
- IDEA Algorithm for encryption.
- Coding language: JAVA / C.
- Steganography software: Image steganography.

Summary

In this chapter, it was all about the methodology that is going to be used for solving the problem. Here IDEA algorithm is the method that is going to be used in this study to solve the security problem while data transfer over the system. The next chapter is about the implementation of the IDEA algorithm and the techniques and methods used to achieve the purpose of the paper. It will deal with the data analysis and data presentation that must be done.

Chapter IV: Implementation

Introduction

Here in steganography first the text file to be hidden is encrypted to get the ciphertext and it is hidden in the selected image file in BMP format. And the reverse process is done to get the text file from the image file.

Glossary

Various terms used in the system are:

- *Carrier File*: It is a file which has been hidden information inside of it.
- *Steganalysis*: It is a process of detecting hidden information inside of a file.
- *Stego-Medium*: It is a medium in which the information is hidden.
- *Redundant Bits*: Pieces of information hidden inside a file which can be altered without damaging the file.
- *LSB*: (Least Significant Bytes) When files are created there are usually some bytes in the file that aren't really needed, or at least aren't that important.
- *Embed*: The process of hiding text in an Image file.
- *Extract*: The process of detecting the hidden text in an audio file.
- *IDEA*: International Data Encryption Algorithm is an algorithm to encrypt the plain text and to decrypt the ciphertext.
- *PBE*: Password-Based Encryption is the process of doing encryption using a key generated from the given password.

Domain: If one could hide the message in the image file in such a way, that there would be no perceivable changes in the image file after the message insertion. At the same time, if the message that is to be hidden were encrypted, the level of security would be raised to quite a satisfactory level.

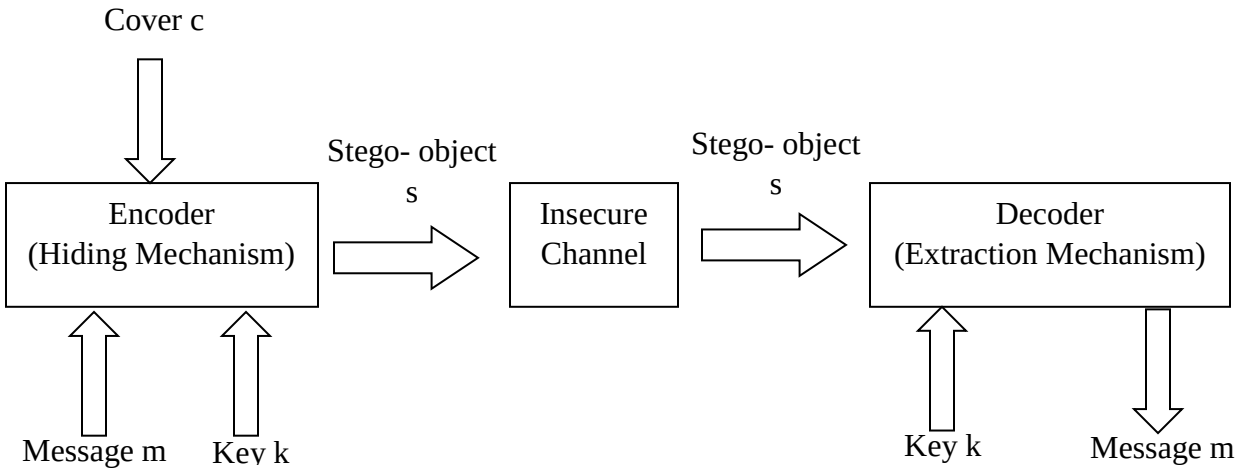

Figure 4.1: Algorithm workflow. (Adapted from Fairley, 1985).

Here the cover is an image file. The hiding mechanism used is “Low bit encoding”. The message and the key are used to generate the ciphertext using IDEA algorithm.

Generating the ciphertext: The ciphertext is generated through IDEA algorithm. A key is generated from the given password, then password-based encryption is done to get the ciphertext.

Encryption method: It uses password-based encryption method. A general approach to PBE for the protection of the password is 64-bit plaintext and a 128-bit key.

Encryption: An encryption scheme has five ingredients:

- Plaintext.

- Encryption algorithm.
- Secret Key.
- Ciphertext.
- Decryption algorithm.

Security depends on the secrecy of the key, not the secrecy of the algorithm.

Cryptography: Classified along three independent dimensions:

- The type of operations used for transforming plaintext to ciphertext.
- The number of keys used.
    - Symmetric (single key).
    - Asymmetric (two-keys, or public-key encryption).
- The way in which the plaintext is processed

Encryption algorithm: International Data Encryption Algorithm (IDEA)

- The most widely used encryption scheme.
- The algorithm is referred to the International Data Encryption Algorithm (IDEA).
- IDEA is a block cipher.
- The plaintext is processed in 64-bit blocks.
- The key is 128-bits in length.

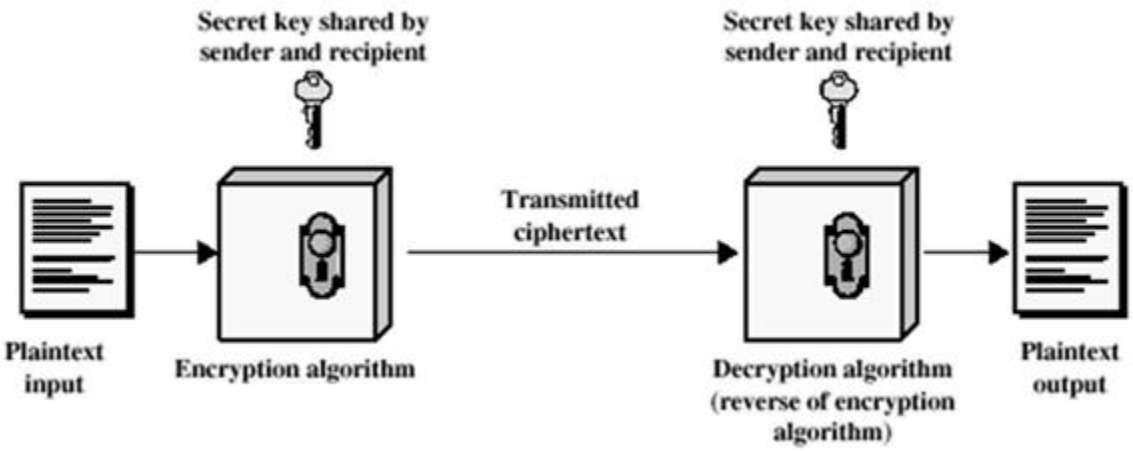

Figure 4.2: IDEA algorithm encryption/decryption model. (Adapted from Stallings, 2006).

Embedding: Now the ciphertext is placed in the image file in such a way that there would be no change in the image. Using the Low Bit Encoding method, the message is inserted in the image. By replacing the least significant bit of each sampling point by a coded binary string, it can encode a large amount of data in an image file (Gupta & Sharma, 2013).

The major disadvantage of this method is its poor immunity to manipulation. The encoded information can be destroyed by channel noise, resampling unless it is encoded using redundancy techniques. This method is useful only in closed, digital to digital environments.

Functional Requirements

Functional requirements describe the interactions between the system and its environment independent of its implementation. The environment includes the user and any other external system with which the system interacts.

The following are the functional requirements for the proposed system.

- The system takes a text file and 16-bit secret key as input to the system.
- After that, it will generate a BMP image by using Encryption /Decryption algorithms.
- The encrypted data is saved using relevant Compression/Decompression algorithms.
- The compressed data is encoded/ decoded to obtain final encrypted data, which must be represented in text format.

The above-mentioned requirements show only the possible interactions between the system and its external world. The above description does not focus on any of the implementation details.

Non- Functional Requirements

Non-Functional requirements describe the user-visible aspects of the system that are not directly related to the functional behavior of the system. Non-functional requirements include quantitative constraints, such as response time or accuracy.

- The algorithm that should be executed by the system has to be selected by the user.
- The user must supply the key to the algorithms.
- User Interface and Human factor: The user need not be technically sound, as these interfaces are explanatory.
- Documentation: User manual is provided to the user for using the system.

- Hardware Considerations: This is the Minimum hardware to run the corresponding this software. The system will not interact with another hardware system.
    - RAM: 64 MB(Minimum)
    - Hard Disk: 4 MB (Minimum)
    - Processor: Intel processor

Performance characteristics: The operations done in this system are according to the algorithms selected. Here the system just executes those operations. So, here the performance will depend on the efficiency of the algorithm used.

Error handling and extreme conditions: There may be cases where the user not providing sufficient input parameters for the method he selected. In those cases, an exception should be raised.

Pseudo Requirements

Pseudo Requirements are requirements imposed by the client that restrict the implementation of the system. Typical pseudo requirements are the implementation language and the platform on which the system is to be implemented. Pseudo requirements have usually no direct effect on the user's view of the system. The client has imposed that the system must be developed or written using JAVA, to comply with the current company policy.

The minimum software requirements are:

- Operating System **:** MS-Windows
- Language **:** JAVA 1.6.1

- Operating System : Windows 8
- Main memory : 128 MB
- Secondary memory : 8 MB for program
- Software : Java JDK 1.6.1

Tasks and procedures currently proposed:

The proposed system should perform the following operations:

- Choose Image file: System should read the image file selected by the user.
- Viewing Image file: System should able to view the selected image file.
- Encryption/ Decryption: System should able to encrypt/ decrypt the selected text.
- Embed: System should able to embed the ciphertext into an image file.
- Extract: System should able to extract the ciphertext from the embedded image file.

Feasibility Analysis

Technical feasibility. The package ‘JAVA' which is being used to develop this software is technically a reliable package to develop any system. It is capable of handling audio files, playing audio clips and to create a most user-friendly GUI.

Development risk. There is no risk in developing the software using packages ‘JAVA’ since it is widely used package and more reliable to develop systems.

Resources availability. The system does not require any additional resources to perform its operations, but to read the audio format is the only task performed by the user manually.

Functional feasibility. Functionally the system has no drawbacks since it is designed under desired conditions.

Economic feasibility. Economically the system has no troubles.

Encrypting Message

Substitutions and transpositions encryption are regarded as the building blocks of Classical cryptography technique (Kahate, 2008). A transposition cipher hides information by reordering the letters of the message. Ramesh et al. (1993) proposed that a transposition cipher the plaintext remains the same, but the order of characters is shuffled around. In 1989, Anderson reviewed that the frequency analysis on the ciphertext would reveal that each letter has approximately the same. A substitution cipher is one in which each character in the plaintext is substituted for another character in the ciphertext. A substitution cipher is an encryption scheme that uses only substitution transformations.

The two other techniques related to transposition and substitution for obscuring the redundancies in a plaintext message are diffusion and confusion. Diffusion dissipates the redundancy of the plaintext by spreading it out over the ciphertext. The simplest way to cause diffusion is through transposition. In 2005, Mollin said that confusion obscures the relationship between the plaintext and the ciphertext. The easiest way to do this is through substitution. In the proposed method substitution encryption method is used.

The orders of the letters are changed in transposition cipher, whereas in substitution cipher the letters are replaced with another letter to make the message unintelligible (Kaijser, Parker, & Pinkas, 1994). In a substitution cipher, the algorithm is

to offset the alphabet and the key is the number of characters to offset it. Ravi and Knight (2009) stated that the receiver inverts the substitution on the ciphertext to recover the plaintext. For example, if it encrypts the word “MESSAGE” by shifting 18 places, then “CRYPTOGRAPHY” encrypts as “UJQHLGYJSHZQ”.

To allow someone else to read the ciphertext, it tells the recipient that the key is 18. Now if it supposes A (sender) wants to send B (recipient) the plaintext message M over the insecure communication line, A encrypts M by computing the ciphertext C = E (K, M) and sends C to B. Upon receipt, B decrypts C by computing M = D (K, C). The adversary may know E and D, are the encryption and decryption algorithms respectively which are being used in the process.

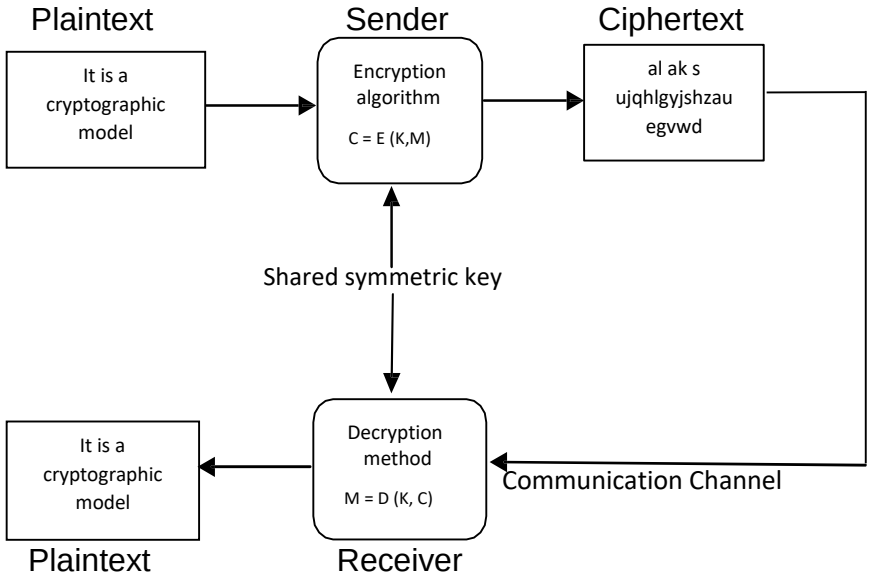

Figure 4.3: Symmetric encryption system. (Reprinted from Menezes, Van Oorschot, & Vanstone, 1996).

Plaintext*:* It is the original message which is to be transmitted. It can be written as M =<m1, m2, . . ., mn > to denote the plaintext.

For example, M =< t,h,i,s, ,i,s, ,a,n, ,e,n,c,r,y,p,t,i,o,n, ,m,e,t,h,o,d >.

Cipher text: It is the translated or encrypted message, which can be denoted as C =< c1, c2, . . . , cm >. Thus, the cipher text C =<j,x,y,i, ,y,i, ,q,d, ,u,d,s,h,o,f,j,y,e,d, ,c,u,j,x,e,t >.

Encryption*:* The process of transformation from plaintext to ciphertext. The encryption is denoted using C = E (M), where C is the ciphertext and M is the plaintext and E is the encryption method.

Decryption*:* It is the process of reversing the encryption, which is the transformation from ciphertext to plaintext, formally denoted as M = D(C).

Key*:* Key is the agreement between the sender and the recipient. In the present encryption method, the key tells how much to shift. It is an input to the encryption and decryption algorithm (Kahate, 2008). The encryption algorithm will produce a different ciphertext depending on the specific key being used. The corresponding key is needed to decrypt the ciphertext to plaintext (Hart, 1994). Zaidan, Zaidan, Al-Frajat, and Jalab (2010) have written that a key gives us flexibility in using an encryption algorithm and provides additional security. Zaidan et al. (2010) proposed that when the same key used for encryption and decryption as shown in Figure 1, they are called symmetric key, or secret key. The encryption process can be denoted as C = E (K, M); and the decryption process is denoted as M = D (K, C). The cryptosystem is called symmetric cryptosystem and needs to satisfy M = D (K, E (K, M)).

Design principles: Software design can be both defined as a process and a model. The design process is a set of iterative steps that enable the designer to describe all aspects of the software to be built. The basic design principles enable the software engineers to navigate through the design process.

Some of the design principles are:

- The design should be traceable.
- The design should not reinvent the wheel.
- The design should "minimize an intellectual distance between and the problem as it exists in the real world".
- The design should exhibit uniformity and integrity.

Identifying subsystems: Finding the subsystems during system design has many similarities to finding objects during analysis. Subsystem decomposition is constantly revised whenever new issues are addressed. Subsystems are merged into one subsystem, a complex subsystem is split into parts, and some subsystems are added to take care of new functionality. Subsystems found in the system are below.

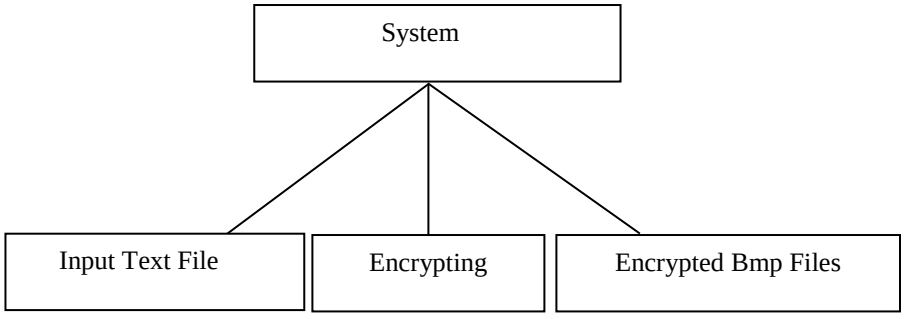

Figure 4.4: Steganography system as a set of three subsystems.

Current Software Architecture

Software Architecture includes the system decomposition, the global control flow, error-handling policies and inter-subsystem communication protocols.

The different architectures for the different systems are:

- Repository Architecture.

- Model/ View/ Controller (MVC).
- Client/ Server Architecture.
- Peer-To-Peer Architecture.
- Pipe and Filter Architecture.

In the Repository Architecture**,** subsystems access and modify data from a single data structure called central Repository. Subsystems are relatively independent and interact only through the central data structure. Control flow can be dictated either by the central repository or by the subsystems.

The Software Architecture of the proposed system can be decided from the above theory is Repository.

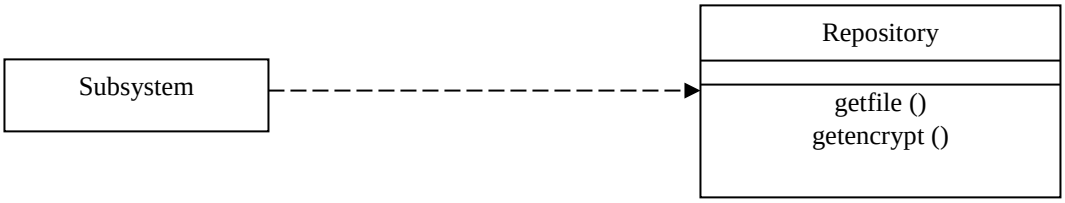

Figure 4.5: Repository software architecture for the proposed system.

Use Case Model

Use Case Diagrams, used during elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from an external point of view. A use case describes a function provided by the system that yields a visible result for an actor.

Actors are external entities that interact with the system. Examples of actors include a user role (e.g., a system administrator, a bank customer) or another system. Actors have unique names and descriptions. Use cases describe the behavior of the system as seen from an actor's point of view. The behavior described by the use case

model is also called external behavior. When actors and use cases exchange information, they are said to communicate.

To describe a use case, it uses a template composed of six fields:

- The name of the use case is unique across the system so that developers can unambiguously refer to the use case.
- Participating actors are the ones interacting with the use case.
- Entry conditions briefly describe the conditions that need to be satisfied before the use case is initiated.
- The flow of events describes the sequence of steps followed by the use case, which are numbered for references. The common use cases and the exception cases are described separately in different use cases for clarity.
- Specific requirements are the requirements that are not related to the functionality of the system. These include constraints on the performance of the system, its implementation, and the hardware platforms it runs on, and so on.

Use cases are written in natural language. This enables developers to use them for communicating with the client and the users, who generally do not have an extensive knowledge of software engineering notations.

In 2004, Fowler stated that rather than describe use cases head-on, it is easier to sneak up on them from behind and start by describing scenarios. A scenario is a sequence of steps describing an interaction between a user and a system.

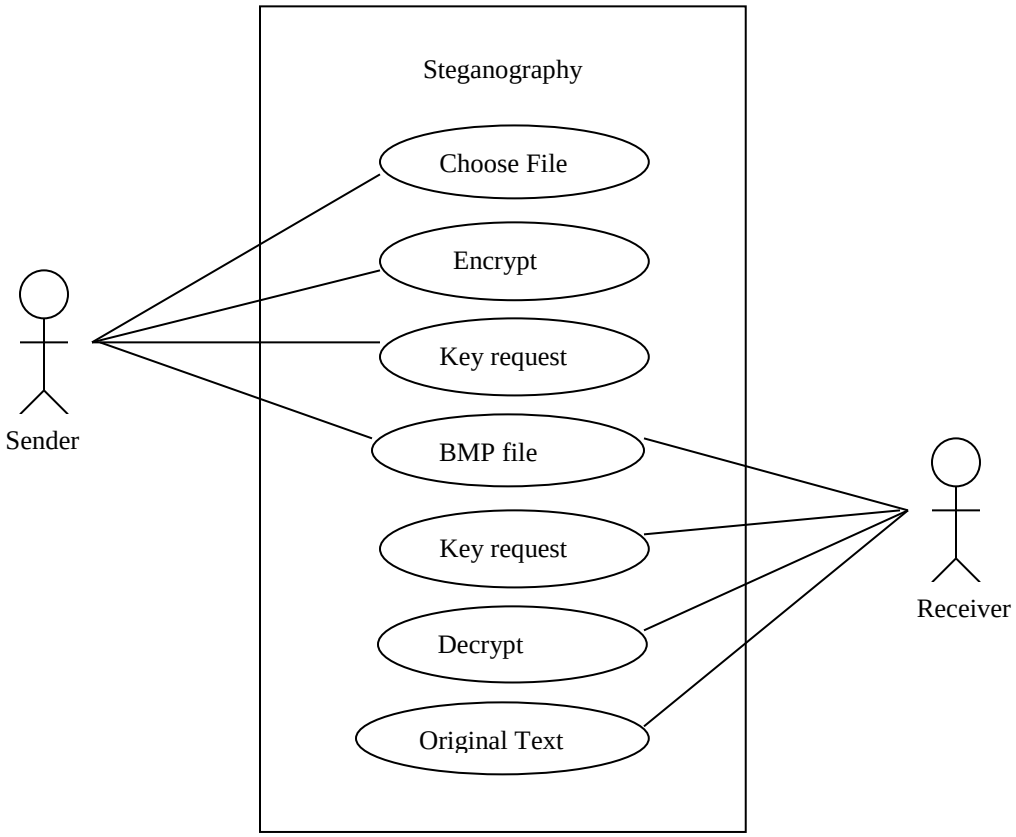

Figure 4.6: Use case for the proposed system (UML–Unified Modelling Language). (Adapted from Fowler, 2004).

Scenarios

A Scenario is a step-by-step detail description of what people do and experience as they try to make use of computer system and applications. A Scenario is solid, focused, informal description of a single feature of the system from the viewpoint of a single actor.

The several types of scenarios are:

- As-is Scenarios: These describe the current situation.
- Visionary Scenarios: These describe a future system.

- Evaluation Scenarios: These describe user talks against which the system is to be evaluated.
- Training Scenarios: These are tutorials used for introducing new users to the system.

Here are the different scenarios identified for the proposed system.

Table 4.1

Use Case Scenario (A)

Use case Name:	Convert text into image file
Participating Actors:	Initiated by the user, communicates with ‘Steganography” system.
Entry Condition	User selects the text file
Flow of Events	1. The user upon initiating, the system asks the user to enter the input parameter as a text file that must be sent. 2. Providing 16-bit secret key. 3. The text file is converting into an image file.
Exit Condition	Text file is converted into BMP image file

Table 4.2

Use Case Scenario (B)

Use case Name:	Encrypt text
Participating Actors:	Initiated by the user, communicates with ‘Steganography” system.
Entry Condition	The user selects the algorithm and enters key.
Flow of Events	1. The file is checked whether it is a text file or not. 2. The data in the text file is retrieved. 3. The user selects the encryption algorithm to encrypt the data present in an image file. 4. User encrypts the data by using the key given.
Exit Condition	The encrypted file (image file) is obtained.

Table 4.3

Use Case Scenario (C)

Use case Name:	Decrypt file
Participating Actors:	Initiated by the user, communicates with ‘Steganography” system.
Entry Condition	The encrypted file is taken as input.
Flow of Events	1. The encoded file is selected. 2. Again, provide the 16-bit secret key. 3. Decrypt the file using the same algorithm as used during encryption. 4. The image file is converted into a text file.
Exit Condition	The encrypted text file is obtained.

Class Diagrams

Class diagrams are used to describe the structure of the system. Classes are abstractions that specify the common structure and behavior of a set of objects. Objects are instances of classes that are created, modified, and destroyed during the execution of the system.

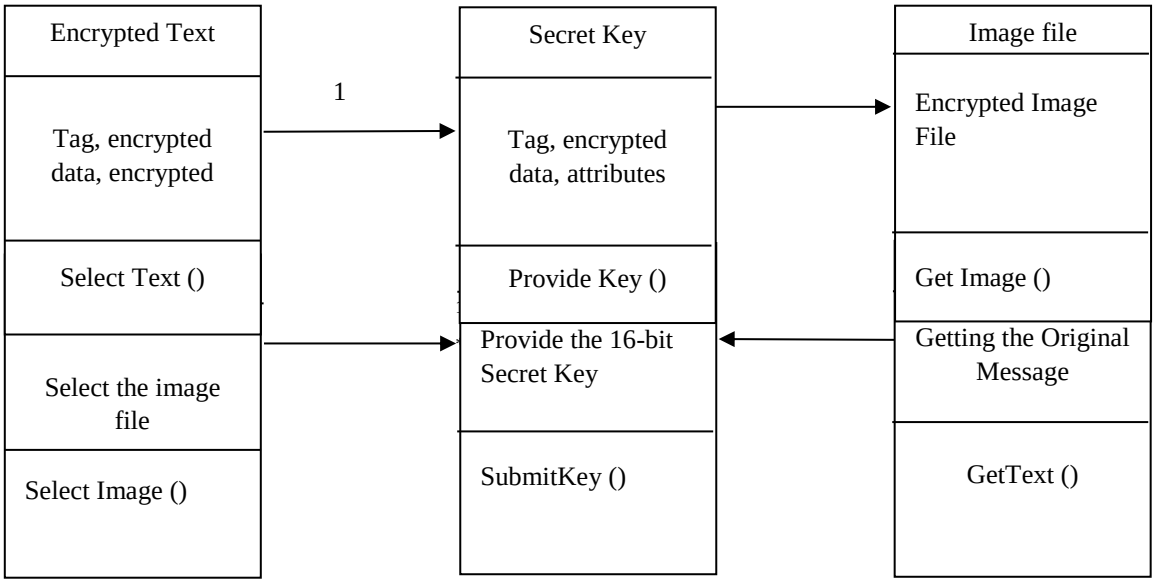

Figure 4.7: UML class diagram for the system.

User interface

The user will be given the main form, which displays different algorithms to be implemented. The user selects any of these entire algorithms and supplies key to the algorithm to encrypt the file. Upon computing, the system should display the encrypted text file.

Identification of Objects

Entity objects: Entity objects represent the persistent information tracked by the system. The entity objects identified for the proposed system are:

- User
- Text Form

Boundary objects: These represent the interactions between the actors and the system. The boundary objects identified for the proposed system are:

- Main form
- Select file Form
- Key form

Control objects: These represent the tasks that are performed by the user and supported by the system. The control objects identified for the proposed system are:

- Encryption algorithm
- Decryption algorithm
- Key

Activity diagrams: Activity diagram describes how a system works and its workflow. Activity diagrams can show activities that are both conditional and parallel**.** The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams show the Flow of activities through the system.

Activity diagrams flow from top to bottom and have branches to describe conditional and parallel activities.

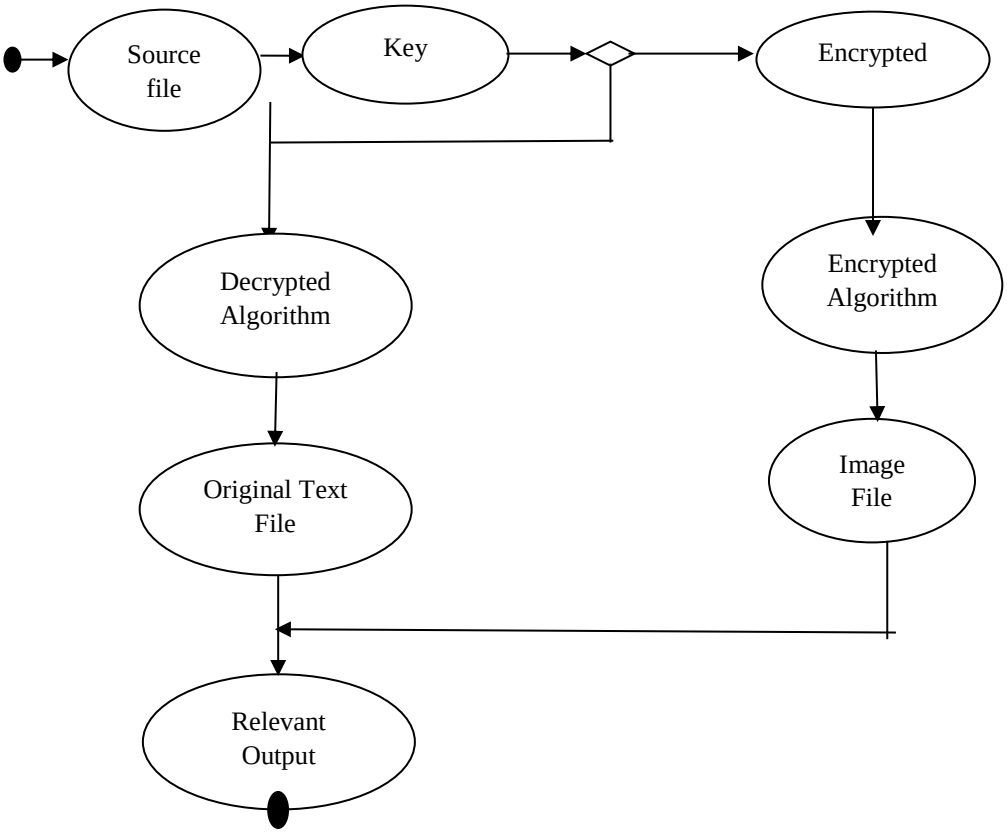

Figure 4.8: Activity diagram for the proposed system.

Figure 4.8 describes that in the proposed system, a text file is taken in the source end. Conversion is performed to create an image file. From there two activates can be performed depending on the user's selection. In one phase, encryption goes with encrypting algorithm with the help of a key. It will be converting into an image file. In the other phase decrypting the file folme key on the image file**.**

User Interface Outline

The User Interface used in the proposed system can be seen in Figure 4.9 below. It is created in JAVA programming and has a homepage that shows the drop-down menu File and the workspace.

Figure 4.9: User interface of the system.

Encryption

The Figure 4.10 below shows the menu items in the drop-down menu of File. There are six options to select from. The detailed description of each option is provided below.

1. *Open Text File:* It is an option that helps a user to select the text file that user wants to encrypt.
2. *Open:* This option allows a user to select an image file that user wants to use to embed user text into it.
3. *Save:* The save option allows a user to save the encrypted image in the system and name as user wish.
4. *Decrypt:* This allows a user to select the encrypted image and decrypt it.
5. *Clear:* This option allows a user to clear the homepage at any point during the encryption and decryption process.
6. *Exit:* This option will take the user out of the program.

The sequence in which the system works is as follows:

1. The user needs to select a text file that has already been created in the system and is ready to be encrypted using the "Open Text File" option from the File drop-down menu.

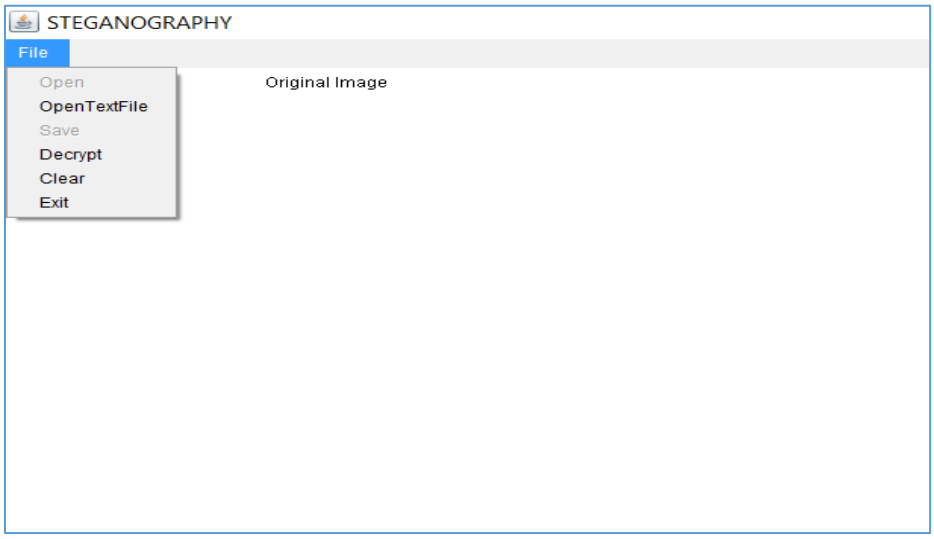

Figure 4.10: Drop down menu of the system.

2. Then the system generates a pop-up and tells a user to enter the 16- bit secret key. Press "OK" after entering the key. In Figure 4.11 it can see the pop-up that asks the user to enter the 16-bit key for encryption.

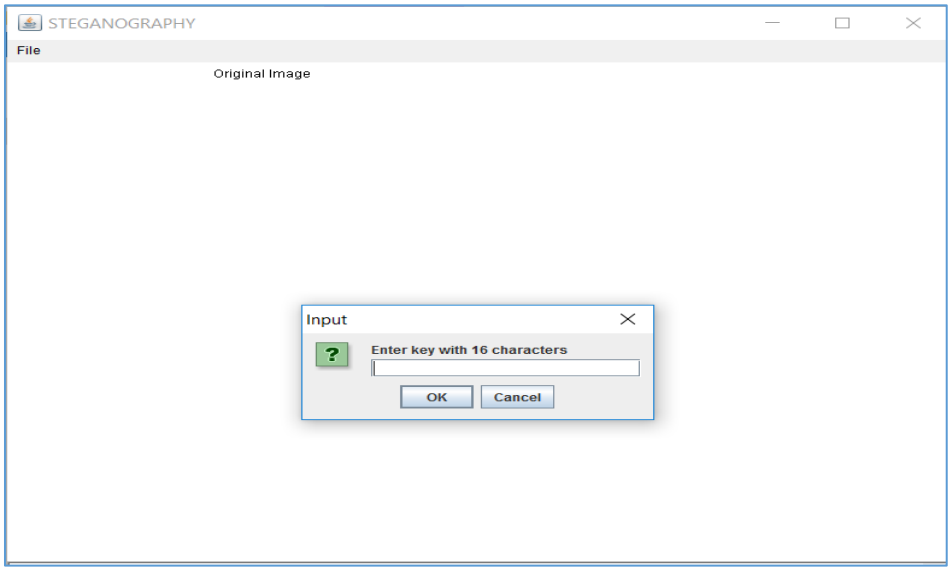

Figure 4.11: Enter secret key pop-up.

3. Then in Figure 4.12 below the user can see that it displays the text written in the text file in the area below.

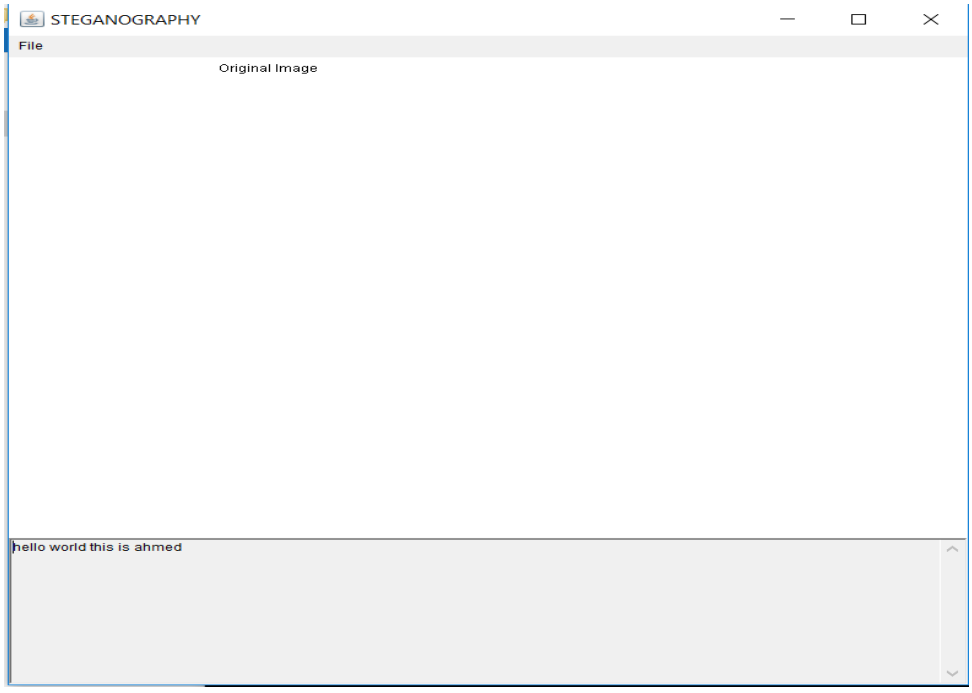

Figure 4.12: "Text to be encrypted" screen.

4. Then select "Open" from file drop-down, a user selects an image that user wants to use for encryption. After selecting the picture, it shows the original and encrypted images side by side as a user can see in Figure 4.13 below. There is no difference in both images that can be detected by the naked eye.

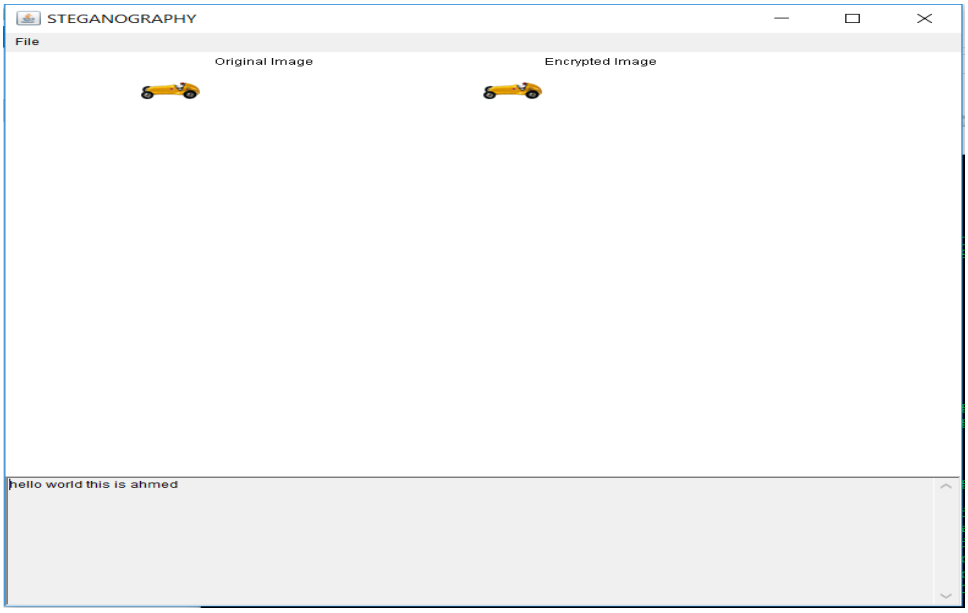

Figure 4.13: Original & encrypted image along with text encrypted.

5. Then save the encrypted image by selecting "Save" from the drop-down menu and save it anywhere in the system.

Decryption

1. For decryption user needs to select the "Decrypt" option from the drop-down menu then a pop-up appears as shown in Figure 4.14 below that tells a user to input the secret key used during encryption.

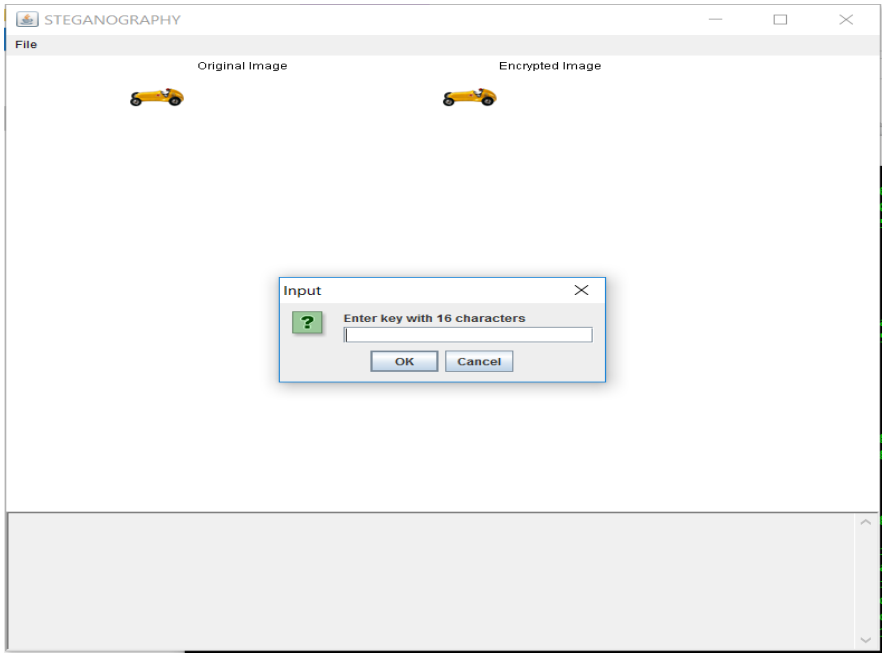

Figure 4.14: Secret key pop-up for decrypting an image.

2. After entering the secret key, as shown in Figure 4.15 below the system will show a user the decrypted and original images along with the text message at the bottom of the page.

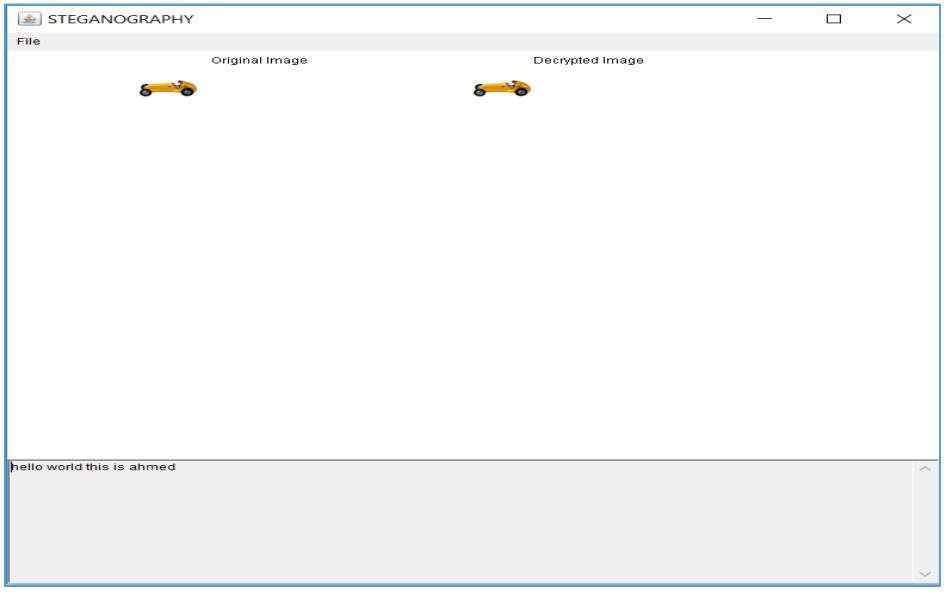

Figure 4.15: Decrypted text along with the image.

Experimental Analysis

The proposed method was experimented using JAVA code. The plaintext is first encrypted to generate the ciphertext using substitution cipher method. A key is used in the encryption which is based on symmetric cryptosystem where the same key is used for both encryption and decryption process. Then the ciphertext is embedded inside the BMP image file using encryption technique that embeds the information in the image domain. The generated stego-image is sent over to the intended recipient. The whole idea of the proposed method is to model a technique that enables secure data communication between sender and receiver (Laskar and Hemachandran, 2012c). By this approach, the messages were successfully embedded into the cover images.

In the experiment messages of varied sizes were successfully embedded into a different set of images of various ranges. In the method of retrieval, the message is first extracted from the stego-image. The message is then decrypted by producing the key used in encryption to get back the original message. If the key does not match the original information will remain unreadable.

The pictures below are set of original and encrypted images from the test.

Figure 4.16: Original and stego image of railway track.

The image on the left-hand side is the original image of a railway track and the image on the right-hand side is the stego image of a railway track that is obtained after encryption.

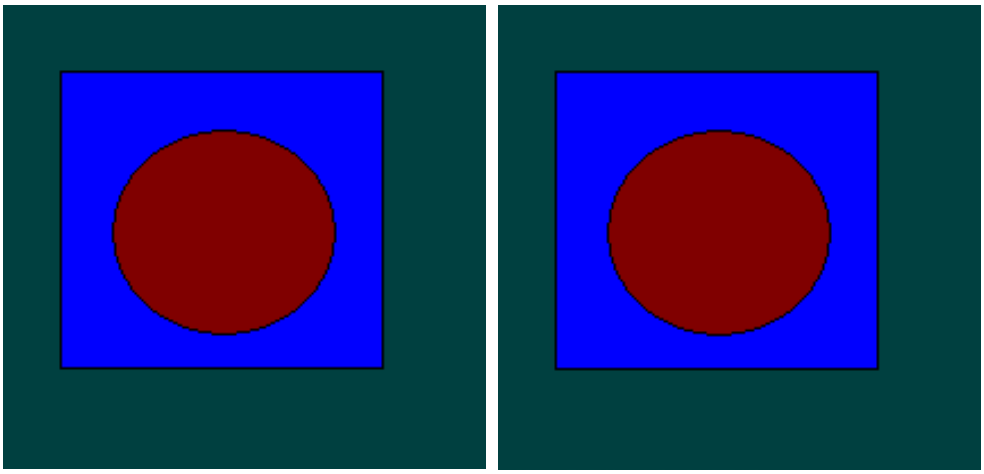

Figure 4.17: Original and stego image of cube.

The image on the left-hand side is the original image of the cube and the image on the right-hand side is the stego image of the cube that is obtained after encryption.

Figure 4.18: Original and stego image of a flower.

The image on the left-hand side is the original image of a flower and the image on the right-hand side is the stego image of a flower that is obtained after encryption.

The messages that were embedded into the images were extracted successfully. It is observed that the human visual system (HVS) cannot distinguish the cover-image and stego image the complexity of the image is not disturbed as shown in Figures 4.16, 4.17, and 4.18. The work not only aims to preserve the visual integrity of the image used for embedding but also the method should be free from statistical attacks because with the advances in steganalysis technique various statistical methods can detect modification in image bits. So, distortion analysis of stego images is carried out by studying distortion/similarity statistically.

It can test the algorithm using the PSNR (Peak signal-to-noise ratio). PSNR is a standard measurement used in steganography technique to test the quality of the stego images. The higher the value of PSNR, the more quality the stego image will have. If the cover image is *C* of size *M* × *M* and the stego image is *S* of size *N* × *N*, then each cover image *C* and stego image *S* will have pixel value *(x, y)* from *0* to *M-1*and *0* to *N-1* respectively. The PSNR is then calculated as follows:

$$PSNR = 10.\log_{10}\left(\frac{MAX^2}{MSE}\right)$$

Where

$$MSE = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(C(x,y) - S(x,y))^2$$

MAX is the maximum pixel value of the images. Here, the pixels are represented using the 8 bits per sample, so the MAX value is 255.If the stego image has a higher PSNR value, and then the stego image has more quality image

Table 4.4

Result Comparison using PSNR Technique

Cover Image	Stego Image	MSE %	PSNR (dB)
Track	Steg_track	4.56%	41.54 DB
Cube	Steg_cube	6.10%	40.27 DB
Flower	Steg_flower	3.19%	43.09 DB

When the payload increases, the MSE increases, and this affects the PSNR inversely (Li & Wang, 2007). So, from trade-off, it was found that MSE decrease causes PSNR increase and vice-versa. PSNR is often expressed on a logarithmic scale in decibels (dB). In 2013, Laskar and Hemachandran quoted that “PSNR values falling below 30 dB indicate a low quality, i.e., distortion caused by embedding can be obvious”. However, a high-quality stego-image should strive for 40 dB and above (Carvajal-Gamez, Gallegos-Funes, & Lopez-Bonilla., 2009).

The results indicate that embedding process introduces less perceptual distortion and higher PSNR (Ulutas, Ulutas, & Nabiyev, 2011). It is to be noted that PSNR ranging from 41 dB to 43 dB means that the quality degradations could hardly be perceived by a human eye.

The pixels of the cover image must fulfill the minimum requirement for the process of data hiding. The minimum image pixel for width is at least 150 while the minimum image pixel for height is at least 112 (Ibrahim & Kuan, 2011).

The pixel and size comparison of test objects or images are shown below. In these pictures, the pixel width and height are well above the required values.

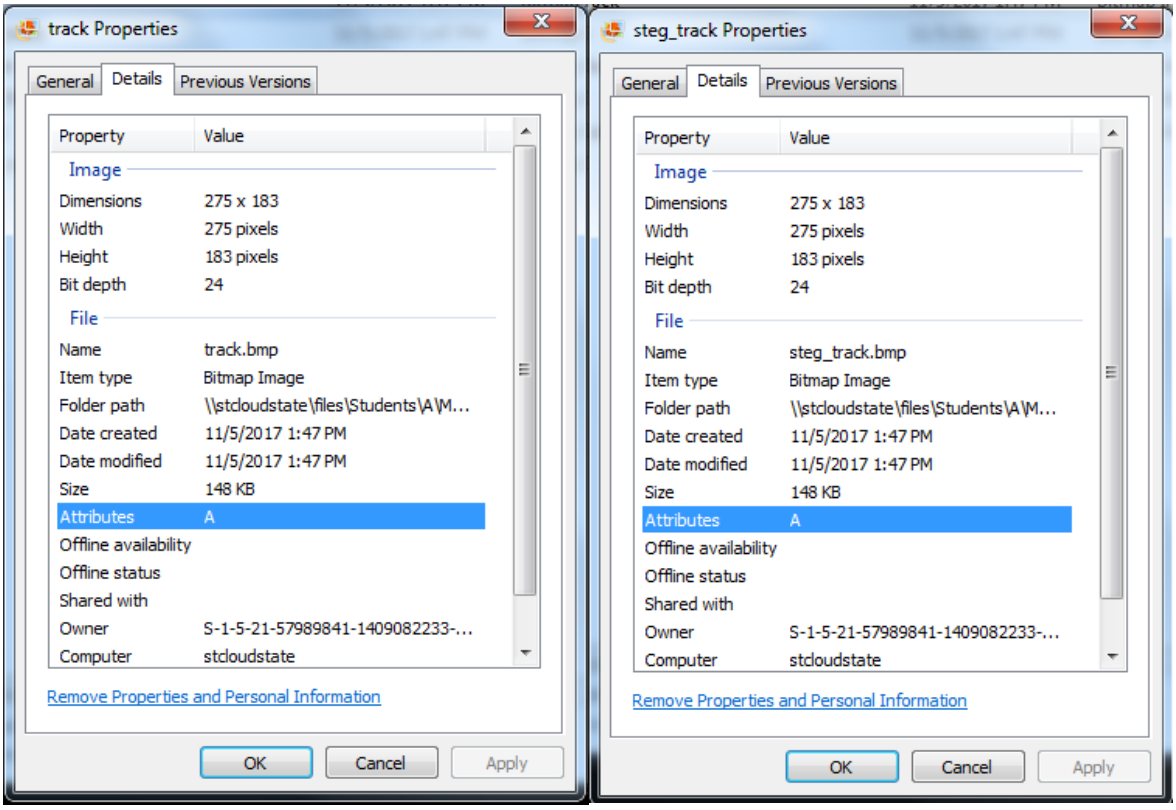

Figure 4.19: Original and stego image properties of a track.

The image on the left-hand side is the original image properties of a railway track and the image on the right-hand side is the stego image properties of a railway track that is obtained after encryption.

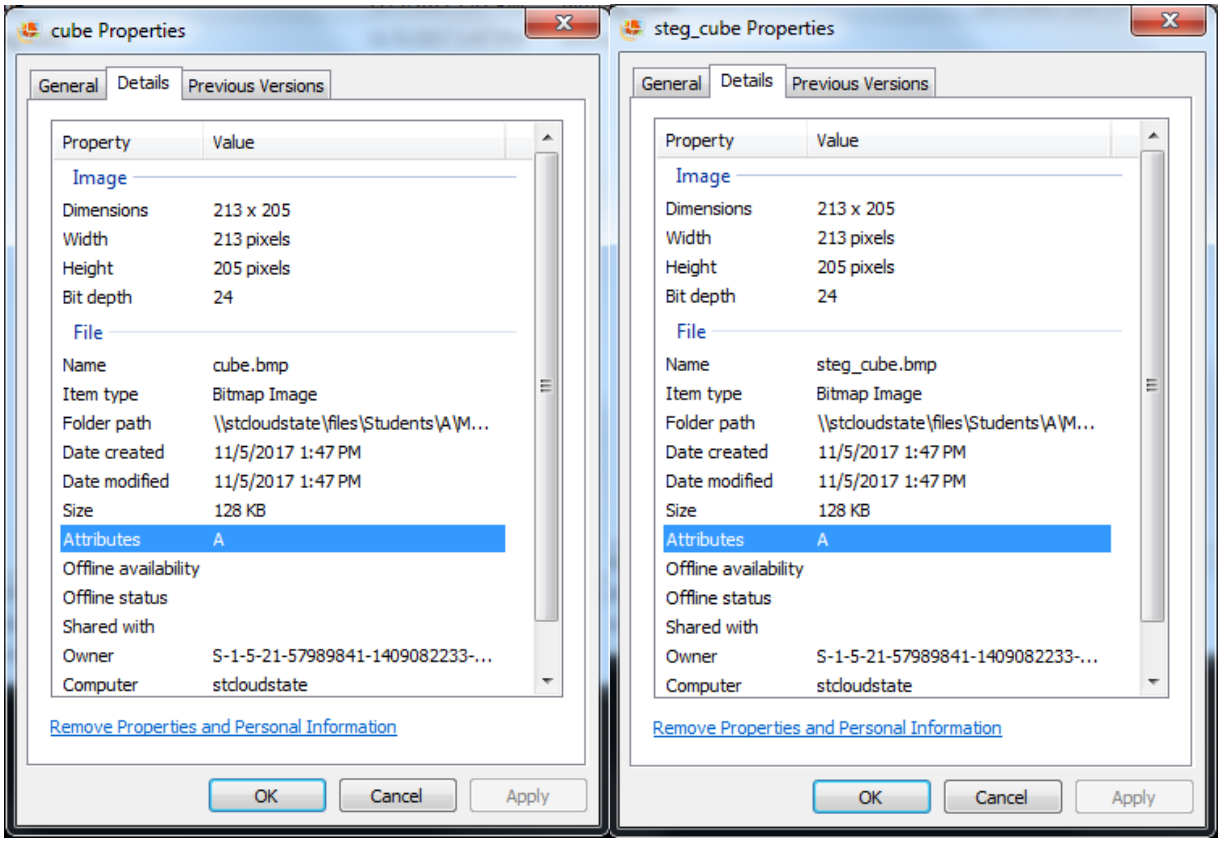

Figure 4.20: Original and stego image properties of a cube.

The image on the left-hand side is the original image properties of a cube and the image on the right-hand side is the stego image properties of a cube that is obtained after encryption.

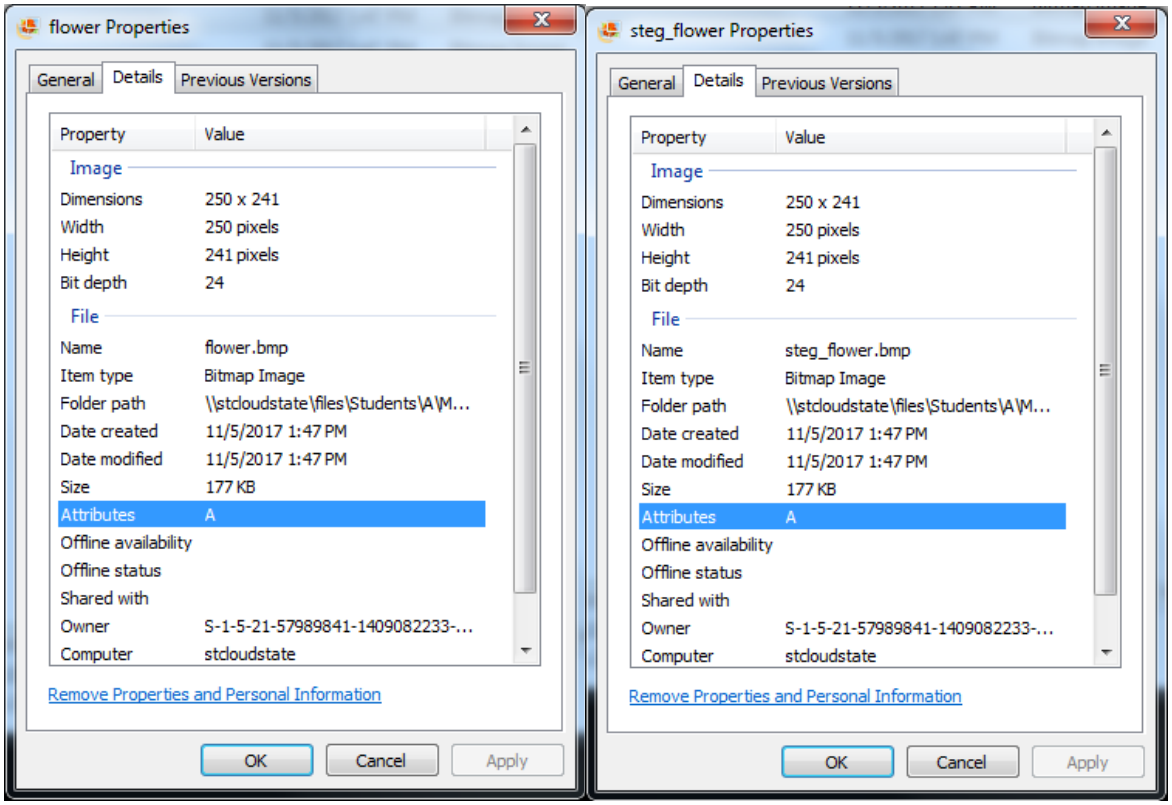

Figure 4.21: Original and stego image properties of a flower.

The image on the left-hand side is the original image properties of flower and the image on the right-hand side is the stego image properties of flower that is obtained after encryption.

The image file format used in the proposed algorithm is focused on bitmap (BMP) format. The BMP image format handles graphics files within the Microsoft Windows OS and is usually uncompressed. Hence, they are large (Ibrahim & Kuan, 2011). The major advantage of using BMP files is the simplicity and wide acceptance of BMP files in Windows programs. Thus, BMP type of image format is chosen to be used in our IDEA algorithm. Since BMP image has a relatively large size, the pixels in an image are relatively large as well. So, it provides more space for binary codes to be encoded

within the image. To increase as much as characters that can be hidden, zip technique is used to reduce the total size of a file and to enhance the security of the file.

Summary

In this implementation phase, there were various phases of execution. The JAVA code was written in the implementation phase. The execution of the program took place and was successfully able to embed the text information into an image file. The encryption and decryption were performed and tested thoroughly. The image properties were kept stable and the result was as expected. The text was encrypted into the image file and able to be decrypted only using the secret used during the encryption process.

Chapter V: Discussion

Study Questions

1. *Is Steganography the best and reliable technique to ensure the security of the secret message without any alterations to the message?*

Steganography is the most reliable and secure way to embed a message into an image, without creating any kind of distortion or change to the image. It retains the original image properties in the stego image as well. Pixel values also remains same.

2. *Can the combination of both steganography and cryptography be able to create a secure and efficient way to transfer data over the web?*

The combination of steganography and cryptography created a very well secured system to transfer the files over the web applications.

Time for program to run with different image sizes

Table 5.1

Program Response Time

Size of the Image	Time for program to Run
121 kb	5 seconds
148 kb	9 seconds
191 kb	15 seconds
500 kb	30 seconds
1 Mb	50 seconds
1.55 Mb	75 seconds
2.1 Mb	95 seconds
2.5 Mb	120 seconds

The table above describes the time taken by the program to compile and run a specific size image. The time taken increases as the size of the image increases the execution time and size of the image are inversely proportional to each other.

Limitations

There are certain limitations to the program and proposed a system in this paper. The list of limitations is as follows:

- The size of the image used during the encryption process should not be more than 2.5 Mb. If the size increases the process becomes time-consuming and the program takes a lot of time to execute it. In this case, results are limited to 2.5 Mb and a program was not checked for the image size more than that. So, this could be one of the further studies that can be done on this paper.
- The use of a symmetric key in this proposed paper is due to open source availability of the symmetric key and the limited resources in hand. Since asymmetric key or public key is a licensed key/product it could not be used in this program. In a public/asymmetric key encryption system, any person can encrypt a message using the public key of the receiver, but such a message can be decrypted only with the receiver's private key**.** Thus, this can also be one of the further studies that can be done on this paper.
- The proposed system does not work on iOS and Linux systems as it is developed for Windows software only.
- If the Windows operating system is older than the Windows 7, the proposed system does not work on that system

Chapter VI: Summary and Conclusion

Summary

The proposed paper here is completed and the total proposed system has been implemented and tested. The research related to the theory of IDEA algorithm, cryptography and steganography of image files is been done and presented in this paper. The practical approach, analyzing and implementation of the algorithm along with the coding took around 6 months to complete as shown in Figure 6.1.

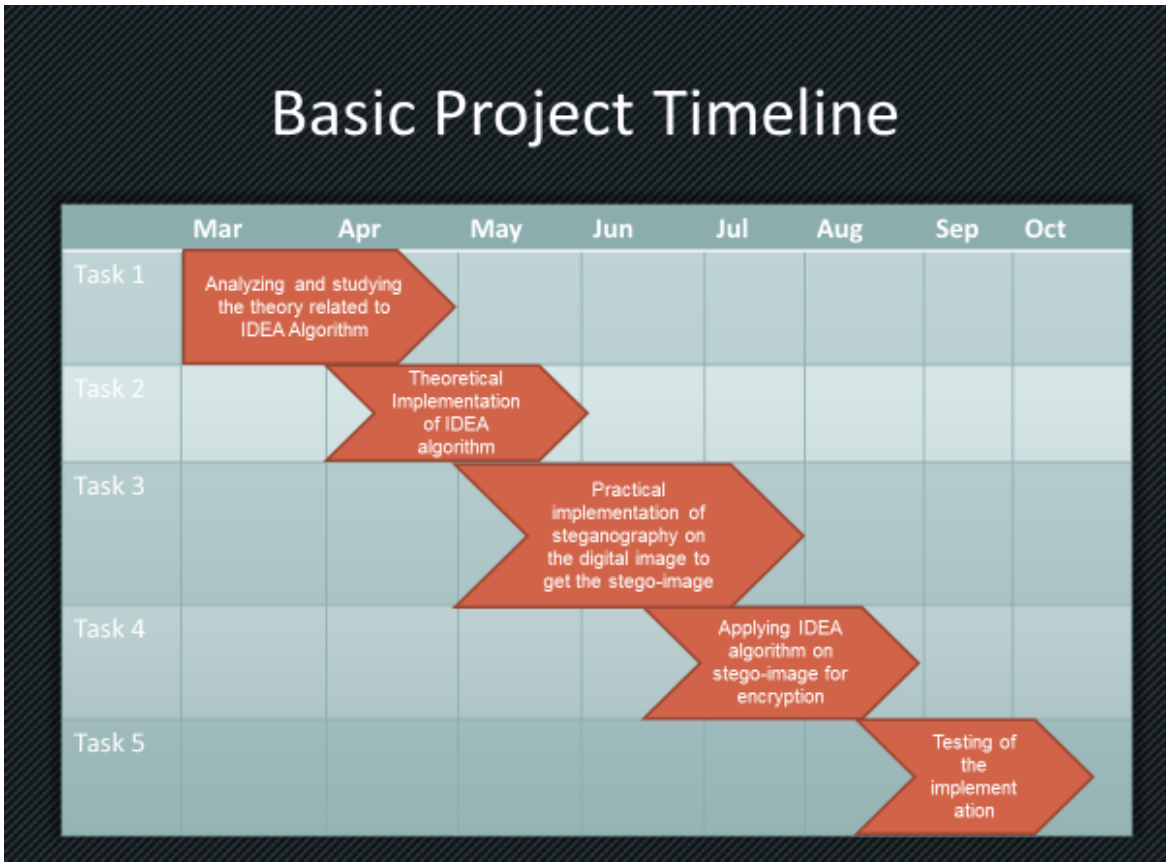

Figure 6.1: Basic project timeline.

Conclusion

This paper proposed a system with a steganography algorithm with two layers of security. A system has been developed using the proposed IDEA algorithm. It is tested with few images of various sizes of data to be hidden. In the proposed algorithm, it is found that the stego image does not have a distortion on it that can be seen by the naked eyes. Testing of stego images using PSNR value was also done. According to the PSNR value of each image, the stego image has a higher PSNR value. Hence this IDEA algorithm for steganography is very efficient to hide the data inside the image.

The combination method of steganography and cryptography will content the requirements such as reliability, security, and robustness for a secured data transmission over web applications. These combined techniques can be used and can replace the current security techniques with the amazing growth in security and power, the increase in security awareness among the individuals, groups, agencies, a government organization.

Thus, the proposed method allows users to send data through the network in a secured fashion and it can be employed for applications that require high-volume embedding with robust against attacks. The steganography method may be further secured if it compresses the secret message first and then encrypt it and then finally embed inside in the cover image.

References

Abed, I. (2010). Finding the best key stream by using genetic algorithm for image encryption. *Journal of Basrah Researches (Sciences), 36*(3), 72-80.

Anderson, R. (1989). Cryptanalytig properties of short substitution ciphers. *Cryptologia, 13*(1), 61-72.

Anderson, R. J., & Petitcolas, F. A. (1998). On the limits of steganography. *IEEE Journal on Selected Areas in Communications, 16*(4), 474-481.

Aura, T. (1996). Practical invisibility in digital communication. In R. Anderson (Ed.), *Information hiding* (pp. 265-278). Proceedings of First International Workshop, Cambridge, U.K., May 30-June 1. Berlin, Heidelberg: Springer Verlage.

Bhowmik, S., & Acharyya, S. (2011). Image cryptography: The genetic algorithm approach. 2011 IEEE International Conference Computer Science and Automation Engineering (CSAE), pp. 223-227.

Biryukov, A., Nakahara Jr., J., Preneel, B., & Vandewalle, J. (2002). New weak-key classes of IDEA. In R. Deng, F. Bao, J. Zhou, & S. Qing (Eds.), *Information and communications security* (pp. 315-326). ICICS 2002. Lecture Notes in Computer Science, Vol. 2513. Berlin, Heidelberg: Springer.

Bloisi, D. D., & Iocchi, L. (2007). Image based steganography and cryptography. *Visapp*, (1), 127-134.

Bozesan, A., Opritoiu, F., & Vladutiu, M. (2013). Hardware implementation of the IDEA NXT crypto-algorithm. *2013 IEEE 19th International Symposium for Design and Technology in Electronic Packaging (SIITME),* pp. 35-38.

Carvajal-Gamez, B., Gallegos-Funes, F., & Lopez-Bonilla, J. (2009). Scaling factor for RGB images to steganography applications. *Journal of Vectorial Relativity, 4*(3), 55-65.

Chang, C., & Tseng, H. (2004). A steganographic method for digital images using side match. *Pattern Recognition Letters, 25*(12), 1431-1437.

Chen, P.-Y., Wu, W.-E. (2009). A modified side match scheme for image steganography. *International Journal of Applied Science and Engineering 7(1),* 53-60.

Cheung, O. Y., Tsoi, K. H., Leong, P. H. W., & Leong, M. (2001). Tradeoffs in parallel and serial implementations of the international data encryption algorithm IDEA. *International Workshop on Cryptographic Hardware and Embedded Systems,* pp. 333-347.

Cole, E. (2003). Hiding in plain sight. *Steganography and the Art of Covert Communication.* Indianapolis, IN: Wiley Publishers, Inc.

Conway, M. (2003). Code wars: Steganography, signals intelligence, and terrorism. *Knowledge, Technology & Policy, 16*(2), 45-62.

Daemen, J., Govaerts, R., & Vandewalle, J. (1993). Weak keys for IDEA. *Annual International Cryptology Conference,* pp. 224-231.

Eggers, J. J., Baeuml, R., & Girod, B. (2002). Communications approach to image steganography. *Electronic Imaging,* pp. 26-37.

El-Emam, N. N. (2007). Hiding a large amount of data with high security using steganography algorithm. *Journal of Computer Science, 3*(4).

Fadhil, M. A. (2010, October). A novel steganography-cryptography system. In *Proceedings of the World Congress on Engineering and Computer Science. USA*, Vol. I, ISSN: 2078-0966.

Fairley, R. (1985). *Software engineering concepts.* New York: McGraw-Hill, Inc.

Fowler, M. (2004). *UML distilled: A brief guide to the standard object modeling language.* Boston, MA: Addison-Itsley Professional.

Fridrich, J., Goljan, M., & Du, R. (2001). Steganalysis based on JPEG compatibility. *ITCom 2001: International Symposium on the Convergence of IT and Communications,* pp. 275-280.

Friedman, W. F. (1967). Cryptology. *Encyclopedia Britannica, 6*, 844-851.

Ghosh, R. A. (2012). A modified improved text encryption approach inspired by genetic algorithm techniques using RSA algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering, 2*(6), 263-268.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Reading. MA: Addison-Itsley, Publishers.

Goyat, J. (2012). *Using 3D/4D CAD modeling for traffic management: Development review, and communication*. (Unpublished thesis), University of Texas at Austin.

Gupta, N., & Sharma, N. (2013). Hiding image in audio using DWT and LSB. *International Journal of Computer Applications, 81*(2).

Hart, G. W. (1994). To decode short cryptograms. *Commun. ACM, 37*(9), 102-108. doi:10.1145/182987.184078

Hawkes, P. (1998). Differential-linear weak key classes of IDEA. *Advances in Cryptology—EUROCRYPT'98,* pp. 112-126.

Hosmer, C. (2006). Discovering hidden evidence. *Journal of Digital Forensic Practice, 1*(1), 47-56.

Ibrahim, R., & Kuan, T. S. (2011). Steganography algorithm to hide secret message inside an image. *Computer Technology and Application, 2*, 102-108

International Data Encryption Algorithm. (n.d.). In *Wikipedia*. Retrieved August 10, 2013, from https://en.wikipedia.org/wiki/International_Data_Encryption_Algorithm.

Jahnke, T., & Seitz, J. (2008). An introduction in digital watermarking: Applications, principles, and problems. In *Information security and ethics: Concepts, methodologies, tools, and applications* (pp. 554-569). Hershey, PA: IGI Global.

Johnson, N. F., & Jajodia, S. (1998). Exploring steganography: Seeing the unseen. *Computer, 31*(2).

Kahate, A. (2008). *Cryptography and network security.* New York: Tata McGraw-Hill Education.

Kaijser, P., Parker, T., & Pinkas, D. (1994). III: Security applications: SESAME: The solution to security for open distributed systems. *Computer and Communication Security, 17*(7), 501-518. doi:10.1016/0140-3664(94)90105-8

Kaur, M., Gupta, S., Sandhu, P. S., & Kaur, J. (2010). A dynamic RGB intensity-based steganography scheme. *World Academy of Science, Engineering and Technology, 67*, 833-838.

Khare, P., Singh, J., & Tiwari, M. (2011). Digital image steganography. *Journal of Engineering Research and Studies, 2*(3), 101-104.

Kharrazi, M., Sencar, H. T., & Memon, N. (2006). Performance study of common image steganography and steganalysis techniques. *Journal of Electronic Imaging, 15*(4), 041104-041104.

Lai, X., & Massey, J. L. (1990). A proposal for a new block encryption standard. *Workshop on the Theory and Application of Cryptographic Techniques,* pp. 389-404.

Laskar, S. A., & Hemachandran, K. (2012a). *Combining JPEG steganography and substitution encryption for secure data communication* Computer Science \& Information Technology (CS \& IT): ISSN.

Laskar, S. A., & Hemachandran, K. (2012b). High capacity data hiding using LSB steganography and encryption. *International Journal of Database Management Systems, 4*(6), 57.

Laskar, S. A., & Hemachandran, K. (2012c). Secure data transmission using steganography and encryption. *International Journal on Cryptography and Information Security, 2*(3), 161-172.

Laskar, S. A., & Hemachandran, K. (2013). Steganography based on random pixel selection for efficient data hiding. *International Journal of Computer Engineering and Technology, 4*(2), 31-44.

Leong, M., Cheung, O. Y., Tsoi, K. H., & Leong, P. H. W. (2000). A bit-serial implementation of the international data encryption algorithm IDEA. *Field-*

Programmable Custom Computing Machines, 2000 IEEE Symposium on, pp. 122-131.

Li, B., He, J., Huang, J., & Shi, Y. Q. (2011). A survey on image steganography and steganalysis. *Journal of Information Hiding and Multimedia Signal Processing, 2*(2), 142-172.

Li, X., & Wang, J. (2007). A steganographic method based upon JPEG and particle swarm optimization algorithm. *Information Sciences, 177*(15), 3099-3109.

Liu, C., & Liao, S. (2008). High-performance JPEG steganography using complementary embedding strategy. *Pattern Recognition, 41*(9), 2945-2955.

McLaughlin, L. (2006). Philip Zimmerman on what's next after PGP. *IEEE Security \& Privacy, 4*(1), 10-13.

Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of applied cryptography.* CRC Press.

Mollin, R. A. (2005). *Codes: The guide to secrecy from ancient to modern times.* CRC Press.

Pekmestzi, K., Efstathiou, C., Moschopoulos, N., & Tsoumanis, K. (2013). Efficient modulo 2 n+ 1 multiplications for the idea block cipher. *Proceedings of the 23rd ACM International Conference on Great Lakes Symposium on VLSI,* pp. 263-268.

Petitcolas, F. A., Anderson, R. J., & Kuhn, M. G. (1999). Information hiding-a survey. *Proceedings of the IEEE, 87*(7), 1062-1078.

Pfitzmann, B. (1996). Information hiding terminology-results of an informal plenary meeting and additional proposals. *Proceedings of the First International Workshop on Information Hiding,* pp. 347-350.

Ramesh, R., Athithan, G., & Thiruvengadam, K. (1993). An automated approach to solve simple substitution ciphers. *Cryptologia, 17*(2), 202-218.

Raphael, A. J., & Sundaram, V. (2011). Secured crypto-stegano communication through unicode ǁ. *World of Computer Science and Information Technology Journal, 1*(4), 138-143, 2221-0741.

Ravi, S., & Knight, K. (2009). Attacking letter substitution ciphers with integer programming. *Cryptologia, 33,* 321-334.

Schneier, B. (2007). *Applied cryptography: Protocols, algorithms, and source code in C.* John Wiley & Sons.

Schneier, B. (2011). *Secrets and lies: Digital security in a networked world.* John Wiley & Sons.

Sharma, V., & Kumar, S. (2013). A new approach to hide text in images using steganography. *International Journal of Advanced Research in Computer Science and Software Engineering, 3*(4).

Simmons, G. J. (1994). Subliminal channels; past and present. *Transactions on Emerging Telecommunications Technologies, 5*(4), 459-474.

Song, S., Zhang, J., Liao, X., Du, J., & Wen, Q. (2011). A novel secure communication protocol combining steganography and cryptography. *Procedia Engineering, 15*, 2767-2772.

Stallings, W. (2006). *Cryptography and network security: Principles and practices.* Pearson Education India.

Ulutas, G., Ulutas, M., & Nabiyev, V. (2011). Distortion free geometry based secret image sharing. *Procedia Computer Science, 3*, 721-726.

Walia, E., Jain, P., & Navdeep, N. (2010). An analysis of LSB \& DCT based steganography. *Global Journal of Computer Science and Technology, 10*(1).

Warkentin, M., Bekkering, E., & Schmidt, M. B. (2008). Steganography: Forensic, security, and legal issues. *The Journal of Digital Forensics, Security and Law: JDFSL, 3*(2), 17.

Watson, A. B. (1994). Image compression using the discrete cosine transform. *Mathematica Journal, 4*(1), 81.

Watson, A. B. (1994). Perceptual optimization of DCT color quantization matrices. *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference, 1,* 100-104.

Watters, P., Martin, F., & Stripf, H. S. (2008). Visual detection of LSB-encoded natural image steganography. *ACM Transactions on Applied Perception (TAP), 5*(1), 5.

Wu, D., & Tsai, W. (2003). A steganographic method for images by pixel-value differencing. *Pattern Recognition Letters, 24*(9), 1613-1626.

Zaidan, B., Zaidan, A., Al-Frajat, A., & Jalab, H. (2010). On the differences between hiding information and cryptography techniques: An overview. *Journal of Applied Sciences, 10*, 1650-1655.

Zhu, D. (2010). An attack on 5.5-round IDEA. *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on, 2,* 18-21.

Appendix

The Java Coding Used in the Execution of the Starred Paper

```
import java.awt.*;
import java.io.*;
import java.awt.event.*;
import javax.swing.JOptionPane.*;
import javax.swing.*;
import java.applet.*;
import java.awt.image.*;
```

// It is the main class

```
public  class  steg extends Frame implements ActionListener
        {
        static  String path;
        static  String path1;
        FileDialog fd,fd1;
        public  Image img=null;
         public Image img1=null;
         public Image img3=null;
        static steg f;
        int index=0;
        int max;
        int  bytes[]=new int[10000];
```

```
  int binary[]=new int[2450000];
  int maxbinary,maxbinary1,maxbinary2,ivalue;
  String msg;

String filename,filename1;
 int flag=0;
        public int pixels[]=new int[350000];
        public int pixels1[]=new int[350000];
        static int width,height,padding;
        static int  index1=0,ch,i=0,l=0,r,g,b,r1,g1,b1;
        int p[]=new int[2000000];
        int p1[]=new int[2000000];
        int p2[]=new int[2000000];
        int p3[]=new int[2000000];
        int pix[]=new int[350000];
        int pix1[]=new int[350000];
        int blue[]=new int[300];
        int green[]=new int[300];
        int red[]=new int[300];
    int maxp;
    int maxp1;
    TextArea ta=new TextArea(10,100);
    MenuBar mb=new MenuBar();
```

```
  Menu m1=new Menu("File  ");
  MenuItem mi1=new MenuItem("Open");
  MenuItem mi2=new MenuItem("OpenTextFile");
   MenuItem mi3=new MenuItem("Save");
   MenuItem mi4=new MenuItem("Decrypt");
   MenuItem mi5=new MenuItem("Reset");
   MenuItem mi6=new MenuItem("Exit");
steg()
        {
    setMenuBar(mb);
    m1.add(mi1);
    mi1.setEnabled(false);
   m1.add(mi2);
    m1.add(mi3);
    mi3.setEnabled(false);
    m1.add(mi4);
    m1.add(mi5);
    m1.add(mi6);
   mb.add(m1);
   mi1.addActionListener(this);
   mi2.addActionListener(this);
    mi3.addActionListener(this);
    mi4.addActionListener(this);
```

```
            mi5.addActionListener(this);
            mi6.addActionListener(this);
    setLayout(new BorderLayout( ));
    add(ta,BorderLayout.SOUTH);
    addWindowListener(new WindowAdapter()
          {
             public void WindowClosing(Windoitvent e)
                {
                     System.exit(0);
                 }
           });
         }
```

// It is the main function

```
public static void main(String args[])
{
             f=new steg();
            f.setSize(800,800);
            f.setTitle("STEGANOGRAPHY ");
             f.show();
       }

public void actionPerformed(ActionEvent ae)
```

```
{
        String s=ae.getActionCommand();
                        // It opens the Bitmap image file
      if(s.equals("Open"))
                {
                        msg="Encrypted Image";
                        flag=0;
                        fd=new FileDialog(f,"Opening Files.....");
                         fd.show();
                         path=fd.getDirectory()+fd.getFile();
                                  if(path!=null)
                                     {
                    mi1.setEnabled(false);
                    mi3.setEnabled(true);
                         openimagefile(path,11);
                         repaint();
                                  }
 }
                              //  It opens the  text file
            else if(s.equals("OpenTextFile"))
                {
                        fd1=new FileDialog(f,"Opening Files.....");
                     fd1.show();
```

```
                path1=fd1.getDirectory()+fd1.getFile();
                    if(path1!=null)
{
                        mi1.setEnabled(true);
                        mi2.setEnabled(false);
                        opentextfile(path1);
                        ta.setEditable(false);
                         }
                 }
    //      It saves the image file which contains the encrypted data
              else if(s.equals("Save"))
                  {
                     flag=0;
                    FileDialog fd2=new FileDialog(f,"save file",FileDialog.SAVE);
                    fd2.show();
                    try
                   {
                   if(fd2.getFile()!=null)
          {
                     filename=fd2.getDirectory()+fd2.getFile();
                      mi3.setEnabled(false);
                       FileOutputStream fout=new FileOutputStream(filename);
                   DataOutputStream d=new DataOutputStream(fout);
```

```
i=0;
                    while(i<maxp)
                {
                d.write(p1[i]);
                                    i++;
            }
                        fout.close();
                    repaint();
                }
                    }
                catch(Exception e)
                    {
                     System.out.println(e);
                      }
                }
//          It gets the data from the image and decrypt it by using IDEA //algorithm
            else if(s.equals("Decrypt"))
                    {
                     ta.setText(" ");
                     FileDialog fd3=new FileDialog(f,"OPEN file");
                    fd3.show();
                     msg="Decrypted Image";
                     try
```

```
    {
      if(fd3.getFile()!=null)
{
        filename1=fd3.getDirectory()+fd3.getFile();
       i=0;
        FileInputStream fis1 = new FileInputStream(filename1);
      DataInputStream dis=new DataInputStream(fis1);
        while((ch=dis.readUnsignedByte())!=-1)
                {
                      p2[i]=ch;
                        p3[i]=ch;
                        i++;
             }f
      }
 }

       catch(Exception e)
        {
              maxp1=i;
             switch(p2[28])
              {
                case 24:
             {
```

```
            init241();
                                break;
                            }    //case 24 ends
                        }     // switchp[28] ends
                    }   //catch() ends
          img3=createImage(new MemoryImageSource(width,height,pixels,0,width));
          img=createImage(new MemoryImageSource(width,height,pixels,0,width));
               flag=1;
                repaint();
                }
        else if(s.equals("Reset"))
          {
           img=null;
           img1=null;
          img3=null;
          mi2.setEnabled(true);
          mi1.setEnabled(false);
          mi3.setEnabled(false);
          repaint();
        }
                                //     It exits the program
else if(s.equals("Exit"))
                    {
```

```
                System.exit(0);
                 }
}
                //          It reads the encrypted data from the image
public void  init241()
 {
                width= p2[21]<<24 | p2[20]<<16 | p2[19]<<8 | p2[18];
                height= p2[25]<<24 | p2[24]<<16 | p2[23]<<8 | p2[22];
                int extra=(width*3)%4;
        if(extra!=0)
        padding=4-extra;
                  int x,z=54;
                  l=0;
                  int j=0;
                i=0;
                for(int q=0;q<height;q++)
                  {
                  x=0;
          while(x<width)
        {
         b=p2[z]&0xff;
                  binary[j++]=b&0x01;
                  g=p2[z+1]&0xff;
```

```
                binary[j++]=g&0x01;
        r=p2[z+2]&0xff;
                binary[j++]=r&0x01;
        pix[l]= 255<<24 | r<<16 | g<<8 | b;
                z=z+3;
                x++;
        l++;
                }
    z=z+padding;
 }
    int k;
    x=0;
    stringcon();
        for(i=l-width;i>=0;i=i-width)
        {
                for(k=0;k<width;k++)
                {
                pixels[x]=pix[i+k];
//              pixels1[x]=pix[i+k];
                x++;
                }
        }
}
```

// It converts the data obtained from the image into binary data

```
public void stringcon()
 {
    int i,j,k;
    int temp[]=new int[8];
    int a[]=new int[32];
    i=0;
   j=0;
   for(i=0;i<10000;i++)
   bytes[i]=0;
   i=0;
   maxbinary1=0;
   for(i=0;i<24;i++)
   a[i]=binary[i];
    for(i=0;i<24;i++)
   maxbinary1+=a[i]*Math.pow(2,23-i);
   maxbinary2=maxbinary1*8;
   ivalue=0;
       for(i=24,j=0;i<32;i++,j++)
         {
           a[j]=binary[i];
           ivalue+=a[j]*Math.pow(2,7-j);
         }
```

```
        if(ivalue==73)
        {
          i=32;
         while(i<maxbinary2)
            {
            for(k=0;k<=7;k++)
            temp[k]=binary[i++];
            for(k=0;k<=7;k++)
            bytes[j]+=temp[k]*Math.pow(2,7-k);
             j++;
            }
   String s=JOptionPane.showInputDialog(this,"Enter key with 16 letters");
   char ch[]=new char[s.length()];
   ch=s.toCharArray();
        try
            {
              FileOutputStream f=new FileOutputStream("key1.txt");
                for(i=0;i<ch.length;i++)
                f.write(ch[i]);
                f.close();
FileOutputStream fout=new FileOutputStream("enc1.txt");
DataOutputStream d=new DataOutputStream(fout);
                    i=8;
```

```
                while(i<(maxbinary1))
                    {
                    d.write(bytes[i]);
                            i++;
                }
               fout.close();
            }
catch(Exception e)
    {
       System.out.println(e);
      }

         ideaalgo b=new ideaalgo();
         b.procedure();
         b.decryption("enc1.txt");
   try
    {
      BufferedReader d;
      StringBuffer sb=new StringBuffer();
      d=new BufferedReader(new FileReader("dec.txt"));
      String line;
      while((line=d.readLine())!=null)
         sb.append(line+"\n");
```

```
            ta.setText(sb.toString());
            d.close();
      }
catch(Exception e)
 {
    System.out.println(e);
   }
 }
}
```

// It opens the text file and converts the data into cipher text

```
public void opentextfile(String path1)
       {
          FileInputStream fin,fin1;
          int i,j=0;
           String s=JOptionPane.showInputDialog(this,"Enter key with 16 letters");
           char ch[]=new char[s.length()];
           ch=s.toCharArray();
           try
             {
              FileOutputStream f=new FileOutputStream("key1.txt");
              for(i=0;i<ch.length;i++)
              f.write(ch[i]);
              f.close();
```

```
    ideaalgo a=new ideaalgo(path1);
    a.procedure();
    a.encrypt();
    BufferedReader d;
    StringBuffer sb=new StringBuffer();
    d=new BufferedReader(new FileReader(path1));
    String line;
    while((line=d.readLine())!=null)
    sb.append(line+"\n");
    ta.setText(sb.toString());
    d.close();
     fin=new FileInputStream("enc.txt");
  do
 {
         i=fin.read();
    if(i!=-1)
         {
           bytes[j++]=i;
            }
       } while(i!=-1);
max=j;
fin.close();
binarycon();
```

```
        }
      catch(Exception e)
        {
       System.out.println(e);
        }
    }
```

// It converts the encrypted data into binary data

```
public void binarycon()
          {
          int i,j,k,t;
          int temp[]=new int[10];
          int  m=0;
           for(i=0;i<600000;i++)
           binary[i]=0;
           int b[]=new int[32];
           int dum;
          dum=max;
     i=0;
    while(dum!=0)
      {
       b[i]=dum%2;
       i=i+1;
      dum/=2;
```

```
    }
j=24-i;
for(k=j,t=i-1;k<(i+j);k++,t--)
binary[k]=b[t];
dum=73;
i=0;
while(dum!=0)
 {
  b[i]=dum%2;
  i=i+1;
 dum/=2;
}
 j=32-i;
 for(k=j,t=i-1;k<32;k++,t--)
 binary[k]= b[t];
 m=32;
for( i=0 ; i < max ; i++)
     {
        j=0;
        while( bytes[i]!= 0 )
           {
               temp[j++]=bytes[i]%2;
               bytes[i]=bytes[i]/2;
```

```
            }
          for( k=0;k<8-j ; k++)
          binary[m++]=0;
          for(k=j-1;  k >=0 ; k--)
          binary[m++]=temp[k];
           }
        maxbinary=m;
       }
public void paint(Graphics g)
        {
         g.drawString ("Original Image",180,50);
        if(msg!=null)
         g.drawString (msg,550,50);
         if(img!=null)
         g.drawImage(img,20,80,this);
         if(img1!=null&&flag!=1)
         g.drawImage(img1,400,80,this);
         if(img3!=null&&flag!=0)
         g.drawImage(img3,400,80,this);
         }
           //          It opens the image file and hides the data in images
 public void openimagefile(String fn,int sno)
       {
```

```
    try
     {
      i=0;
      FileInputStream fis = new FileInputStream(fn);
      DataInputStream dis=new DataInputStream(fis);
      while((ch=dis.readUnsignedByte())!=-1)
       {
         p[i]=ch;
           p1[i]=ch;
         i++;
       }
    fis.close();
   dis.close();
}
catch(Exception e)
{
       maxp=i;
       switch(p[28])
          {
       case 24:
          {
             init24();
          break;
```

```
                }    //case 24 ends
              }     // switchp[28] ends
        }   //catch() ends
            img=createImage(new MemoryImageSource(width,height,pixels,0,width));
            img1=createImage(new MemoryImageSource(width,height,pixels1,0,width));
 }       //openfile ends
        //          It  hides the binary data in the least significant bit of the image
public void  init24()
 {
                width= p[21]<<24 | p[20]<<16 | p[19]<<8 | p[18];
                height= p[25]<<24 | p[24]<<16 | p[23]<<8 | p[22];
                int extra=(width*3)%4;
        if(extra!=0)
        padding=4-extra;
                  int x,z=54;
                  l=0;
                  int j=0;
             for(int q=0;q<height;q++)
              {
                x=0;
          while(x<width)
                    {
              b=p[z]&0xff;
```

```
                if(j<maxbinary)
                  {
                   if(binary[j]!=0)
                    {
                      p1[z]=p[z]&0xff|binary[j++];
                      b1=p1[z]&0xff;
                    }
                  else
                    {
                      p1[z]=p[z]&0xff & (binary[j++]|0xfe);
                      b1=p1[z]&0xff;
                    }
               }
          else
             b1=p[z]&0xff;
g=p[z+1]&0xff;
           if(j<maxbinary)
             {
                   if(binary[j]!=0)
                    {
                      p1[z+1]=p[z+1]&0xff|binary[j++];
                      g1=p[z+1]&0xff;
                    }
```

```
                else
                 {
                    p1[z+1]=p[z+1]&0xff & (binary[j++]|0xfe);
                     g1=p1[z+1]&0xff;
                 }
            }
          else
              g1=p[z]&0xff;
r=p[z+2]&0xff;
          if(j<maxbinary)
            {
                if(binary[j]!=0)
                {
                  p1[z+2]=p[z+2]&0xfe|binary[j++];
                  r1=p[z+2]&0xff;
                 }
                else
                  {
                    p1[z+2]=p[z+2]&0xff & (binary[j++]|0xfe);
                    r1=p1[z+2]&0xff;
                    }
            }
          else
```

```
                    r1=p[z]&0xff;
        z=z+3;
        pix[l]= 255<<24 | r<<16 | g<<8 | b;
         pix1[l]= 255<<24 | r1<<16 | g1<<8 | b1;
        l++;
        x++;
        }
z=z+padding;
}
int k;
x=0;
        for(i=l-width;i>=0;i=i-width)      //l=WIDTH * height
        {
                for(k=0;k<width;k++)
                {
                pixels[x]=pix[i+k];
                  pixels1[x]=pix[i+k];
                x++;
                }
        }
    }
}
```

Algorithm implementation

```
class ideaalgo
   {
      FileInputStream fin,fkey,fenc1;
      DataOutputStream fdec,fenc;
 int
step1,step2,step3,step4,step5,step6,step7,step8,step9,step10,step11,step12,step13,ste
p14;
      int t;
      int index=0,j1,i1,i,mark,k1,k2,k;
      int iz[]=new int[52];
      int size;
      byte buf[],keybuf[];
      int ft,ft1,nl,np=0,p;
      int  x[]=new int[5];
      int  z[]=new int[52];
      byte   buf1[];
      String file2,file3,file4;
      ideaalgo()
          {
                file2="key1.txt";
           }
```

```
    ideaalgo(String file1)
     {
                file2="key1.txt";
                file3="enc.txt";
                file4="dec.txt";
     try
          {
         fin=new FileInputStream(file1);
         fenc1=new FileInputStream(file3);
         fenc=new DataOutputStream(new FileOutputStream(file3));
         }
        catch(Exception e)
         {
          System.out.println(e);
         }
     }
```

// It generates 52 keys which is used to encrypt the data

```
void  procedure()
     {
    try
     {
      fkey=new FileInputStream(file2);
     }
```

```
    catch(Exception e)
     {
        System.out.println(e);
     }
keybuf= new byte[16];
     try
      {
       fkey.read(keybuf);
       z= new int[52];
       j1=0;
       i1=0;
for(i=0;i<52;i++)
       z[i]=0;
   while( i1<8)
       {
          if((j1+1)%2==0)
           {
           z[i1]|=keybuf[j1];        // dividing 64-bit cypher block into four 16-bit registers
          i1++;
            }
          else
          {
               z[i1]=keybuf[j1];
```

```
                z[i1]<<=8;
              }
          j1++;
          }
   i=0;
          for(j1=1;j1<=5;j1++)
            {
               i++;
                z[i+7]=((z[i]<<9)&0xfe00)|((z[i+1]>>7)&0x1ff);
                z[i+8]=((z[i+1]<<9)&0xfe00)|((z[i+2]>>7)&0x1ff);
                z[i+9]=((z[i+2]<<9)&0xfe00)|((z[i+3]>>7)&0x1ff);
                z[i+10]=((z[i+3]<<9)&0xfe00)|((z[i+4]>>7)&0x1ff);
                z[i+11]=((z[i+4]<<9)&0xfe00)|((z[i+5]>>7)&0x1ff);
                z[i+12]=((z[i+5]<<9)&0xfe00)|((z[i+6]>>7)&0x1ff);
                 z[i+13]=((z[i+6]<<9)&0xfe00)|((z[i-1]>>7)&0x1ff);
                z[i+14]=((z[i-1]<<9)&0xfe00)|((z[i]>>7)&0x1ff);
            i=i+7;
          }
i1=41;
          z[48]=((z[i1]<<9)&0xfe00)|((z[i1+1]>>7)&0x1ff);
          z[49]=((z[i1+1]<<9)&0xfe00)|((z[i1+2]>>7)&0x1ff);
          z[50]=((z[i1+2]<<9)&0xfe00)|((z[i1+3]>>7)&0x1ff);
          z[51]=((z[i1+3]<<9)&0xfe00)|((z[i1+4]>>7)&0x1ff);
```

```
}
catch(Exception e)
{
System.out.println(e);
}
}
//              This function encrypts the data using 52 keys and place the //o/p in a file
void encrypt()
     {
     try
      {
      size=fin.available();
      p=size%8;
      if(p!=0)
         np=8-p;
         size+=np;
         if(p==0)
         nl=size/8;
         else
         nl=size/8+1;
         buf=new byte[8];
         buf1=new byte[size+10];
         fin.read(buf1);
```

```
int enc[]=new int[size];
   mark=-8;
   k2=0;
   for(k=0;k<nl;k++)
    {
    mark+=8;
    for(int k1=0;k1<8;k1++)
        buf[k1]=buf1[mark+k1];
        i=0;
      for(i=0;i<4;i++)
             x[i]=0;
          j1=0;i1=0;
         while( i1<=3)
          {
                if((j1+1)%2==0)
                  {
              x[i1]|=buf[j1];        // dividing 64-bit cipher block into four 16-bit registers
              i1++;
               }
              else
               {
               x[i1]=buf[j1];
               x[i1]<<=8;
```

```
            }
          j1++;
       }
// 7 rounds and 14 steps
     for(i=0 ; i <48 ; )
         {
              step1=mul(x[0],z[i+0]);
              step2=(x[1]+z[i+1])%65536;
              step3=(x[2]+z[i+2])%65536;
              step4=mul(x[3],z[i+3]);
              step5=step1^step3;
              step6=step2^step4;
              step7=mul(step5,z[i+4]);
              step8=(step6+step7)%65536;
              step9=mul(step8,z[i+5]);
              step10=(step7+step9)%65536;
              step11=step1^step9;
              step12=step3^step9;
              step13=step2^step10;
              step14=step4^step10;
       x[0]=step11;
       x[1]=step13;
       x[2]=step12;
```

```
                x[3]=step14;
                i=i+6;
            }
   x[0]=mul(x[0],z[48]);
   x[1]=(x[1]+z[49])%65536;
   x[2]=(x[2]+z[50])%65536;
   x[3]=mul(x[3],z[51]);
   for(int t=0;t<4;t++)
   {
    ft1 =x[t]&255;
    ft=x[t]>>8;
    fenc.write((char)ft);
    fenc.write((char)ft1);
  }
}
fin.close();
    }
 catch(Exception e)
   {
   System.out.println(e);
  }
}
```

// This function decrypts the data which in text file by using 52 keys

```
void decryption(String file)
 {
      try
         {
            fdec=new DataOutputStream(new FileOutputStream("dec.txt"));
            fenc1=new FileInputStream(file);
         }
        catch(Exception e)
        {
           System.out.println(e);
         }
      try
      {
        size=fenc1.available();
        np=0;
        p=size%8;
        if(p!=0)
        np=8-p;
        size+=np;
        if(p==0)
        nl=size/8;
        else
         nl=size/8+1;
```

```
buf1=new byte[size+10];
buf=new byte[8];
fenc1.read(buf1);
mark=-8;
int arr[]=new int [8];
for(k1=0;k1<nl;k1++)
 {
      mark+=8;
      for(int k2=0;k2<8;k2++)
       buf[k2]=buf1[mark+k2];
      for(int k2=0;k2<8;k2++)
       {
          arr[k2]=0;
          if(buf[k2]>=0)
          arr[k2]=buf[k2];
          else
          arr[k2]=buf[k2]+256;
         }
         j1=0;i1=0;
          while( i1<=3)
              {
                 if((j1+1)%2==0)
                      {
```

```
            x[i1]|=arr[j1];        // dividing 64-bit cypher block into four 16-bit registers
          i1++;
              }
           else
              {
              x[i1]=arr[j1];
               x[i1]<<=8;
              }
                j1++;
           }
for(int t=0;t<4;t++)
       {
       ft1 =x[t]&255;
       ft=x[t]>>8;
      }
            step1=mul( x[0] , minverse( z[48] ));
            step2=( x[1] + ainverse( z[49] )) % 65536;
            step3=( x[2] + ainverse( z[50] )) % 65536;
            step4=mul( x[3] , minverse( z[51] ));
            step5=step1^step3;
            step6=step2^step4;
            step7=mul(step5,z[46]);
            step8=(step6+step7)%65536;
```

```
            step9=mul(step8,z[47]);
            step10=(step7+step9)%65536;
            step11=step1^step9;
            step12=step3^step9;
            step13=step2^step10;
            step14=step4^step10;
  x[0]=step11;
  x[1]=step12;
  x[2]=step13;
  x[3]=step14;
```

// 2nd round

```
int j2=40;
      for(j1=1;j1<=7;j1++)
        {
            step1=mul( x[0] , minverse( z[j2+2] ));
            step2=( x[1] + ainverse( z[j2+4] )) % 65536;
            step3=( x[2] + ainverse( z[j2+3] )) % 65536;
            t=step2;
            step2=step3;
            step3=t;
            step4=mul( x[3] , minverse( z[j2+5] ));
            step5=step1^step3;
            step6=step2^step4;
```

```
                step7=mul(step5,z[j2+0]);
                step8=(step6+step7)%65536;
                step9=mul(step8,z[j2+1]);
                step10=(step7+step9)%65536;
                step11=step1^step9;
                step12=step3^step9;
                step13=step2^step10;
                step14=step4^step10;
          x[0]=step11;
          x[1]=step12;
          x[2]=step13;
          x[3]=step14;
        j2=j2-6;
    }
x[0]=mul(x[0],minverse(z[0]));
x[1]=(x[1]+ainverse(z[2]))%65536;
x[2]=(x[2]+ainverse(z[1]))%65536;
x[3]=mul(x[3],minverse(z[3]));
t=x[1];
x[1]=x[2];
x[2]=t;
          for(int t=0;t<4;t++)
            {
```

```
            ft1 =x[t]&255;
            ft=x[t]>>8;
            fdec.write((char)ft);
            fdec.write((char)ft1);
          }
    }

  }
    catch(Exception e)
  {
  System.out.println(e);
    }
}
```

// It multiplies two numbers and returns (a*b)%65537

```
int mul( int a  , int b)
    {
     double c,d;
        if (a==0)
          c=65536;
        if(b==0)
         d=65536;
       c=(double)a;
       d=(double)b;
```

```
        a= (int)((c*d)%65537);
         return a;
    }
              //          It returns  the multiplicative inverse modulo 65537 of z
int minverse(int z)
     {
            int to,t1;
            int q,y;
            if(z<=1)
            return z;
            t1=0x10001/z;
            y=0x10001%z;
            if(y==1)
            return (0xffff&(1-t1));
            to=1;
           do
            {
              q=z/y;
              z=z%y;
              to+=q*t1;
              if(z==1)
              return to;
              q=y/z;
```

```
            y=y%z;
            t1+=q*to;
          }while(y!=1);
       return (0xffff&(1-t1));
}
```

// It returns the additive inverse of z

```
 int ainverse(int z)
            {
               return (65536-z);
             }
}
```