Table of Contents

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

Data security is paramount concern for all the net users irrespective of the network. The present day hackers are a threat to the data and the threat hangs like a Damocles sword. The transmission of data through any channel of communication needs strong encryption techniques for the purpose of data security. The recent trends and development in information technology highlights the need for safe, secure and protected transmission of data. The conventional encryption methods failed to give the desired result of protecting the data. Simple way is to come up with unique id and passwords, and a combination of alphabets & numerical. AES has emerged as a frontrunner and efficient algorithm because of inherent inbuilt in advantage of better security with less implementation complexity (Cacciaguerra, 2003)

After extensive research in image coding, for image compression application, LSB works as a standard tool, for their data reduction capability. The complete image is compressed and transformed into a single data object by wavelet compression system, rather than block by block as in a DCT-based compression system (Altigani and Barry, 2013)

When the entire image is achieved there will be a uniform distribution of compression error across that image. An image resolution enhancement in the wavelet domain is a subject of interest for further research and recently many new algorithms have been proposed. Of

these the Least Significant Bit (LSB) is the most-suited application. LSB decomposes an image into different sub-band images. Which can be named as low-low (LL), low-high (LH), high-low (HL), and high-high (HH).Here the sub-bands have the same size as the input image. Xuan et al. method is based on the integer wavelet transform to improve the embedding capacity.(Chan and Cheng, 2004)

Advanced Encryption Standard (AES) is a symmetric encryption algorithm in which we can use only one key for both encryption and decryption that can be used by sender and receiver. In AES we can use 128,192 or 259 bits long with each of them contains 2128, 2192 and 2256 combinations. The secrecy maintained by the key is secured and authentication is maintained the key itself. In this both the keys must be kept secret. But without knowing private key or at least other information impossible to decode the cipher text. With the help of public key and algorithm it must be insufficient to find the private key. We need secrecy and authentication, only one key is enough that is private key for encryption. (Cacciaguerra, 2003)

In cryptographic solutions DES and AES will provide the security but from cryptography point of view they differ one is symmetric and another one is asymmetric. AES key is harder to break than DES, and both need more dealing out to distribute keys between sender and receiver. The AES algorithm formulated in the figure 1 below.

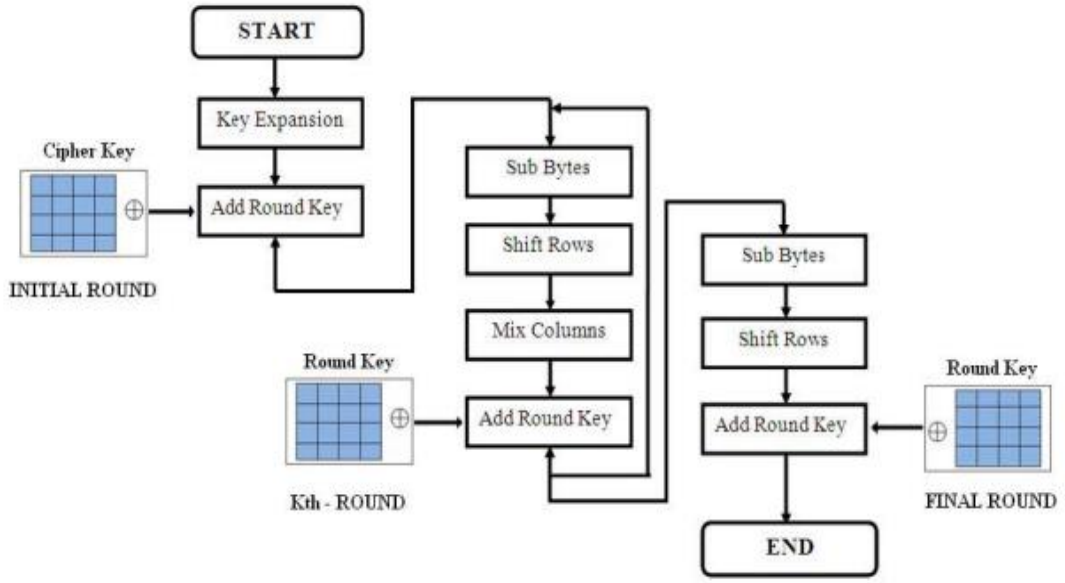

Figure 1: displaying the AES algorithm

In recent years, the rapid growth in the demand of transmitting images via public networks has raised a lot of interest on image compression and encryption. The need to apply both compression and encryption to digital images keeps rising. Hence, image security/protection from unauthorized access becomes very important. Image compression consists of processes leading to compact representation of an image, so as to reduce total storage/transmission requirements. While image encryption refers to converting an image to such a format, so that it becomes unreadable to unauthorized access and can be transmitted securely over the internet. On the other hand, image decryption means to convert the unreadable format of an image to an original image.(Kant, Nath, and Chaudhary, 2008)

As Information Communication Technology (ICT) has being on top gear of improvement so also hackers and attackers too have being on their toes to get a way of retrieving information from organizations and agencies' database or data-bank. Used in terror-related crimes to gain access to resources and locations and information, information is key and very important when it comes to information technology (IT), Patric, Ebele & Chinedu, (2012) attackers and hackers are on the lookout to fetch information and useful tools to enhance their detrimental work or job to defraud individuals, corporate organization, military settings, government parastatals, etc. most of these atrocities happens via the internet mostly.

The internet as a whole does not use secure links, thus information in transit may be vulnerable to interception as well. The importance of reducing a chance of the information being detected during the transmission is being an issue nowadays. Some solutions to be discussed is how to pass information in a manner that the existence of the message is unknown in order to repel attention of potential attackers. Besides hiding data for confidentiality, this approach of information hiding can be extended to copyright protection for digital media. Patric, Ebele & Chinedu, (2012)

Least Significant Bit (LSB)

Over the past few years, numerous steganography techniques that embed hidden messages in multimedia objects have been proposed. There have been many techniques for hiding information or messages in images in such a manner that alteration made to the image is

perceptually indiscernible. Commonly approaches are include LSB, Masking and filtering and Transform techniques. (Juneja and Sandhu, 2012)

Least significant bit (LSB) insertion is a simple approach to embedding information in image file. The simplest steganography techniques embed the bits of the message directly into least significant bit plane of the cover-image in a deterministic sequence. Modulating the least significant bit does not result in human perceptible difference because the amplitude of the change is small. In this technique, the embedding capacity can be increased by using two or more least significant bits. At the same time, not only the risk of making the embedded message statistically detectable increase but also the image fidelity degrades. Hence a variable size LSB embedding schema is presented, in which the number of LSBs used for message embedding/extracting depends on the local characteristics of the pixel. The advantage of LSB-based method is easy to implement and high message pay-load. (Juneja and Sandhu, 2012)

Although LSB hides the message in such way that the humans do not perceive it, it is still possible for the opponent to retrieve the message due to the simplicity of the technique. Therefore, malicious people can easily try to extract the message from the beginning of the image if they are suspicious that there exists secret information that was embedded in the image.(Juneja and Sandhu, 2012)

Therefore, a system named Secure Information Hiding System (SIHS) is proposed to improve the LSB scheme. It overcomes the sequence-mapping problem by embedding the massage into a set of random pixels, which are scattered on the cover-image.

Masking and filtering techniques, usually restricted to 24 bits and gray scale image, hide information by marking an image, in a manner similar to paper watermarks. The technique perform analysis of the image, thus embed the information in significant areas so that the hidden message is more integral to cover image than just hiding it in the noise level.

Transform techniques embed the message by modulating coefficient in a transform domain, such as the Discrete Fourier Transform, or Wavelet Transform. These methods hide messages in significant areas of the cover image, which make them more robust to attack. Transformations can be applied over the entire image, to block throughout the image, or other variant.

1.2 Problem of Statement

This project work has been able to identify that when sensitive data that falls into the hands of the unwanted users can be very dangerous to the affected party which can lead to extortion of funds, phishing and also cyber or data impersonation.

The problem that is identified from this research is that of the cyber/data impersonation, phishing, and illegal extortion of funds from the affected victims which keeps the affected victims vulnerable.

Based on research done, more people join the cyberspace revolution on daily basis and this serves as threat to information on transit and storage as attackers and hackers are always on their toes to fetching information from the cyberspace so as to use for their own benefits and causing large detriments to the entire populace that will be affected Aisha & Wilson, (2013).

IT ubiquitous has made fraud easier as transactions can be done just anywhere, any day and at any time. This project therefore proposes a method that helps encrypt data and then hide the data from the normal eyes this called cryptography and steganography respectively, with the use of hybridization of algorithms i.e Advanced Encryption Standard (AES) for Cryptography and Least Significant Bit (LSB) for steganography. It follows that if this system is being implemented the fear of being ripped off of data or information being leaked out of organization or parastatals will reduce to the barest minimum if not eradicated completely.

The internet as a whole does not use secure links, thus information in transit experiences lots of processes and are vulnerable to interception as well (The white house Washington, 2011).

1.3 Aim and Objectives

The aim of this research is to propose, build and implement a system that will ensure the integrity and confidentiality of data.

In order to achieve this aim, the following objectives are formulated:

1. To encrypt data using the AES algorithm.

2. To hide the existence of data in image using the LSB steganography.
3. To integrate and evaluate the Least significant bit (LSB) and the AES encryption algorithm.

1.4 Significance of Study

In the age of explosive worldwide growth of electronic data storage and communications, many vital national and individual interests require the effective protection of information. The aforementioned has called for several researches in order to safeguard information from unauthorized personnel. This research work will look at how the information can be secured better by combining cryptography and steganography respectively, cryptography which is encrypting data i.e. making information meaningless to the viewer before it is converted back to its original form steganography which is hiding of information termed steganography i.e. the process of keeping data away from the normal view of the eyes.

1.5 Scope of Study

The scope of this project work will be targeted around trying to combine both cryptography and steganography for the process of designing a system that will first help to encrypt information and then later hide information from the normal eyes and keep it away from attackers and still retain the normal quality of the initial information and avoiding suspicion to the existence of a hidden message most importantly in applications like digital audio, video and pictures.

The main purpose of this process is to protect our information and documents from attacker and waste of resources. This project therefore seeks that an effective preventive measure

be put in place to avert the case of unforeseen circumstances by attackers or hackers as the case maybe. It therefore follows that it prevents access from unauthorized copyright owner except given right to otherwise access it.

1.6 Project Layout

This dissertation is divided into five (5) chapters. These are summarized below:

1. Chapter one presents the introduction to the study, statement of the problem, the aim, objectives, significance of study, scope of the study, project layout and definition of terms.
2. Chapter two discussed the literature review based on the related concept and the related works.
3. Chapter Three discussed methodology of the project.
4. Chapter Four presents the implementation showing graphical images and also discussion of those images.
5. Chapter Five presents the summary, conclusion and the recommendation.

1.7 Definition of Terms

AES: it is an acronym for advanced Encryption Standard and it is an algorithm used for Encryption and Decryption and it replaces the Data Encryption Standard (DES).

LSB: it is an acronym for Least Significant Bit (LSB) which is a mathematical tool for hierarchically decomposing an image.

Algorithm: it is a step by step procedure for solving or accomplishes a task.

Asymmetric key: is form of encryption where the key used to encrypt is different from the one used to decrypt. It is also referred to as Public key.

Bit: a bit is a single numeric value, either "1" or "0", that encodes a single unit of digital information.

Byte: a byte is a sequence of bits; usually eight bits equal one byte.

Cipher: a cipher is any method of encrypting text (concealing its readability and meaning). It is also sometimes used to refer to the encrypted text message itself.

Computer Security: The protection afforded to an automated information system in order to attain the applicable objective of preserving the integrity, availability and confidentiality of information system resources.

Computer: Computer is a machine or device which under the control of a stored program, accept data from input device, perform arithmetic & logical operations in accordance with predefined instructions (program) and finally transfers the processed data to an output device.

Cryptography: it is a branch of engineering that deals with active, intelligent and malevolent opposition. It is also chiefly concerned with linguistic pattern.

Decryption: it is the reverse process of encryption. It involves the process of transforming cipher text into plain text.

Encryption: it is the process of transformation (referred to as plaintext) using an algorithm (called cipher) to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key.

Key: a key in cryptography is a piece of instruction (a parameter) that determines the functional output of a cryptographic or cipher. Without a key, the algorithm will have no result. A key also specify the particular transformation of plaintext into cipher text of vice versa.

Plaintext: The information in its original form. This is also known as cleartext.

Private Key: it is a key that is kept secret. The private key cannot be practically derived from the public key.

Public Key: a public key in cryptography is also known as asymmetric cryptography is a form of cryptography in which the key used to encrypt a message differs from the key used to decrypt it. The public key may also be widely distributed.

Security: The mechanism by which information and services are protected from unintended or unauthorized access, change or destruction.

Image Encryption: While image encryption refers to converting an image to such a format, so that it becomes unreadable to unauthorized access and can be transmitted securely over the internet.

Image Decryption: image decryption means to convert the unreadable format of an image to an original image.

Steganography: The practice of hiding messages, so that the presence of the message itself is hidden, often by writing them in places where they may not be found.

Temporal Domain: Temporal characterization occurs when you have a series of images taken at different time. Correlations between the images are often used to monitor the dynamic changes of the object.

Spatial Domain: Spatial characterization applies when you are analyzing one image.

Symmetric key: A form of encryption where a single key is used to encrypt and decrypt. It is also referred to as secret key.

CHAPTER TWO

REVIEW OF RELATED LITERATURE

2.0 Introduction

In this chapter the researcher discussed about the reviews of related literature which is related to the Data security using the Advance Encryption Standard (AES) and the Least Significant Point (LSB), in which we also be looking at the related concepts to this project work and also the related works that people have done so as to tailor this design to fill in the gap/loophole that is found in this related works.

2.1 Related Concepts

2.2 Advance Encryption Standard (AES)

The AES algorithm is a symmetric block cipher that processes data blocks of 128-bits using a cipher key of length 128,192 or 256 bits each data block consist of a (4x4) array of bytes called the state, on which the basic operations of the AES algorithm are performed. The AES encryption procedure is shown in Figure 1. For full encryption, the data is passed through Nr rounds (Nr = 10, 12, 14) Altigani & Barry (2013) These rounds are governed by the following transformations:

- SubBytes transformation: is a nonlinear byte substitution that operates independently on each byte of the state using a substitution table (the SBox).
- ShiftRows transformation: is a circular shifting operation on the rows of the state with different numbers of bytes (offsets).

- MixColumns transformation: is equivalent to a matrix multiplication of columns of the states. It should be noted that the bytes are treated as polynomials rather than numbers.
- AddRoundKey transformation: is an XOR operation that adds a round key to the state in each iteration, where the round keys are generated during the key expansion.

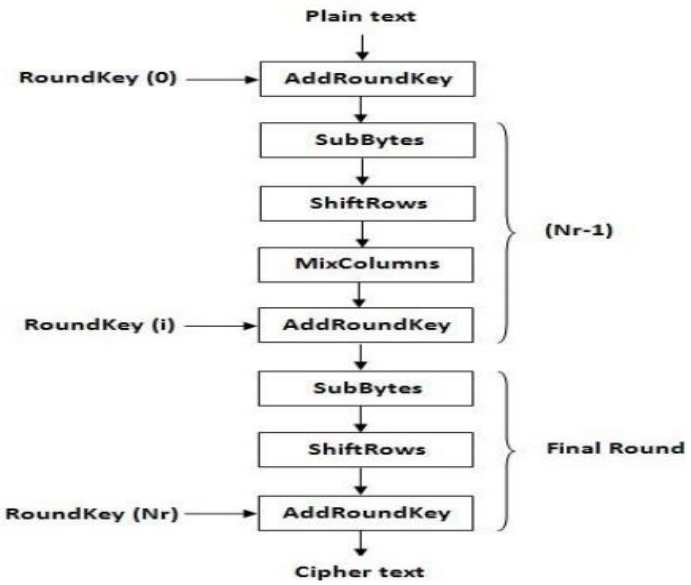

Figure 2: AES Algorithm – Encryption Structure

Here author Aidi Zhang, Nanrun Zhou (2010) proposed a new color image encryption algorithm joining compressive sensing with Arnold transform is suggested which can encrypt the color image into a gray image. Color images are more attractive in vision and take more information than gray images thus color image encryption has turn out to be a significant issue for information security. They explore a color image encryption algorithm

combining CS with Arnold transform, where the three color components are encrypted and compressed to be a gray image, which can confuse the attacker to enhance the security. Making an allowance for the dimensional decline and random projection of compressive sensing, they use compressive sensing to encrypt and condense the three color components of color image at the same time. The three encrypted and reduced color components' dimensions are less significant than the original image thus they can be grouped into a gray image, and then the gray image is mix up by Arnold transform to improve the security level. The experimental results shows that proposed algorithm can also be applied in the multiple-image encryption gives the legitimacy and the consistency real time application

2.3 Least Significant Bit (LSB)

At the beginning of 2000, the International Organization for Standardization (ISO) and the International Electro technical Commission (IEC) made the core of JPEG2000 which adopts Least Significant Bit (LSB) as the standard compression tool. Least significant bit (LSB) insertion is a simple approach to embedding information in image file. The simplest steganography techniques embed the bits of the message directly into least significant bit plane of the cover-image in a deterministic sequence. Modulating the least significant bit does not result in human perceptible difference because the amplitude of the change is small. In this technique, the embedding capacity can be increased by using two or more least significant bits. At the same time, not only the risk of making the embedded message statistically detectable increase but also the image fidelity degrades. Hence a variable size LSB embedding schema is presented, in which the number of LSBs used for message

embedding/extracting depends on the local characteristics of the pixel. The advantage of LSB-based method is easy to implement and high message pay-load.

Although LSB hides the message in such way that the humans do not perceive it, it is still possible for the opponent to retrieve the message due to the simplicity of the technique. Therefore, malicious people can easily try to extract the message from the beginning of the image if they are suspicious that there exists secret information that was embedded in the image.

Therefore, a system named Secure Information Hiding System (SIHS) is proposed to improve the LSB scheme. It overcomes the sequence-mapping problem by embedding the massage into a set of random pixels, which are scattered on the cover-image.

Masking and filtering techniques, usually restricted to 24 bits and gray scale image, hide information by marking an image, in a manner similar to paper watermarks. The technique perform analysis of the image, thus embed the information in significant areas so that the hidden message is more integral to cover image than just hiding it in the noise level.

Transform techniques embed the message by modulating coefficient in a transform domain, such as the Discrete Fourier Transform, or Wavelet Transform. These methods hide messages in significant areas of the cover image, which make them more robust to attack. Transformations can be applied over the entire image, to block throughout the image, or other variant. Another useful feature of a wavelet transform is its symmetric

nature meaning that both the forward and the inverse transforms have the same complexity, allowing building fast compression and decompression routines. Wavelet transform divides the information of an image into an approximation (i.e. LL) and detail sub-band.

2.4 Image Compression

Least Significant Bit (LSB) is the first phase in the proposed Technique, to produces four sub-bands. The top-left corner is called LL, represents low-frequency coefficients, and the top-right called HL consists of residual vertical frequencies. The bottom-left corner LH, and bottom-right corner HH are represents residual horizontal and residual vertical frequencies respectively. The high-frequency components (LH1, HL1 and HH1) are zero or insignificant. This reflects the fact that much of the important information is contained in the LL sub-band. For this reason all the high frequency domains are discarded in this research (i.e. set all values to zero). In particular, the Daubechies wavelet transform has the ability to reconstruct approximately the original image. This property allows higher compression ratios; this is because high frequencies from the first level can be ignored without loss of accuracy. Figure 3 shows the compression image by Daubechies wavelet transform.

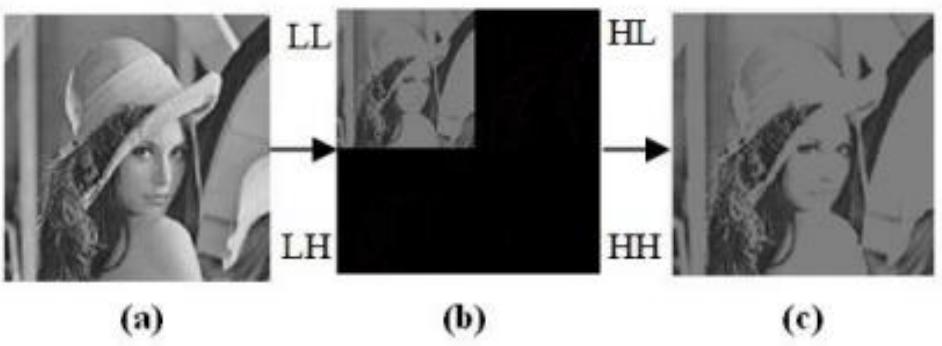

Figure 3: Compression process (a) Original Image (b) Single stage DWT (c) Compression Image.

2.5 Encryption Image

The stage of encryption is composed of two processes for encrypting the compressed image which is produced by the previous stage. The compressed image would be passed to the 128-AES algorithm as it is shown in the Figure 4. Then, the image encrypted by means of Arnold transform which the image can be permuted for more security.

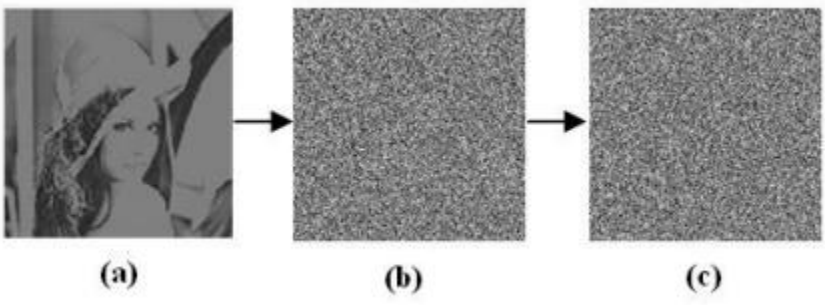

Figure 4: Encryption process: (a) compressed image (b) Encrypted image by AES (Permuted image by Arnold transform (Cipher Image).

2.6 Reconstruction Image

The image can be reconstructed by using reverse proposed technique. The cipher image is passed to the inverse Arnold transform. Afterwards, the produced image passed to the inverse 128-AES algorithm which is called decryption process. Then, the process of decompression is performed by inverse discrete wavelet transform (IDWT) by means of which the image reconstructed. Figure 5 illustrates the process of the reconstructed image.

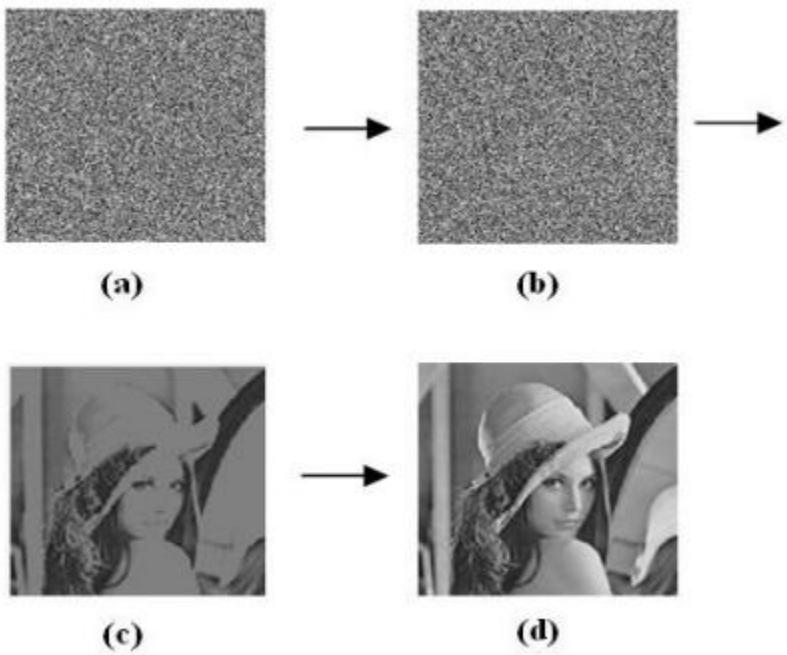

Figure 5: Reconstruction process: (a) Cipher image (b) Decrypted image by Arnold transform (c) Decrypted image by AES (d) Decompressed image by IDWT (Reconstructed Image)

2.7 Cryptography

The field of cryptography has a rich and important history, ranging from pen and paper methods, to specially built machines, to the mathematical functions that are used today. This dissertation only briefly discusses essential information regarding cryptography.

2.7.1 Definition of Cryptography

As defined by Gollmann (1999) cryptography is the science of secret writing through the enciphering and deciphering of encoded messages (Moerland 2003). It deals with the scenario where two parties, A and B, communicate over an insecure channel, with an eavesdropper possibly being able to intercept their communication as illustrated in figure 6 below.

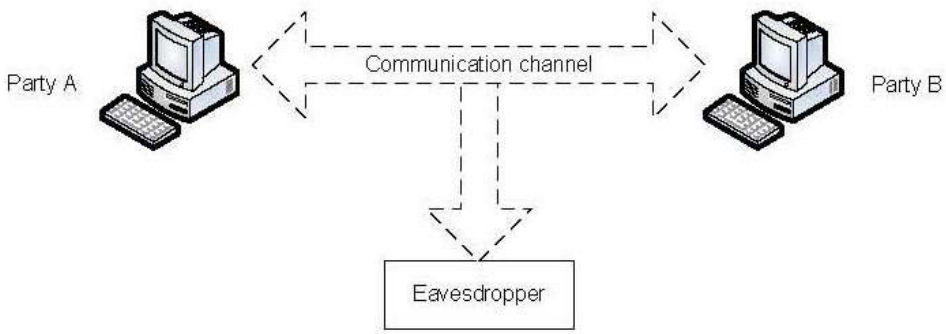

Figure 6: Communication over an unsecured channel

Image source: (Tayana, 2012)

Gollmann (1999) states that the term cryptography generally refers to a collection of cryptographic mechanisms that include:

- Encryption and decryption algorithms

- Integrity check functions
- Digital signature schemes

Encryption algorithms focus on the privacy of a secret message by scrambling the data to make it illegible to an unauthorized party. Decryption algorithms, on the other hand, unscramble the encrypted message again so that an authorized party can read it. One example of an integrity check function is a cryptographic hash function (Whitman & Mattord 2003) a mathematical function that calculates small pieces of information that can uniquely identify larger digital objects. Different objects result in different hash values. Therefore it is computationally infeasible to create another object that will have the same hash value as an existing object (Schneier 1963). These hash functions are used to verify that a message was not changed in transit. Another example of integrity check functions are message authentication codes (MACs) (Gollmann 1999). A MAC is computed from two inputs: the message and a secret cryptographic key, and also checks that information has not been tampered with. Digital signature schemes are a mechanism for detecting whether a message was altered by an eavesdropper on the communication channel by using the same principles as asymmetric encryption.

As cryptography only transforms data into gibberish form and transmits it, it tends to attract the intruders attention. On the other hand steganography made the task of the intruder even more difficult by hiding the very existence of the secret data. However, steganography could be a failure even if the presence of secret data is detected. Thus to enhance the security of communications over overt channels, it is proposed to first encrypt the secret

data using a suitable encryption algorithm, compress it and then embed it into a cover using an appropriate steganographic method Fernandes & Jeberson (2010). The wavelet decomposition method is widely based on two types of filters, i.e low pass filter and high pass filter.

Steganography is different from classical encryption, which seeks to conceal the content of secret messages; steganography is about hiding the very existence of the secret messages while cryptography is encrypting data, pictures and or information of any kind Aisha & Wilson, (2013). Steganographic protocols have a long and intriguing history that goes back to antiquity. There are stories of secret messages written in invisible ink or hidden in love letters (the first character of each sentence can be used to spell a secret, for instance).

It is confirmed that information in transit experiences lots of external attack and or, hacking thereby causing detriment to both the sender and receiver which might causes chaos to organization or the community at large. The world presently is guiding against the activities of hackers because of the detriments they cause to life and properties. This dissertation actually looked into a number of methods being used by various researchers out of which are treated below.

In Xuan. (2001), Proposed a novel distortion less image data hiding algorithm based on integer wavelet transform that can invert the stego-image into the original image without any distortion after the hidden data are extracted. This algorithm hide data into one (or more) middle bit-plane(s) of the integer wavelet transform coefficients in the middle and high frequency sub-bands (LH, HL, HH). The proposed invertible data embedding

technique is able to embed about 15k-94k bits into a grayscale image of 512 × 512 × 8 imperceptibly. In Imran, (2006). Presented a novel method for data hiding based on neighborhood pixels information to calculate the number of bits that can be used for substitution and modified Least Significant Bits (LSB) technique for data embedding. The modified solution is independent of the nature of the data to be hidden and gives correct results along with un-noticeable image degradation. The technique find the number of bits that can be used for data hiding, using the green component of the image as it is less sensitive to human eye and thus it is totally impossible for human eye to predict whether the image is encrypted or not. The application further encrypts the data using a custom designed algorithm before embedding bits into image for further security. In Singh, (2007). Presented a novel least significant bit embedding algorithm for hiding encrypted messages in nonadjacent and random pixel locations in edges of images. It first encrypts the secret message using the simplified data encryption standard (S-DES), and then detects edges in the cover image. Message bits are then, embedded in the least significant bits and random locations of the edge pixels. It ensured that the eavesdroppers will not have any suspicion that the message bits are hidden in the image and the standard steganography detection methods can not estimate the length of the secret message correctly. In Navas, (2007). Proposed a novel approach to blind reversible data hiding based on integer wavelet transform. The algorithm organizes wavelet coefficients to generate wavelet blocks, and applies a novel method to classify these wavelet blocks based on Human Visual System (HVS). The Electronic Patient Report (EPR) data are inserted based on the result of

classification. The portions of an image which contains the significant information for diagnosis are called Region of Interest (ROI) and must be stored without distortion. This concept is implemented in the newly proposed method. It is desirable to embed data outside ROI to give better protection. Encryption of EPR is done to provide additional security. The proposed scheme also has large capacity, which is important for EPR data hiding and has higher value of PSNR. In (Guo, 2008) presented a reversible watermarking technique that aims at medical record protection and biometric recognition systems. The goal was to design a system that can better store sensitive information and to protect privacy. The proposed technique used the integer wavelet transform to successfully create embedding space in the high pass frequency sub-bands. The advantages of the proposed algorithm are the simplicity and robustness against common image processing operations such as compression, filtering, and additive noise. The previous methods were depended on encrypting data with stream cipher or any other symmetric cryptographic algorithm like (S-DES) then embedding the encrypted data directly in spatial domain or embedding the encrypted data in (LL) or (LH, HL) or (LL, LH, HL) sub-bands of wavelet transform domain, or encrypting data with asymmetric cryptographic algorithm but embedding the encrypted data in spatial domain directly or embedding data in wavelet transform domain directly without encrypting data before embedding so the previous methods did not combine many cases to form integrated algorithm while the proposed algorithm in this study comprise many cases with coordinated way to get desired results of improving the capacity with keeping the security degree as high as possible.

Steganography and cryptography represent two methods for ensuring security that have been used for several years now. Like everything else in the information technology area, the two are in continuous research and improvement. Combining these methods within the same system is a relatively new direction but we can find several outstanding works in literature. One such work is presented in paper (Mamta & Parvinder, 2013). The authors propose a system that will improve the least significant bit (LSB) method, which is probably the most popular steganographical method. The described system has a private key transmitted between the sender and the receiver and used to extract the hidden message. In the area of combining steganography with cryptography is the system described in (Chander, Rajender & Sheetal). This time, the key necessary to decrypt the message will also be embedded in the image file. The authors of paper Dipti & Neha, (2010) propose a security system with three modules: one module for cryptography, another one for steganography and finally a third module for the security of the system as a whole. The encryption algorithm used is AES. In short, the idea is to first encrypt the data and to later hide the encrypted form in a steganographic file. The same technique, but with different algorithms can be found in papers (Arun, 2013; Vipula & Suresh, 2013).

In paper Tayana, (2012) the method used for securing data depends on the type of the communication. Thus, in the case of self-communication storing digital data, using only steganography is considered to be sufficient. For one-to-one communication or the exchange of information between two parties both cryptography and steganography are used. The paper (Abdelraham & Bazara, 2013) uses linguistic steganography, more

specifically the word shift coding protocol, which is combined for better security with the well-known encryption algorithm AES. The idea behind the paper is simply providing an additional layer of protection to a communication network.

However with the above methods that have been looked into by the researcher we will notice that there has not being any researcher that has used the proposed method that the researcher intends to use in solving the problem. The researcher intends to combine two methods in solving the problem of attackers bumping information in transit that combining cryptography and steganography in other words cryptography helps to encrypt the information concerned where there will be a secrecy that will be used to decrypt it when the need arises and the steganography on the other hand helps to hide the information concerned that is making double security on the information on transit that is after encrypting the information then the information is then hidden to the normal eyes. Even if the information's presence is discovered by the attacker he would be left with the option of trying to decrypt the information which he won't be able to resolve because he doesn't have the key to decrypt the information.

2.7.2 Traditional uses of Cryptography

Primarily, the encryption part of cryptography is used as a mechanism for protecting sensitive information from unauthorized parties. This includes encrypting information for stored data, as well as encrypting information to enable secure communication (Conklin et al 2004). Should the eavesdropper manage to intercept a message, it should be impossible to read the message once it is encrypted.

2.7.3 Encryption Algorithms and the Cryptographic Key

Auguste Kerckhoff formulated the first principles of cryptographic engineering in 1883 (Petitcolas, Anderson & Kuhn 1999). The Kerckhoff principle states that the technique of encryption might be publicly known, but knowledge of the key is crucial to decrypt the message (Moerland 2003). This key is used both in the encryption phase as well as in the decryption phase, and without the key the encrypted message cannot be deciphered, even when the encryption algorithm is known. Modern encryption algorithms can be divided into two groups, namely symmetric encryption and asymmetric encryption, based on the functionality of the keys in each technique. Also known as secret-key encryption, symmetric encryption systems require that the sender and receiver have the same secret key. This single key is required for both encryption and decryption of the message. The principle of asymmetric encryption systems, also referred to as public key encryption is that parties, the sender as well as the receiver, have a pair of keys. One of the keys is publicly available while the other is kept private. Both of these encryption algorithms offer security services that can counteract the vulnerabilities of communication over an insecure channel. In order to compare cryptography with steganography

2.7.4 Security Services Offered by Cryptography

Confidentiality is the most fundamental security service offered by cryptography, through the implementation of encryption algorithms. Both symmetric, as well as asymmetric encryption algorithms provide the privacy of data. In both, however, it is the technique used and the length of the keys that ensure the level of secrecy of the information. When a message is sent, both the sender and the receiver need to know that the information was

not altered during the communication process; this alteration could have been intentional or unintentional. Cryptographic hash functions are thus used to ensure the integrity of data. By using hash functions, combined with cryptographic keys, MACs provide data integrity (Gollmann 1999) as well as authentication. The sender uses a shared secret key to compute a MAC for a specific message. When the receiver computes the MAC and compares it with the MAC received from the sender, the receiver can determine that the message was not altered in transit. Through a comparison of the two MAC values, the receiver can also determine that the message came from the person from whom it is claimed to have come, thus offering the identification and authentication of the sender. Digital signature schemes also offer data origin authentication (Schneier 1963), as well as support non-repudiation (Gollmann 1999). Based on the same principles as asymmetric encryption, digital signature schemes encrypt the message with a private key. The encrypted message acts as a signature, since only a specific private key could have produced the specific result. To summarize, of the five security services identified by the ISO 7498-2, cryptography offers the following:

- Confidentiality
- Data integrity
- Identification and authentication
- Non-repudiation

However, the investigation into cryptography does not stop here since the problems associated with cryptography also forms part of the comparison measures.

2.7.5 Encryption Problems

Starting with the different encryption algorithms, an obvious problem with symmetric encryption is that the communication can be compromised if the key is stolen. This causes another problem: the secure distribution of keys (Schneier 1963). Key distribution involves either both parties to meet face-to-face, the use of a trusted courier, or communicating the key through an existing cryptographic channel. The first two options are often impractical as well as unsafe, while the third depends on the security of a previous key exchange. It is also not enough to distribute the keys securely: keys have to be stored securely, used securely and ultimately destroyed securely. Public key encryption solves the key distribution problem of symmetric encryption, but not without problems of its own. The mathematical functions that public key encryption relies on has not yet been proven to be unsolvable (Gisin, 2002). At the moment algorithms to quickly calculate the mathematical relationship between the public/private key pair in order to use the one key to uncover the other, do not exist but cannot be ruled out. If a scientist were to develop such an algorithm, the encryption method might be compromised and the algorithm will be vulnerable (Gisin, 2002). Cryptography then also has the added limitations ensued by law enforcement as discussed in the first chapter. Finally, all the security services offered by cryptography are vulnerable to cryptanalysis the study of mathematical functions that attempts to defeat the security of cryptographic mechanisms (Menezes, van Oorschot & Vanstone 1996). Certain encryption algorithms, as well as certain hash functions, have already been broken by cryptanalysis (Wang & Yu 2005; Gilbert & Peyrin 2010; Bogdanov, Khovratovich &

Rechberger 2011). According to Gollmann (1999), cryptography is rarely a solution to a security problem, but more often a mechanism to convert one problem into another. By implementing cryptography in a security system, the problem is often only converted from a secure communication problem into a key management problem. This is usually done in the hope that the resulting problem will be easier to solve than the original one. To summarize, cryptography suffers from:

- Key distribution problem
- Mathematical vulnerabilities of asymmetric encryption
- Legal limitations by governments
- Cryptanalysis

Thus far, background information on cryptography was given, security services have been discussed, as well as common problems. The question now remains whether steganography can be seen as a suitable alternative to cryptography.

In order to answer this question, Basic concepts are described and the original uses of steganography are highlighted. Most importantly, the security services offered by steganography, a number of methods are been used to prevent data or information in transit in order to prevent attackers and hackers from mal-handling of data in transit thus giving maximum protection to information in transit.

2.8 Steganography and Cryptography

Cryptography and steganography achieve separate goals. Cryptography conceals only the meaning or contents of a secret message from an eavesdropper. However, steganography conceals even the existence of this message (Lou and Liu, 2002). Furthermore, steganography provides more confidentiality and information security than cryptography since it conceals the mere existence of secret message rather than only protecting the message contents. Therefore, one of the major weaknesses of cryptosystems is that even though the message has been encrypted, it still exists. Even though both cryptographic and steganographic systems provide secret communications, they have different definitions in terms of system breaking. A cryptographic system is considered broken if an attacker can read the secret message. However, a steganographic system is considered broken if an attacker can detect the existence or read the contents of the hidden message. Moreover, a steganographic system will be considered to have failed if an attacker suspects a specific file or steganography method even without decoding the message. As a result, this consideration makes steganographic systems more fragile than cryptography systems in terms of system failure. Additionally, steganographic systems must avoid all kinds of suspicion in order to achieve security and not be considered failed systems. Since steganography adds an extra layer of protection to cryptography, combining steganography and encryption gives the ultimate in private communication. Therefore, the purpose of steganography is to complement cryptography and to avoid raising the suspicion of system attackers but not to replace cryptography.

2.8.1 Steganography Defined

Linguistically, steganography means secret writing since the word "Steganography" is originally made up of two Greek words steganos (secret) and graphy (writing). Practically, it means the art and science of hiding or camouflaging secret data in an innocent looking dummy container in such a way that the existence of the embedded data is imperceptible and undetectable (Bailey et al., 2004; Cachin, 1998; Kahn, 1996). Therefore, steganography is the process of hiding secret data within public information. Secret data can be a plaintext or cipher text, or any kind of data that can be hidden in digital media. Since all kinds of secret data must be translated into binary, we always hide binary data whatever this secret data or file is. However, types of cover files that can be used for steganography will be presented in the next section. Basically, digital steganography can be considered as a multidisciplinary field since it combines digital signal and data compression methods, information theory, signal coding theory, digital communication theory, digital signal processing, cryptography and the theory of human visual perception, all employed to satisfy the needs of information security (Cole, 2003; Rabah, 2004).

2.9 Steganography Techniques

2.9.1 Cover Files used for Steganography

Basically, cover files represent the container of hidden data or secret messages. Additionally, some parts or characteristics of cover files will be modified, changed, or manipulated in order to hide these secret messages. However, these manipulations, which occur during the hiding procedure, should remain imperceptible to anyone not involved in the communication process. Therefore, the appearance or format of cover files must remain

intact after hiding the secret data. As a result, it is not possible to use all types of files or data as cover files of steganography since every cover file must have a sufficient redundant area to be replaced by the secret message (Katzenbeisser and Petitcolas, 2000). There is a variety of files that can be used as cover files of steganography such as executable files (i.e. exe files), HTML files, XML files and TCP headers. Essentially, many kinds of digital media such as image, audio, text, and video files can be used as cover files of steganography. However, the ability of such files to embed secret data depends on the availability of redundant or insignificant areas within these files. Thus, the cover files represent the container of hidden data and their size may determine the secret data size that can be embedded. To this end, cover files are the fundamental component of steganographic systems. However, the relationship between cover files and the other main components of steganographic systems will be discussed in the next section.

2.9.2 Main Components of Steganographic Systems

Steganographic systems have one general principle, described by Katzenbeisser and Petitcolas (2000) as follows. The sender (Alice), who wants to send a secret message (m) to the recipient (Bob), randomly chooses a harmless cover file (c). Afterwards, Alice embeds the secret message (m) in the cover (c) and probably uses a stego key (k). As a result, Alice gets a stego file (s) which must be undistinguishable from the cover file (c) neither by a human nor by a computer system. Therefore, the stego file (s) represents the original (cover) file (c) along with the secret message (m) embedded inside this cover file. Then, Alice transmits the stego file (s) to Bob over a communication channel. The purpose

of the system is to prevent Wendy (a third party) from observing or noticing the hidden message (m). On the other side, Bob extracts the embedded message (m) since he knows the embedding method and the stego key (k) used in the embedding process (Katzenbeisser and Petitcolas, 2000). Only the transmitter and the intended recipient should have the stego key. Therefore, most of steganographic systems prompt users to provide a stego key or password when they try to embed information in a cover file.

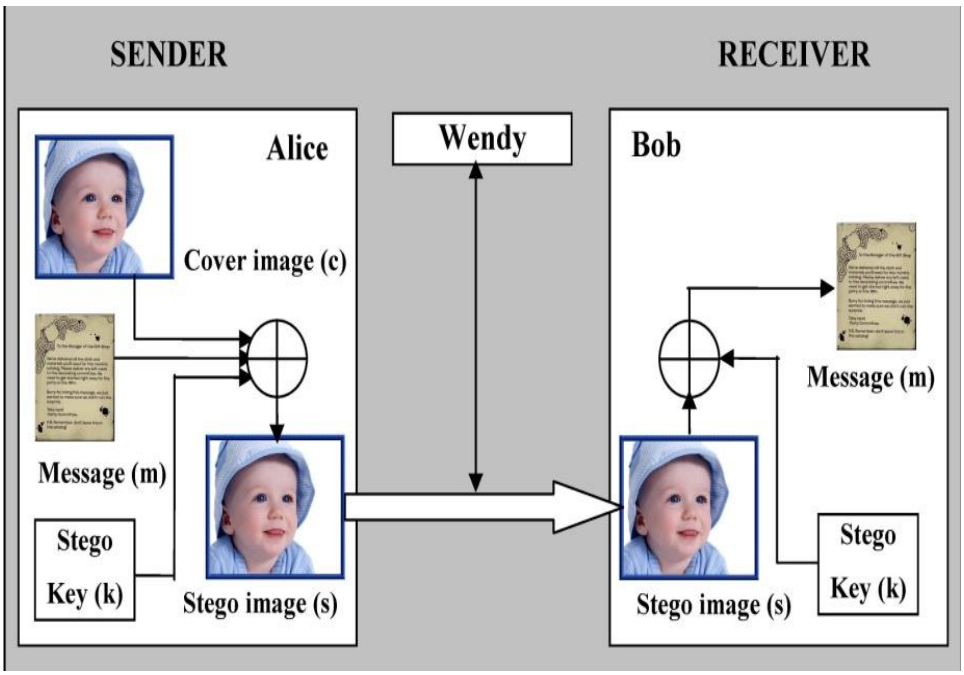

Figure 7: General Principle of Image Steganographic System

Image source: (Adel, 2010)

Sometimes, attackers (like Wendy) can detect a hidden message in a stego file and determine how the message was embedded, but they are unable to extract the hidden message. This system is known as a secure steganographic system because the secret

message is unreadable unless one has the related stego key. Therefore, these stego keys must be chosen as strong as possible in order to prevent attackers from breaking the steganographic systems using all possible stego keys (Cox, 2008). Thus, the security of steganographic systems must satisfy Kerckhoff's principle. Accordingly, the security of steganographic systems must be based on the assumption that attackers have full knowledge of the steganographic system design (the embedding and extracting algorithm). However, attackers only miss the stego key to suspect that a secret communication is taking place. Therefore, most of steganographic systems available nowadays meet this principle (Rabah, 2004). If the stego key used for embedding and the one used for extraction are the same, the steganographic system is symmetric. However, if these two stego keys are different then the steganographic system is asymmetric (Liu and Liao, 2008). The next section now describes three general types of steganography attacks.

2.10 Steganography Methods of Classification

There are two general approaches to classify steganographic systems. The first approach is based on the type of cover file while the second approach is based on the hiding method or the layout of modification used in the embedding process (Cole, 2003; Katzenbeisser and Petitcolas, 2000). These two general classification approaches of steganography are explained in the next subsections.

2.10.1 Cover Type Based Classification

Since many kinds of digital media can be used as cover files of steganography, the first approach of classification breaks down steganography according to the type of the cover

file used. However, the properties of these cover files vary from one type to another and these properties control how the secret data can be hidden in these cover files. To this end, knowing the type of cover file can give us an indication or idea where the secret data might be hidden (Cole, 2003). Mostly, steganographic systems are classified according to the cover file used. Accordingly, different steganography types can be distinguished such as: image, audio, video, text, and HTML steganography. For example, the steganographic system that uses digital images as cover files is an image-based steganographic system.

2.10.2 Hiding Method-Based Classification

Regardless of the cover type used for data hiding, steganography can be classified according to the method used to hide secret data. Furthermore, this approach of steganography classification is the most preferred approach in the steganography research community. Accordingly, there are three ways to hide secret data in cover files: insertion-based, substitution-based, and generation-based method (Cole, 2003; Kipper, 2004).

2.10.3 Insertion-Based Method

This method depends on finding some areas in cover files which are usually ignored by applications that read this cover file and then embedding the secret data in these areas. Since this method inserts the secret data inside the cover file, the size of the stego file would be larger than the size of the cover file. As a result, the main advantage of this method is that the contents of the cover file would not be changed after the embedding process since this method relies on accumulating or adding the secret data to the cover file. An example of such a method is using a Word document to write a secret message in the areas between the end text and begin-text markers. Because of the configuration of Word documents,

which depends on ignoring anything written in such areas, the hidden message will not appear when this document is viewed in Word (Cole, 2003; Kipper, 2004).

2.10.4 Substitution-Based Method

Unlike the insertion-based method, this method does not add the secret data to the cover file data. However, substitution-based method depends on finding some insignificant regions or information in cover files and replacing this information with the secret data. Therefore, the sizes of both the stego file and the cover file are similar since some of the cover data is just modified or replaced without any additional data. However, the quality of the cover file can be degraded after the embedding process. Additionally, the limited amount of insignificant information in cover files restricts the size of secret data that can be hidden (Cole, 2003; Kipper, 2004).

2.10.5 Generation-Based Method

Unlike both methods explained above, this method does not need a cover file since it uses secret data to generate appropriate stego files. One of the steganography detection techniques depends on comparing cover files with their stego files. Therefore, one advantage of the generation-based steganography is preventing such kind of detection since only stego files are available and there is no cover files used. The major limitation of this method is the limited stego files which can be generated. Moreover, the generated stego files might be unrealistic files for end users (e.g. an image contains different shapes and colors without any sense or a text without any meaning). Therefore, the main media for such techniques are random looking images and English text files (Cole, 2003).

2.11 Related Works

2.11.1 Steganography vs Watermarking:

Steganography pay attention to the degree of Invisibility while watermarking pay most of its attribute to the robustness of the message and its ability to withstand attacks of removal, such as image operations(rotation, cropping, filtering), audio operations(rerecording, filtering)in the case of images and audio files being watermarked respectively.

It is a non-questionable fact that delectability of a vessel with an introduced data (steganographic message or a watermark) is a function of the changeability function of the algorithm over the vessel.

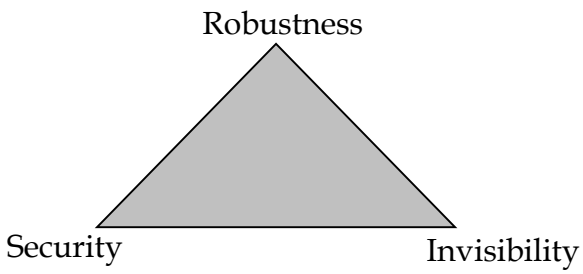

Figure 8: showing steganography characteristics

That is the way the algorithm changes the vessel and the severity of such an operation determines with no doubt the delectability of the message, since delectability is a function of file characteristics deviation from the norm, embedding operation attitude and change severity of such change decides vessel file delectability.

A typical triangle of conflict is message Invisibility, Robustness, and Security. Invisibility is a measure of the in notability of the contents of the message within the vessel.

Security is sinominous to the cryptographic idea to message security, meaning inability of reconstruction of the message without the proper secret key material shared.

Robustness refers to the endurance capability of the message to survive distortion or removal attacks intact. It is often used in the watermarking field since watermarking seeks the persistence of the watermark over attacks, steganographic messages on the other hand tend to be of high sensitivity to such attacks. The more invisible the message is the less secure it is (cryptography needs space) and the less robust it is (no error checking/recovery introduced).The more robust the message is embedded the more size it requires and the more visible it is.

2.11.2 Image Steganography and bitmap pictures:

Using bitmap pictures for hiding secret information is one of most popular choices for Steganography. Many types of software built for this purpose, some of these software use password protection to encrypting information on picture. To use these software you must have a 'BMP' format of a pictures to use it, but using other type of pictures like "JPEG", "GIF" or any other types is rather or never used, because of algorithm of "BMP" pictures for Steganography is simple. Also we know that in the web most popular of image types are "JPEG" and other types not "BPM", so we should have a solution for this problem.

This software provide the solution of this problem, it can accept any type of image to hide information file, but finally it give the only “BMP” image as an output that has hidden file inside it.

2.11.3 Data encryption using LSB matching algorithm and Reserving Room before Encryption

Here we are investigating the data hiding technique which is reversible in nature. Using the encrypted image as a cover data in which the data is embedded. Reversible data hiding is a technique to embed additional message into some distortion unacceptable cover media, in fields such as military or medical images, with a reversible manner so that the original cover content can be perfectly recovered after extraction of the hidden message(Kede, Weiming, Xianfeng, Nenghai and Fenghua 2013).

Data hiding is referred to as a process to hide data, i.e, the data hiding process links two sets of data, a set of the embedded data and another set of the cover media data. In most cases of data hiding, the cover media becomes distorted due to data hiding and cannot be inverted back to the original media. That is, cover media has permanent distortion even after the hidden data have been removed. In some applications, such as medical diagnosis and law enforcement it is desired that the original cover media can be recovered efficiently with no loss.(Kede et. al., 2013)

In proposed method can achieve real reversibility, that is, data extraction and image recovery are free of any error. If we reverse the order of encryption and vacating room, i.e., reserving room prior to image encryption at content owner side, the RDH tasks in encrypted

images would be more natural and much easier which leads us to the novel framework, “reserving room before encryption (RRBE)”. Not only does the proposed method separate data extraction from image decryption but also achieves excellent performance in two different prospects. Real reversibility is realized, that is, data extraction and image recovery are free of any error. For given embedding rates, the PSNRs of decrypted image containing the embedded data are significantly improved; and for the acceptable PSNR, the range of embedding rates is greatly enlarged.

Reversible data hiding in encrypted images is a new topic drawing attention because of the privacy preserving requirements from cloud data management. Previous methods implement RDH in encrypted images by vacating room after encryption, as opposed to which we proposed by reserving room before encryption. Thus the data hider can benefit from the extra space emptied out in previous stage to make data hiding process effortless. The proposed method can take advantage of all traditional RDH techniques for plain images and achieve excellent performance without loss of perfect secrecy. Furthermore, by this novel method can achieve reversibility as well as separate data extraction and greatly improvement on the quality of marked decrypted images

2.11.4 Proposed System for data hiding using Cryptography and Steganography

According to the method presented in this paper, the message is inserted into the DCT domain of the host image. The hidden message is a stream of “1” and “0” giving a total number of 56 bits. The transform is applied to the image as a multiple factor of 8x8 blocks. The next step of the technique after the DCT is to select the 56 larger positive coefficients,

in the low-mid frequency range. The high frequency coefficients represent the image details and are vulnerable to most common image manipulation like filtering (Johnson and Katzenbeisser, 2000) compression etc. Of course one might argue that this is the place where changes that come from watermarking are more imperceptible, but this is true only if we're speaking of small sized blocks. Our scheme is applied to the whole image and since robustness is the main issue, the low and mid frequency coefficients are the most appropriate. The selected coefficients *ci* are ordered by magnitude and then modified by the corresponding bit in the message stream. If the *i*th message bit ***s(i)*** to be embedded is "1", a quantity *D* is added to the coefficient. This *D* quantity represents the persistence factor.

This Project is mainly developed in VC6.0 plate-form using VC++. Here mainly three modules involved – a) Crypto Module - AES Implementation Module b) Security Module – Newly developed technique c) Stego Module - DCT Techniques Implementation Module These modules are design as reusable components and can work independently.

CHAPTER THREE

METHODOLOGY

3.1 Introduction

This chapter adopts a hybrid algorithm using AES and LSB for cryptography and steganography for data security management system in organizations and security agencies. The process involves encrypting information, data, important document etc. termed cryptography and then hide the said information from the normal eyes so as to protect the integrity of the information involved and increase the layer of protection. Since robustness, invisibility, security, and capacity are the most important properties of this method (Cacciaguerra & Ferretti, 2013), the researcher starts the process by decomposing the ground for data encryption process using the most recent technique i.e. encrypting data using the Advance Encryption Algorithm (AES) and afterwards hide the information from the normal eyes i.e. hiding the data at the back of an image e.g. making a dog look like a cockroach so as to confuse the attacker or intruder as the case may be to secure data (steganography).

User needs to run the application. The user has two tab options – encrypt and decrypt. If user select encrypt, application give the screen to select image file, information file and option to save the image file. If user select decrypt, application gives the screen to select only image file and ask path where user want to save the secrete file.

This project has two phases – Encrypt and Decrypt of file using the Advance Encryption standard (AES) algorithm and the second part which lies in the Steganography aspect, which is the Encryption and Decryption Using Least Significant Bit (LSB).

In encryption the secret information is hiding in with any type of image file. Decryption is getting the secret information from image file.

3.2 AES Algorithm:

Input: Secrete message

Output: Reconstructed message ready to be steganographed

1. Start
2. Input secrete message
3. Application of keys for encryption process to take place
4. Encryption process
5. Generation of encrypted message (Ready for Steganography)
6. Application of secrete key
7. Decryption process (To get secrete message back to its original form)
8. End

3.2.1 Architectural Design for the AES Algorithm

The figure below shows the flow of the process of encryption by the AES algorithm.

The process actually starts by the input of the secrete message which requires integrity in which the sender and the receiver doesn't want a third party to have knowledge of.

The next stage is the input of secrete key that is known to only the sender and the receiver alone so as to give the information in transit a high-level of protection then the encryption process can take place after the insertion of the secrete key with all the above done we now have our encrypted files or folders as the case maybe

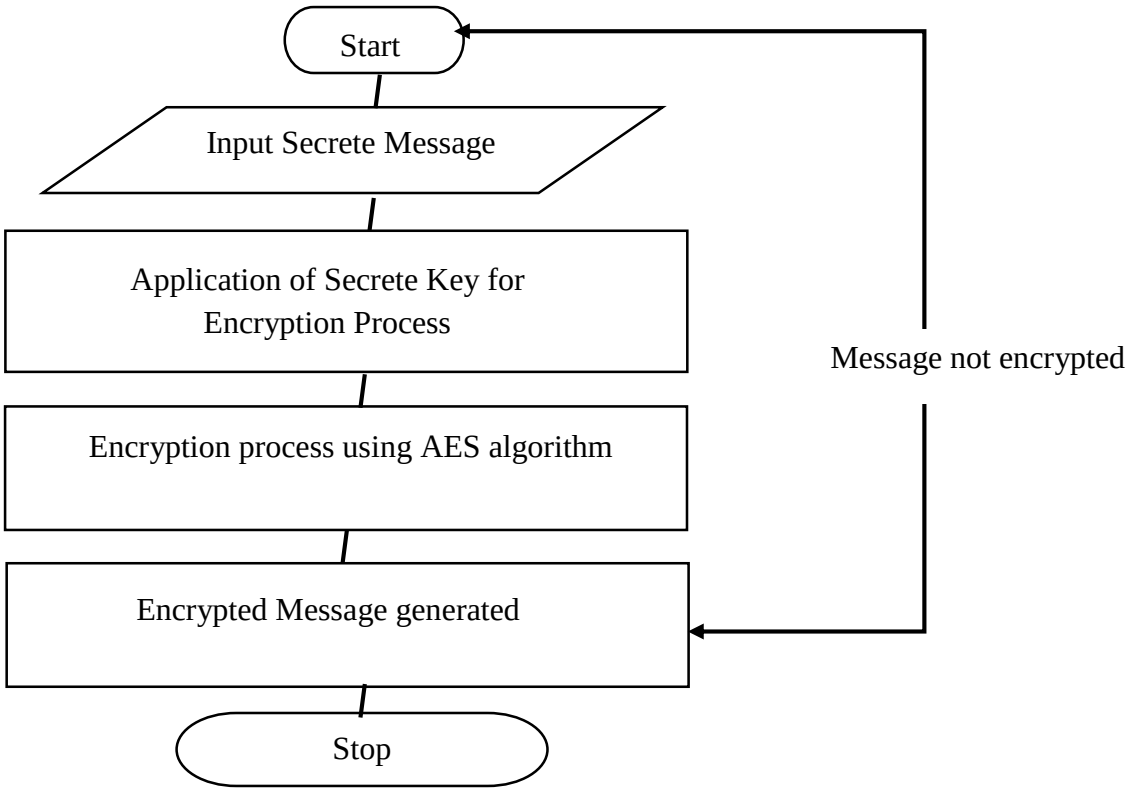

Figure 9: Flow chat for the AES algorithm

3.2.2 Stego Algorithm: (At the sender's end)

Input: Cover image and encrypted message.

Output: Stego image with secrete data.

Start

1. Normalized the encrypted message.

2. Transform the cover image using discrete wavelet transform

3. Embedding the normalized encrypted image in vertical detail coefficients and diagonal detail coefficients.

4. Inverse LSB of all the sub-bands.

5. De-normalize (decryption using secrete key).

6. Stego image is generated.

7. Stop.

The process then begins with the inputs that is the cover image and the already encrypted secrete message then the encrypted message is then normalized, the result gotten is then transformed using discrete wavelet transform, embedding the normalized image for portability and compression i.e in vertical details coefficients and diagonal detail coefficients then the result is then inverted i.e the sub-bands. The result is then de-normalized using the secrete key this can be termed decryption then the initial original image is then gotten back.

Extraction Algorithm (At the receiver's end)

Input: Stego Image.

Output: Encrypted message.

Begin

1. Transform the stego image using discrete wavelet transform.

2. Extracting the normalized encrypted image from the vertical detail coefficients and diagonal detail coefficients.

3. Normalization.

4. The encrypted message is generated.

End.

It must be noted that the encryption and embedding processes are achieved in the transmitter to generate the stego image, while the extraction and decryption processes are achieved in the receiver to recover the original message.

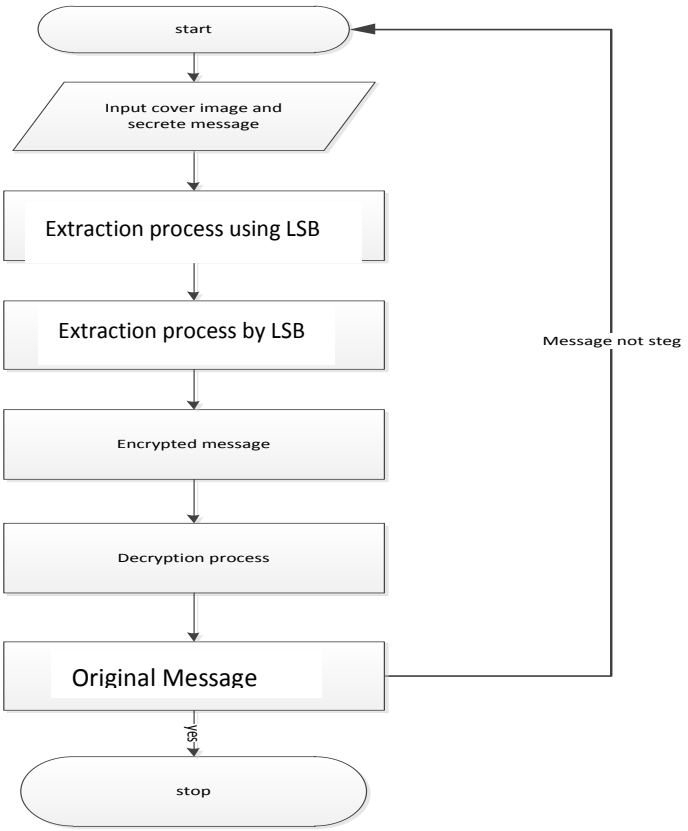

Figure 10: The flowchart diagram for the steganography system

3.3 The Hybrid Algorithm (Cryptography and Steganography Framework)

The cryptography and steganography algorithms are combined to form a hybrid algorithm as presented in figure.

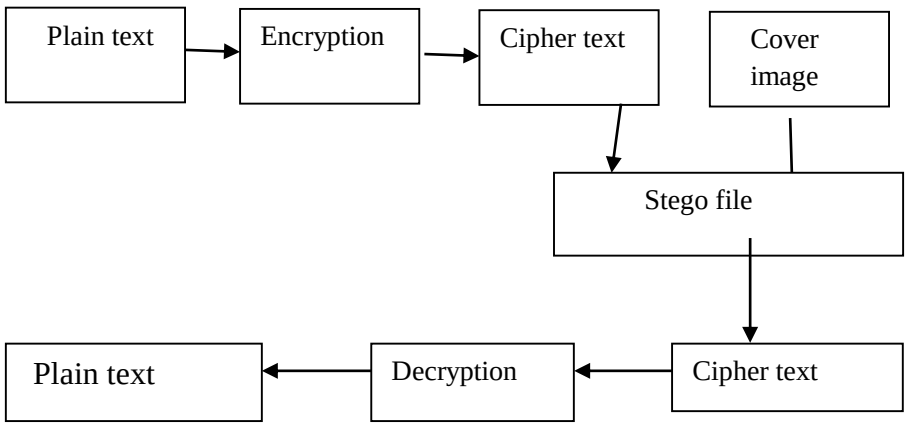

Figure 11: showing the hybrid algorithm

3.3.1 Combination of cryptography and steganography

The combination of the cryptography process as in the encryption using the AES algorithm and steganography using LSB algorithm gave birth to a more secure process of data security and protection from intruders and attackers.

The process involves the use of AES algorithm that is used for the encryption, the process starts by an input of the secrete message, the system alerts the user for the input of the encryption key which is the password afterward the encryption process follows by the advance encryption algorithm finally the encrypted message is generated and if the process fails maybe because of the any of the necessary factors the loop returns to the beginning and the whole process is taken again. It follows that the resulting encrypted message is

ready for steganography. The steganography algorithm that is the LSB algorithm takes up the task from where the AES algorithm has stopped worked and completes the process for a more secure environment to work with. The process begins by an input of cover image and the secrete message that has been encrypted is embedded into the cover image which is now a protected data in other word a cat will look like a rat so as to confuse attacker. For the process to be completed at the receiver's end the receiver then extracts the image been sent to him/her through the LSB algorithm the encrypted message is gotten the receiver then decrypts the message through the secrete key and he/she gets the original message with no information or data loss.

3.4 Software Requirements:

- **.NET Framework 3.5**

Hardware Requirements:

Processor: Preferably 1.0 GHz or Greater.

RAM : 512 MB or Greater.

3.5 System Analysis & Design

Hybridized system that integrate the LSB and AES algorithm together requires any type of image file and the information or message that is to be hidden. It has two modules encrypt and decrypt.

Microsoft .Net framework prepares a huge amount of tool and options for programmers that they simples programming. One of .Net tools for pictures and images is auto-converting most types of pictures to BMP format. I used this tool in this software called "CryptoSteg" that is written in C#.Net language and you can use this software to encrypt and hide your information in any type of pictures without any converting its format to BMP (software converts inside it).

The algorithm used for Encryption and Decryption in this application provides using several layers lieu of using both the AES and the LSB layer of image. Writing data starts from last layer; because significant of this layer is least and every upper layer has doubled significant from its down layer. So every step we go to upper layer image quality decreases and image retouching transpires.

The encrypt module is used to hide information into the image; no one can see that information or file. This module requires any type of image and message and gives the only one image file in destination.

The decrypt module is used to get the hidden information in an image file. It take the image file as an output, and give two file at destination folder, one is the same image file and another is the message file that is hidden it that.

Before encrypting file inside image we must save name and size of file in a definite place of image. We could save file name before file information in LSB layer and save file size

and file name size in most right-down pixels of image. Writing this information is needed to retrieve file from encrypted image in decryption state.

The graphical representation of this system is as follows:

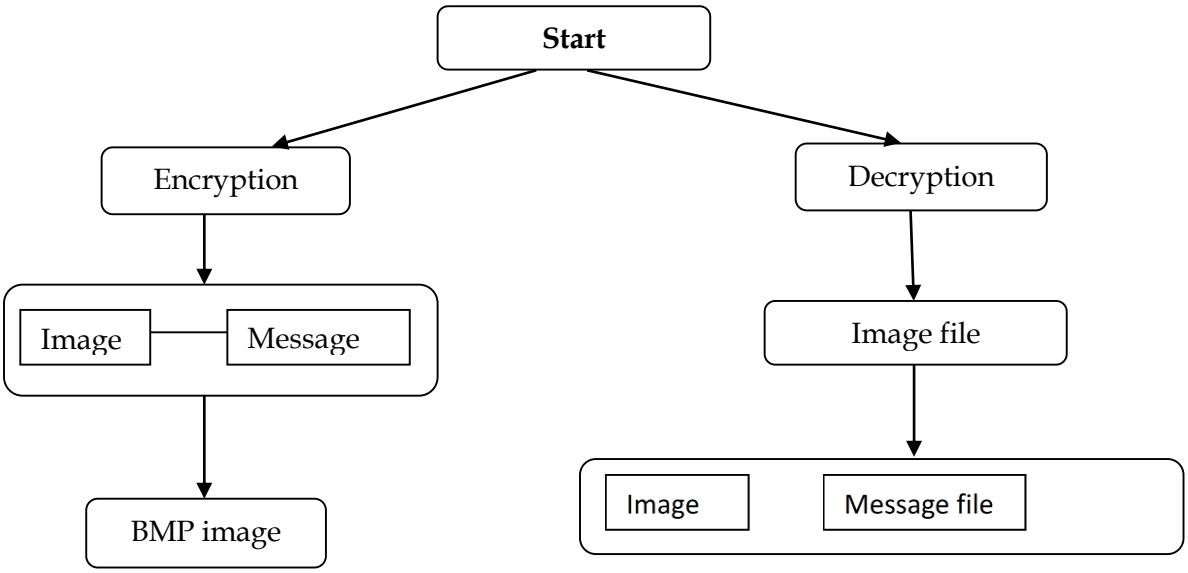

Figure 12 showing the graphical representation of how the system works

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.1 Introduction

This chapter presents the implementation of the data security using Advanced Encryption Standard (AES) and Least Significant Bit (LSB) as discussed in chapter three.

4.2 Implementation Result

The implementation results of the the hybridized design consisting of Advanced Encryption Standard (AES) and Discrete Wavelength Transform are presented in the following interfaces:

(i) The login page;

(ii) The login page process

(iii) The action page;

(iv) The encryption environment page;

(v) The encryption process page;

(vi) The encryption process with the wrong private key displaying error message

4.3 Login Page

Each time the application is run, a page appears which shows the login credentials that prompts the admin to enter username and password then click on the sign in button before he/she can be allowed access into the software application environment in order to begin

the cryptography or steganography or the combination of the two process depending on what the intended user has in mind. This interface is as presented in figure 13 below.

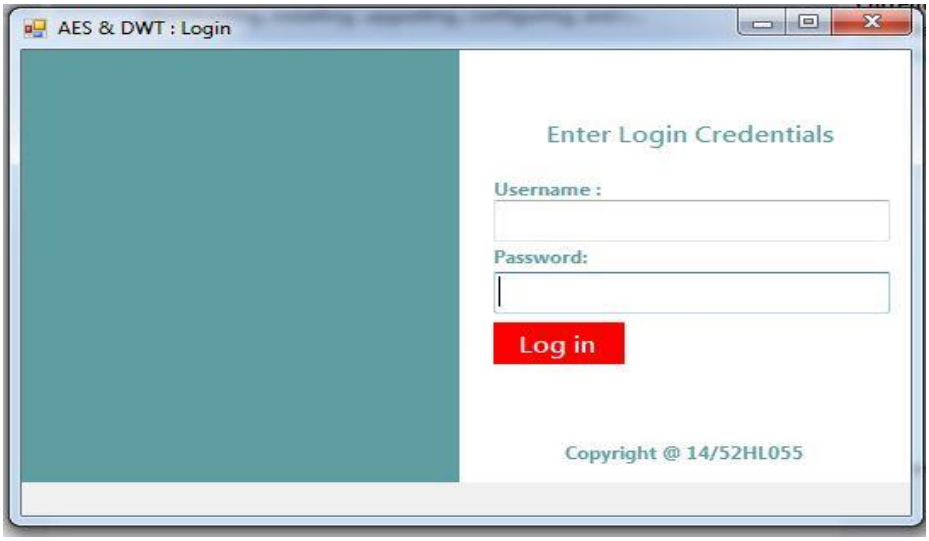

Figure 13 showing the login page of the system.

4.4 The Login Page Process

After the admin enters the username and password. The admin then clicks on the sign-in button and the application takes the admin to the action Page as seen figure 14

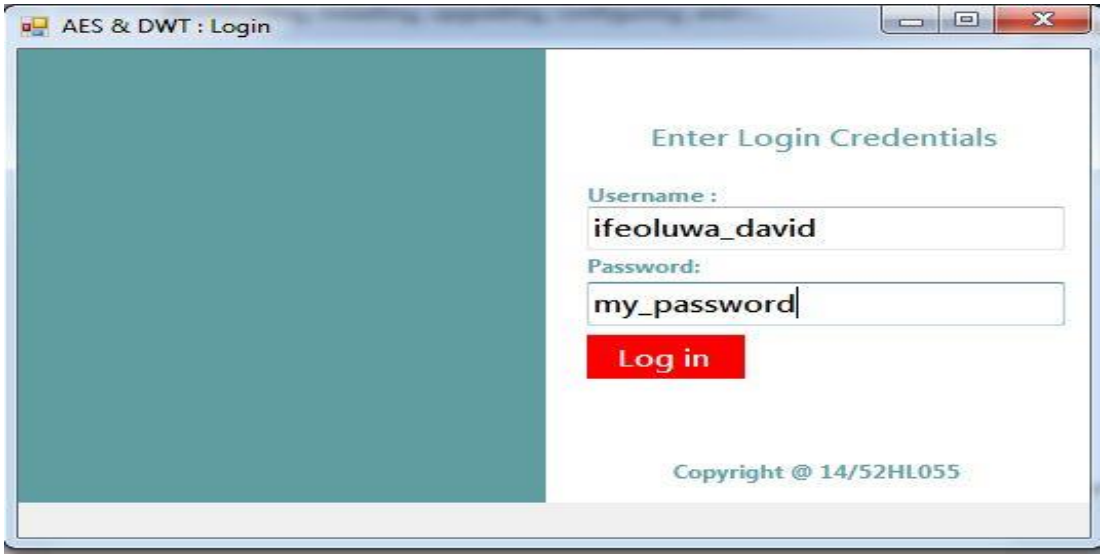

Figure 14: The Login Credentials Page

4.5 The Action Page

The action page interface is where the cryptography and steganography options are displayed. The admin can chooses which technique he wants to execute and follows the necessary instructions need to complete the task from here. It must be noted that one technique enhances the other in other words the two techniques goes hand-in-hand.

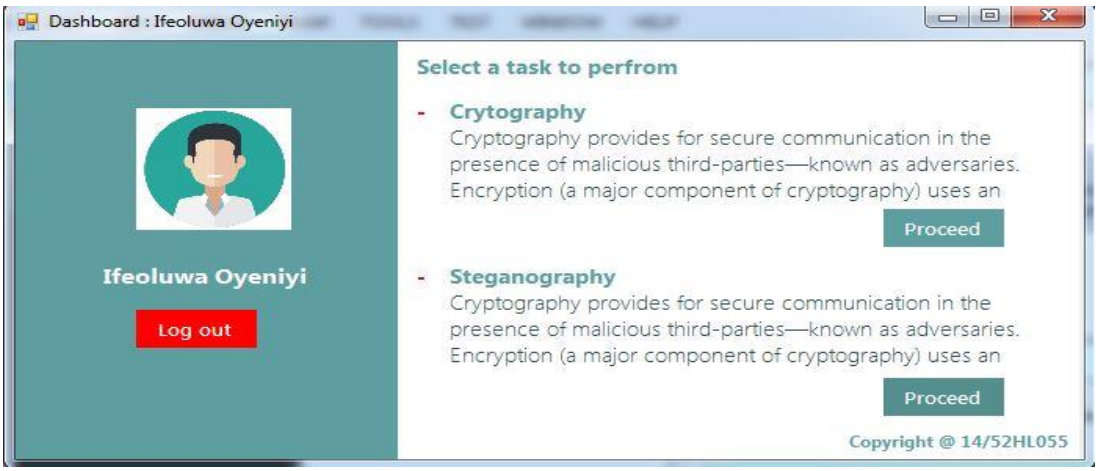

Figure 15: The Action Page

4.6 The Encryption Environment

Once the action page shows up the admin then clicks on either the cryptography button or the steganography button. If admin clicks on the cryptography button, this command takes him/her to the Encryption environment page as seen in figure 16 where the secret messages can be encrypted. If on the other hand, the admin clicks on the steganography option button.

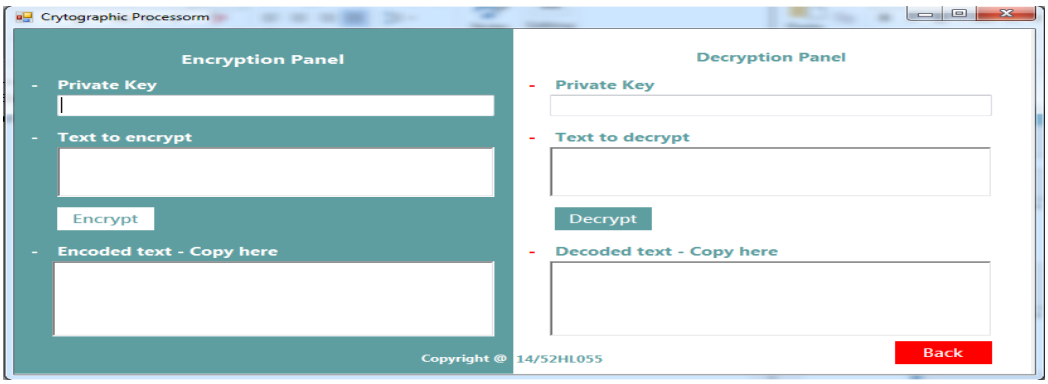

Figure 16: The Encryption environment page

4.7 The Encryption process

The encryption environment interface page (Figure 17) enable the message sender to enters his/her password and the message to be encrypted, then click on the encrypt button. The message as encrypted will be displayed as shown in figure 17.

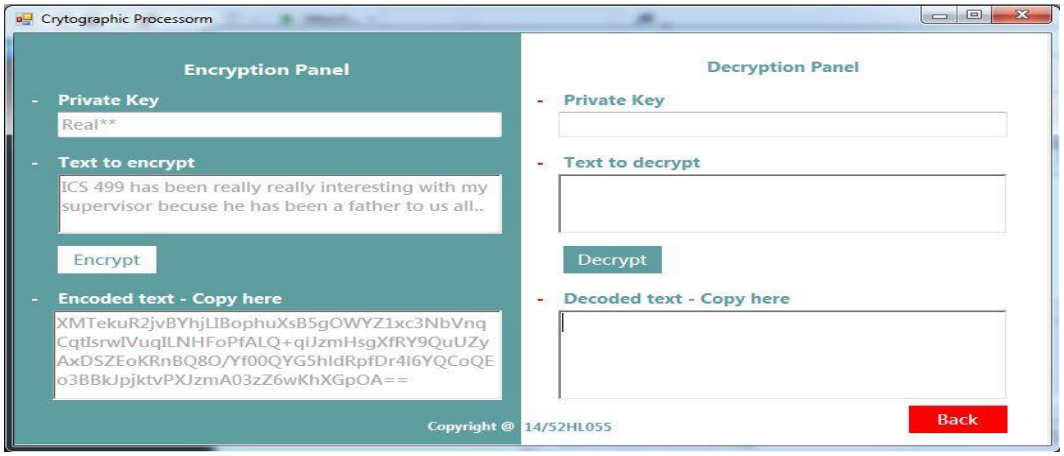

Figure 17: encryption environment after encrytpion has been done

4.8 Decryption environment process.

The decryption environment interface page (Figure 18) enable the message that is encrypted to be sent to the sender then the same private key will be shared to the receiver so as to decrypt the encrypted message. As shown in figure 18

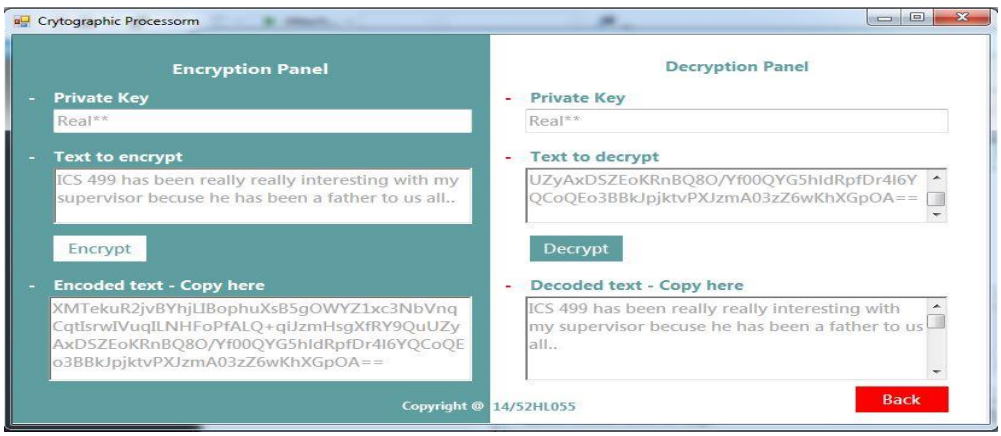

Figure 18 Decryption Environment

4.9 Error Page

one good thing of about the system is that when the private keys are different it will pop up a error message that mill deliver gibberish to you on the decoded text.

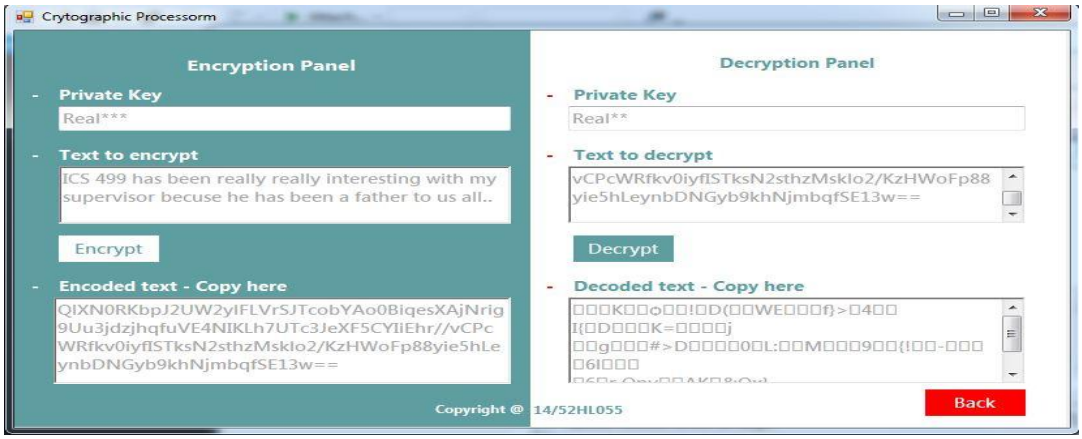

Figure 19 Error page

4.10 The Action Page to the Steganography Aspect

The action page interface is where the cryptography and steganography options are displayed. The admin can chooses which technique he wants to execute and follows the necessary instructions need to complete the task from here. It must be noted that one technique enhances the other in other words the two techniques goes hand-in-hand. Then the Steganography process will be clicked on.

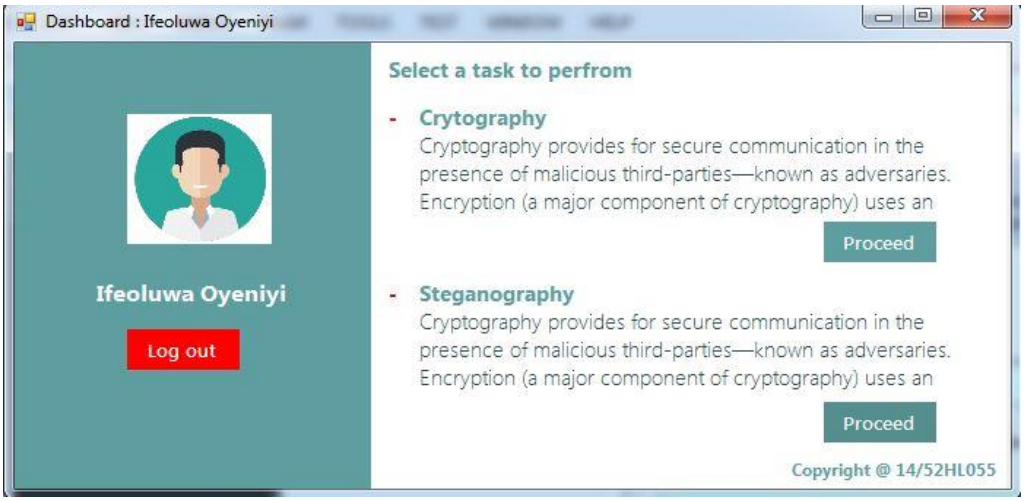

Figure 20 showing the action page to the steganography part

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

A serious concern in information communication technology world now is the level of increased intrusion and attack of information and data mostly in transit which are perpetrated by hackers and attackers who breaks into an organization or security agencies' archive and fetch useful secret information which they use for their own benefit either financial or to aide their work thereby causing detriment to the organization in question.

This dissertation work has reached a far reaching goal by developing a system that helps protect secret messages i.e files and folders from attackers and intruders by designing a system using cryptography and steganography. The two techniques are being combined to form a more secure and integrity proofing system, the system involves using cryptography to encrypt secret messages using Advance Encryption Algorithm and steganography to hide the message from the normal eyes the message is then embedded in an image using Least Significant Bit (LSB) before it is being sent to the intender user to avoid being seen by the third eyes, this system confuses a third party as Lion look like a rat to intruders and attackers. The receiver then uses a password before he/she can have access to the secret message. The process of accessing the secret message is called decryption. It follows that if this system is properly implemented using the appropriate tools and equipment the

problem associated with intruders, attackers and hijack of information in transit will be reduced to the barest minimum or even a forgotten.

5.2 Recommendation for Future Work

For the purpose of this research work the researcher is looking forward to designing a system that will encrypt secret information and at the same time hide the existence from the normal eyes and its existence will not be felt in other word effort should be made to encrypt information and hide it from the physical that is the message will be saved on the hard disk and the normal hard disk space will be chopped off but the message cannot be discovered except the admin that hide the information comes to the rescue.

5.3 Conclusion

As information communication has being on top gear of improvement so also hackers and attackers too have been on their toes to get a way of retrieving information from organizations and agencies' database or data-bank. This dissertation work has being able to proffer solution to the problem associated with attackers and intruders breaking into an organization's database to fetch information and use for their own financial benefit or for their own advantage thereby causing detriment to the organization. This project work has been able to combine two known security algorithms for the process protection i.e Advance Encryption Algorithm and Least Significant Bit Algorithms. The AES algorithm is used to encrypt the data and the LSB is used to hide the secret message before it is being stored or sent to the intended receiver. The researcher here with the level security of the system then concludes that if this system is implemented with the appropriate tools and equipment and

all necessary trainings are being given to the operator that will be using the system then the problem of panicking with information in transit or information stored in the database will be reduced to the barest minimum if not totally eradicated in information communication technology environment.

REFERENCES

Abdelrahmaham I. C., & Bazara T. W. (2013) *Preserving query execution of fragmented outsourced data.*" Information and Communication Technology, Int. Conf., ICT-EurAsia 2013, Yogyakarta, Indonesia, March 25-29, 2013. LNCS Vol. 7804, Springer, 426-440, 2013.

Adel S. S., (2010)"*Securing against brute-force attack: A hash-based on steganographic system with rfid mutual authentication protocol using a secret value*" Computer Communications 34 (3) pp 391– 397, 2010

Aidi G., Zhang Z., & Zhou Q (2010). A novel image encryption method based on total shuffling scheme. *Optics Communications*. Vol. 284, No.12, 2775-2780.

Aisha R. & Wilson J. (2013). An Efficient Image Encryption Technique by Using Cascaded Combined Permutation. *International Journal of Computer Science and Information Security* (IJCSIS). Vol.14, No. 6, 576-588.

Altigani, A., & Barry, B. (2013, August). A hybrid approach to secure transmitted messages using advanced encryption standard (AES) and Word Shift Coding Protocol. In *Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on* (134-139). IEEE.

Arun C.H., Vipula A.C., & Suresh M. W., (2013)*Outsourcing chaotic selective image encryption to the cloud with steganography*", Digital Signal Processing 43, 28–37, 2013

Brifcani, A. M. A., & Brifcani, W. M. A. (2010). Stego-based-crypto technique for high security applications. *International Journal of Computer Theory and Engineering*, *2*(6), 835.

Cacciaguerra, S., & Ferretti, S. (2003). Data hiding: steganography and copyright marking. *Department of Computer Science, University of Bologna, Italy. http://www. cs. unibo. it/-people/phdstudents/scacciag/home_files/-teach/datahiding. pdf*, 12.

Chan, C. K., & Cheng, L. M. (2004). Hiding data in images by simple LSB substitution. *Pattern recognition*, *37*(3), 469-474.

Cheddad, A., Condell, J., Curran, K., & Mc Kevitt, P. (2010). Digital image steganography: Survey and analysis of current methods. *Signal processing*, *90*(3), 727-752

Chen, P. Y., & Lin, H. J. (2006). A DWT based approach for image steganography. *International Journal of Applied Science and Engineering*, *4*(3), 275-290.

Chan, C. K., & Cheng, L. M. (2004). Hiding data in images by simple LSB substitution. *Pattern recognition*, *37*(3), 469-474.

Conklin D. J., Ning Cai and T. Chan (2004)"*Theory of secure network coding*" Proc. of the IEEE, Vol. 99 (3) pp 421-437, March 2004.

Cox P (2008), "Robust detection of R-wave using wavelet technique," World Academy of Science, Engineering and Technology 32 2008.

Dipti N. & Neha D. Y., (2010)"*A French Anonymization Experiment with Health Data*." PSD 2010 : Privacy in Statistical Databases, 17-19 september 2010, Eivissa, Spain, 2010.

El Safy, R. O., Zayed, H. H., & El Dessouki, A. (2009, March). An adaptive steganographic technique based on integer wavelet transform. In *Networking and Media Convergence, 2009. ICNM 2009. International Conference on* (pp. 111-117). IEEE.

Fernandes, A., & Jeberson, W. (2013). A Simple Steganographic Technique with a Good Embedding Capacity. *International Journal of Darshan Institute on Engineering Research and Emerging Technology*, *2*(2), 56-61.

Gisin A. R., (2002) *General Secure Multi-Party Computation from any Linear Secret Sharing Scheme" Proc.* EUROCRYPT, Int. Conf., ICT-EurAsia 2002, Yogyakarta, Indonesia, March 25-29, 2012. LNCS Vol. 7804, Springer, 426-440, 2002.

Gollman C. (1999) "*Cryptography on FPGAs:* "*Exploring the Feasability of Fully Homomorphic Encryption*" ACM Transactions on Embedded Computing Systems, vol. 3, n. 3, 534–574, August 1999.

Gollman C. (1999) "*Cryptography on FPGAs:* "*Exploring the Feasability of Fully Homomorphic Encryption*" ACM Transactions on Embedded Computing Systems, vol. 3, n. 3, 534–574, August 1999.

Guo, X. C. (2008). *Methodologies in digital watermarking: Robust and reversible watermarking techniques for authentication, security and privacy protection*. University of Toronto.

Imran, A. S., Javed, M. Y., & Khattak, N. S. (2007). A robust method for encrypted data hiding technique based on neighborhood pixels information. *World Academy of Science, Engineering and Technology*, *31*, 2007.

Johnson S., Katzenbeisser S. H., (2000)"QRS detection using wavelet transform," IJEAT: 2249 – 8958, Vol1, Issue-6, August 2000.

Juneja, M., & Sandhu, P. S. (2012). An improved LSB based steganography with enhanced security and embedding/extraction. *Pragyan: Journal of Information Technology*, *10*(2), 1-8.

Kant, C., Nath, R., & Chaudhary, S. (2008). Biometrics security using steganography. *International Journal of Security*, *2*(1), 1-5.

Kede Ma, Weiming Zhang, Xianfeng Zhao, Member, IEEE, Nenghai Yu, and Fenghua Li MARCH 2013 "*Reversible Data Hiding in Encrypted Images by Reserving Room Before Encryption", 55-67*.

Liu, R., & Tan, T. (2002). An SVD-based watermarking scheme for protecting rightful ownership. *IEEE transactions on multimedia*, *4*(1), 121-128.

Lin, C. C., & Shiu, P. F. (2010). High capacity data hiding scheme for DCT-based images. *Journal of Information Hiding and Multimedia Signal Processing*, *1*(3), 220-240.

Moerland P. (2003) "*Cryptography on FPGAs: State of the Art Implementation and Attacks*" ACM Transactions on Embedded Computing Systems, vol. 3, n. 3, 534–574, August 2003.

N. F. Johnson and S. Katzenbeisser (2000), *.A survey of steganographic techniques*., in S. Katzenbeisser and F. Peticolas (Eds.): *Information Hiding*, 43-78. Artech House, Norwood

Navas, K. A (2007). EPR. *International Journal of Biomedical Sciences*, ACM Transactions on Embedded Computing Systems *3*(1), 44-47.

Navas, K. A., Thampy, S. A., & Sasikumar, M. (2007). EPR hiding in medical images for telemedicine. *International Journal of Biomedical Sciences*, *3*(1), 44-47.

Obama, B. (2012). *Strategy to Combat Transnational Organized Crime: Addressing Converging Threats to National Security*. DIANE Publishing.

Patrick R, Ebele O & Chinedu D (2012) Robust Digital Water marking of Images using Wavelets. International Journal of Computer and Electrical Engineering, Vol. 1, No. 2, June 2012.

Petitcolas, Anderson & Kuhn (1999) *Combining Fragmentation and Encryption to Protect Privacy in Data Storage*", in ACM Transactions on Information and System Security (TISSEC), July, 1999

Reddy, M. I. S., & Kumar, A. S. (2016). Secured data transmission using wavelet based steganography and cryptography by using AES algorithm. *Procedia Computer Science*, *85*, 62-69.

Reddy, H. M., & Raja, K. B. (2009). High capacity and security steganography using discrete wavelet transform. *International Journal of Computer Science and Security (IJCSS)*, *3*(6), 462.

Singh, K. M., Singh, S. B., & Singh, L. S. S. (2007). Hiding encrypted message in the features of images. *IJCSNS*, *7*(4), 302-307.

Saraireh, S., & Benaissa, M. (2009, June). A scalable block cipher design using filter banks and lifting over finite fields. In *Communications, 2009. ICC'09. IEEE International Conference on* (pp. 1-5). IEEE.

Shakar, A. K. (2013). Enhancing the data security features of communication by means of media files through improvising the cryptographic and steganographic techniques. *ASM's International E-Journal of Ongoing Research in Management and IT*.

Tayana S. (2012) "*Encryption and Fragmentation for Data Confidentiality in the Cloud*." Foundations of Security Analysis and Design VII, 212-243, 2012.

The White House Washington (2011). Wavelet-Based Image Compression Anti-Forensics. In Proceedings of IEEE 17th international conference on image processing, Washington, 1737–1740.

Thakur, V. S., Dewangan, N. K., & Thakur, K. (2014, January). A highly efficient gray image compression codec using neuro fuzzy based soft hybrid JPEG standard. In *Proceedings of Second International Conference, Emerging Research in*

Computing, Information, Communication and Applications (ERCICA) (Vol. 1, pp. 625-631).

Vegh, L., & Miclea, L. (2014, May). Enhancing security in cyber-physical systems through cryptographic and steganographic techniques. In *2014 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)* (pp. 1-6). IEEE.

Wu, H. T., & Cheung, Y. M. (2010). Reversible watermarking by modulation and security enhancement. *IEEE transactions on Instrumentation and measurement*, *59*(1), 221-228.

Zaidan, B. B., Zaidan, A. A., Al-Frajat, A. K., & Jalab, H. A. (2010). On the differences between hiding information and cryptography techniques: An overview. *Journal of Applied Sciences(Faisalabad)*, *10*(15), 1650-1655.

Zeng, X. T., & Li, Z. (2012). Reversible data hiding scheme using reference pixel and multi-layer embedding. *AEU-International Journal of Electronics and Communications*, *66*(7), 532-539.

Appendix: Source Code

Form1.cs

```
Public Class Form1

    Private Sub ComboBox1_SelectedIndexChanged(sender As
Object, e As EventArgs) Handles
ComboBox1.SelectedIndexChanged
        Dim iAs Integer = ComboBox1.SelectedIndex
        If i = 0 Then
            PictureBox1.Image = ScaleImage(My.Resources.A,
PictureBox1.Width, PictureBox1.Height)
ElseIfi = 1 Then
            PictureBox1.Image = ScaleImage(My.Resources.B,
PictureBox1.Width, PictureBox1.Height)
ElseIfi = 2 Then
            PictureBox1.Image = ScaleImage(My.Resources.C,
PictureBox1.Width, PictureBox1.Height)
ElseIfi = 3 Then
            PictureBox1.Image = ScaleImage(My.Resources.D,
PictureBox1.Width, PictureBox1.Height)
ElseIfi = 4 Then
            PictureBox1.Image = ScaleImage(My.Resources.E,
PictureBox1.Width, PictureBox1.Height)
ElseIfi = 5 Then
getFileImage()
        End If
    End Sub

    Sub getFileImage()
        Dim opf As New OpenFileDialog()
AddHandleropf.FileOk, Sub()
                                    Dim s As String =
opf.FileName.ToLower()
                                    If s.EndsWith("jpg") Or
s.EndsWith("jpeg") Or
s.EndsWith("png") Or
```

```
s.EndsWith("png") Then
                                        Dim b As Bitmap = New
Bitmap(opf.FileName)
                                        PictureBox1.Image = b
                                    Else
                                        PictureBox1.Image   =
Nothing
                                    End If
                                End Sub
opf.ShowDialog()
    End Sub

    Private Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
ErrorLabel.Text = ""
        If TextBox1.Text <> "" And RichTextBox1.Text <> ""
And TextBox1.Text.Length >= 4 Then
Encryption.pass = TextBox1.Text
            RichTextBox2.Text                                =
Encryption.EncryptString(RichTextBox1.Text)
            RichTextBox3.Text = RichTextBox2.Text

        Else
ErrorLabel.Text = "Please supply a password and text
information to be encrypted. Password length should be
greater than 4"
        End If

    End Sub

    Function reverseEngineer() As String
        Dim bt As New List(Of Byte)
        Dim bm As Bitmap = PictureBox1.Image
        For x = 0 Tobm.Width - 1
            For y = 1 Tobm.Height - 2 Step 2
                Dim c1 As New Color(bm.GetPixel(x, y))
```

```
                Dim c2 As New Color(bm.GetPixel(x, y + 1))
                Dim str As String = ""
str = (c1.R Mod 2) &str
str = (c1.G Mod 2) &str
str = (c1.B Mod 2) &str
str = (c1.A Mod 2) &str
str = (c2.R Mod 2) &str
str = (c2.G Mod 2) &str
str = (c2.B Mod 2) &str
str = (c2.A Mod 2) &str
                Dim b As Byte = toInteger(str)
bt.Add(b)
            Next
        Next
        Dim         ret         As         String         =
System.Text.ASCIIEncoding.ASCII.GetString(bt.ToArray())
ret = ret.Replace("?", "")
        'Dim s() As String = ret.Split("==")
        'If s.Length> 0 Then
        '    ret = s(0) & "=="
        'End If
        Return ret
    End Function

    Private   Sub   NumericUpDown1_ValueChanged(sender   As
Object, e As EventArgs) Handles NumericUpDown1.ValueChanged
        Dim c1 As New Color(253, 255, 255, 0)
        Dim c2 As New Color(253, 255, 255, 0)
        Dim t As Integer = 0
        Dim N As Byte = NumericUpDown1.Value
        Dim a As New Color()
a.R = N Mod 2
        t = a.R
a.G = ((N - t) / 2) Mod 2
        t = (t + 2) * a.G
a.B = ((N - t) / 4) Mod 2
```

```
        t = (t + 4) * a.B
a.A = ((N - t) / 8) Mod 2
        t = (t + 8) * a.A
        Dim b As New Color()
b.R = ((N - t) / 16) Mod 2
        t = (t + 16) * b.R
b.G = ((N - t) / 32) Mod 2
        t = (t + 32) * b.G
b.B = ((N - t) / 64) Mod 2
        t = (t + 64) * b.B
b.A = ((N - t) / 128) Mod 2
        t = (t + 128) * b.A
        c1.R = c1.R - (c1.R Mod 2) + a.R
        c1.G = c1.G - (c1.G Mod 2) + a.G
        c1.B = c1.B - (c1.B Mod 2) + a.B
        c1.A = c1.A - (c1.A Mod 2) + a.A
        c2.R = c2.R - (c2.R Mod 2) + b.R
        c2.G = c2.G - (c2.G Mod 2) + b.G
        c2.B = c2.B - (c2.B Mod 2) + b.B
        c2.A = c2.A - (c2.A Mod 2) + b.A
        RichTextBox2.Text = toJSON(c1) &CS.CrLf&toJSON(c2)
    End Sub

    Private Sub Button2_Click(sender As Object, e As
EventArgs) Handles Button2.Click
        Dim svf As New SaveFileDialog()
AddHandlersvf.FileOk, Sub()
                                    If Not PictureBox1.Image
Is Nothing Andsvf.FileName<> "" Then
PictureBox1.Image.Save(svf.FileName,
Drawing.Imaging.ImageFormat.Png)
                                    End If
                                End Sub
svf.ShowDialog()
    End Sub
```

```
    Private Sub Button3_Click(sender As Object, e As
EventArgs) Handles Button3.Click
        Dim opf As New OpenFileDialog()
AddHandleropf.FileOk, Sub()
                                    Dim     th     As     New
System.Threading.Thread(Sub()

PictureBox1.Image = Image.FromFile(opf.FileName)

If Not PictureBox1.Image Is Nothing Then
MsgBox("Image Loaded")

Dim str As String = reverseEngineer()
MsgBox("Encrypted Message: " &str)
RichTextBox2.BeginInvoke(Sub()

RichTextBox2.Text = str

End Sub)


End If

End Sub)
th.SetApartmentState(Threading.ApartmentState.STA)
th.Start()

                                End Sub
opf.ShowDialog()

    End Sub

    Private Sub Button4_Click(sender As Object, e As
EventArgs) Handles Button4.Click 'DWT Transform
        If Not PictureBox1.Image Is Nothing Then
            Dim bm As Bitmap = PictureBox1.Image
```

```
            For y = 0 Tobm.Height - 1
                Dim HL As New List(Of Color)
                Dim LL As New List(Of Color)
                Dim midpoint As Integer = bm.Width / 2
                For x = 0 Tobm.Width - 2 Step 2
                    Dim c1 As New Color(bm.GetPixel(x, y))
                    Dim c2 As New Color(bm.GetPixel(x + 1,
y))
                    Dim c1_ As New Color()
                    c1_.R = c1.R + c2.R
                    c1_.G = c1.G + c2.G
                    c1_.B = c1.B + c2.B
                    c1_.A = c1.A + c2.A
normaliseColor(c1_)
HL.Add(c1_.clone())
                    c1_.R = Math.Abs(c1.R - c2.R)
                    c1_.G = Math.Abs(c1.G - c2.G)
                    c1_.B = Math.Abs(c1.B - c2.B)
                    c1_.A = Math.Abs(c1.A - c2.A)
normaliseColor(c1_)
LL.Add(c1_.clone())
                Next
                For index = 0 To midpoint - 2
bm.SetPixel(index,  y,  Drawing.Color.FromArgb(HL(index).A,
HL(index).R, HL(index).G, HL(index).B))

bm.SetPixel(index          +          midpoint,          y,
Drawing.Color.FromArgb(LL(index).A,            LL(index).R,
LL(index).G, LL(index).B))
                Next
            Next
            For x = 0 Tobm.Width - 1
                Dim HL As New List(Of Color)
                Dim LL As New List(Of Color)
                Dim midpoint As Integer = bm.Height / 2
                For y = 0 Tobm.Height - 2 Step 2
```

```
                    Dim c1 As New Color(bm.GetPixel(x, y))
                    Dim c2 As New Color(bm.GetPixel(x, y +
1))
                    Dim c1_ As New Color()
                    c1_.R = c1.R + c2.R
                    c1_.G = c1.G + c2.G
                    c1_.B = c1.B + c2.B
                    c1_.A = c1.A + c2.A
normaliseColor(c1_)
HL.Add(c1_.clone())
bm.SetPixel(x,   y,   Drawing.Color.FromArgb(c1_.A,   c1_.R,
c1_.G, c1_.B))
                    c1_.R = Math.Abs(c1.R - c2.R)
                    c1_.G = Math.Abs(c1.G - c2.G)
                    c1_.B = Math.Abs(c1.B - c2.B)
                    c1_.A = Math.Abs(c1.A - c2.A)
normaliseColor(c1_)
LL.Add(c1_.clone())
bm.SetPixel(x, y + 1, Drawing.Color.FromArgb(c1_.A, c1_.R,
c1_.G, c1_.B))
                Next
                For index = 0 To midpoint - 2
bm.SetPixel(x,   index,   Drawing.Color.FromArgb(HL(index).A,
HL(index).R, HL(index).G, HL(index).B))

bm.SetPixel(x,          index          +          midpoint,
Drawing.Color.FromArgb(LL(index).A,             LL(index).R,
LL(index).G, LL(index).B))
                Next
            Next
            PictureBox1.Image = bm
MsgBox("Image has been DWT Transformed")

        End If
    End Sub
```

```
    Private Sub Button5_Click(sender As Object, e As
EventArgs) Handles Button5.Click
        Error1.Text = ""
        If UsernameTxt.Text.ToLower = "admin" And
PasswordTxt.Text = "password" Then
vanish(Panel1)
unvanish(Panel2)
        Else
            Error1.Text = "Invalid Username or Password"
        End If
    End Sub

    Private Sub Button6_Click(sender As Object, e As
EventArgs) Handles Button6.Click
        'Show Cryptography
vanish(Panel2)
unvanish(Panel3)
    End Sub

    Private Sub Button7_Click(sender As Object, e As
EventArgs) Handles Button7.Click
        'Show Steganography
vanish(Panel2)
unvanish(Panel4)
    End Sub

    Private Sub Button8_Click(sender As Object, e As
EventArgs) Handles Button8.Click
        'Hide Cryptography Panel (Panel 3)
vanish(Panel3)
unvanish(Panel2)
    End Sub

    Private Sub Button9_Click(sender As Object, e As
EventArgs) Handles Button9.Click
        'Hide Message in Image using Steganography
```

```
        If RichTextBox3.Text <> "" And Not PictureBox1.Image
Is Nothing Then
            Dim bm As Bitmap = PictureBox1.Image.Clone()
            Dim         b         As         Byte()         =
System.Text.ASCIIEncoding.ASCII.GetBytes(RichTextBox3.Text)
            Dim count As Integer = 0
            'Button4_Click(Button4, New EventArgs())
            If bm.Width * bm.Height / 2 <b.Length Then
MsgBox("Message too large to be embedded in this Image")
                Exit Sub
            End If
            For x = 0 Tobm.Width - 1
                For y = 1 Tobm.Height - 2 Step 2
                    Dim c1 As New Color(bm.GetPixel(x, y))
                    Dim  c2  As  New  Color(bm.GetPixel(x,  y  +
1))
insertData(c1, c2, b(count))
bm.SetPixel(x,  y,  Drawing.Color.FromArgb(c1.A,  c1.R,  c1.G,
c1.B))
bm.SetPixel(x,  y  +  1,  Drawing.Color.FromArgb(c2.A,  c2.R,
c2.G, c2.B))
count += 1
                    If count = b.Length Then
                        PictureBox1.Image = bm
MsgBox("Message hidden ... You may now save image")
                        Exit Sub
                    End If
                Next
            Next
        End If

    End Sub

    Private  Sub  Button10_Click(sender  As  Object,  e  As
EventArgs) Handles Button10.Click
        'hide steganography panel (panel 4)
```

```
vanish(Panel4)
unvanish(Panel2)
    End Sub

    Private Sub Form1_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
unvanish(Panel1)
    End Sub

    Private Sub Button11_Click(sender As Object, e As
EventArgs) Handles Button11.Click
        Dim p As New PasswordDialog()
        Dim passwd As String = p.ShowDialog()
Encryption.pass = passwd
        If passwd<> "" And RichTextBox2.Text <> "" Then
            RichTextBox4.Text                                =
Encryption.DecryptString(RichTextBox2.Text)
        End If
    End Sub
End Class
```

PasswordDialog.CS

```
Public Class PasswordDialog

    Private Sub PasswordDialog_Load(sender As Object, e As
EventArgs) Handles MyBase.Load

    End Sub

    Overloads Function ShowDialog() As String
MyBase.ShowDialog()
        Return TextBox1.Text
    End Function
```

```
    Private Sub TextBox1_Keyup(sender As Object, e As
KeyEventArgs) Handles TextBox1.KeyUp
        If e.KeyCode = Keys.EnterAnd TextBox1.Text <> "" Then
Me.DialogResult = Windows.Forms.DialogResult.OK
        End If
    End Sub
End Class
```

Module1.CS

```
Module Module1
    Public Sub unvanish(ByVal c As Control)
c.Location = New Point(0, c.Location.Y)
    End Sub

    Public Sub vanish(ByVal c As Control)
c.Location = New Point(-10000, c.Location.Y)
    End Sub

    Function toJSON(obj As Object) As String
        Return
Newtonsoft.Json.JsonConvert.SerializeObject(obj)
    End Function

    Function toBinary(num As Integer) As String
        Dim s As String = ""
        While num> 0
            s = (num Mod 2) & s
num = num / 2
        End While
        s = s.PadLeft(8, "0")
        Return s
    End Function

    Function toInteger(b As String) As Integer
```

```
        Dim l As Integer = b.Length
        Dim temp As Integer = 0
        For x = 1 To l
temp = temp + CInt(Mid(b, l - x + 1, 1)) * 2 ^ (x - 1)
        Next
        Return temp
    End Function

    Sub insertData(ByRef c1 As Color, ByRef c2 As Color, data
As Byte)
        Dim t As Integer = 0
        Dim N As Byte = data
        Dim a As New Color()
a.R = N Mod 2
        t = a.R
a.G = ((N - t) / 2) Mod 2
        t = (t + 2) * a.G
a.B = ((N - t) / 4) Mod 2
        t = (t + 4) * a.B
a.A = ((N - t) / 8) Mod 2
        t = (t + 8) * a.A
        Dim b As New Color()
b.R = ((N - t) / 16) Mod 2
        t = (t + 16) * b.R
b.G = ((N - t) / 32) Mod 2
        t = (t + 32) * b.G
b.B = ((N - t) / 64) Mod 2
        t = (t + 64) * b.B
b.A = ((N - t) / 128) Mod 2
        t = (t + 128) * b.A
        c1.R = c1.R - (c1.R Mod 2) + a.R
        c1.G = c1.G - (c1.G Mod 2) + a.G
        c1.B = c1.B - (c1.B Mod 2) + a.B
        c1.A = c1.A - (c1.A Mod 2) + a.A
        c2.R = c2.R - (c2.R Mod 2) + b.R
        c2.G = c2.G - (c2.G Mod 2) + b.G
```

```
        c2.B = c2.B - (c2.B Mod 2) + b.B
        c2.A = c2.A - (c2.A Mod 2) + b.A
    End Sub

    Sub normaliseColor(ByRef c1_ As Color)
        If c1_.R > 255 Then
            c1_.R = 255
        End If
        If c1_.G > 255 Then
            c1_.G = 255
        End If
        If c1_.B > 255 Then
            c1_.B = 255
        End If
        If c1_.A > 255 Then
            c1_.A = 255
        End If
        If c1_.R < 0 Then
            c1_.R = 0
        End If
        If c1_.G < 0 Then
            c1_.G = 0
        End If
        If c1_.B < 0 Then
            c1_.B = 0
        End If
        If c1_.A < 0 Then
            c1_.A = 0
        End If
    End Sub

    Public Function ScaleImage(source As
System.Drawing.Bitmap, x As Integer, y As Integer) As
System.Drawing.Bitmap
        Try
            Dim scale As Single = x / source.Width
```

```
            Dim            mypng            As            New
System.Drawing.Bitmap(CInt(source.Width      *      scale),
CInt(source.Height * scale), source.PixelFormat)
            Dim    gr    As    System.Drawing.Graphics    =
System.Drawing.Graphics.FromImage(mypng)
gr.DrawImage(source, 0, 0, mypng.Width + 1, mypng.Height + 1)
            Return mypng
        Catch ex As Exception
MsgBox("Error in Scaling Image")
            Return New System.Drawing.Bitmap(300, 300)
        End Try
    End Function

    Function  toBytes(ByVal  b  As  System.Drawing.Bitmap)  As
Byte()
        Dim s AsSystem.IO.Stream = New System.IO.MemoryStream

b.Save(s, System.Drawing.Imaging.ImageFormat.png)
s.Position = 0
        Dim bb(s.Length) As Byte
s.Read(bb, 0, s.Length)
        Return bb
    End Function

    Function    toBitmap(ByVal    b    As    Byte())    As
System.Drawing.Bitmap
        Dim s AsSystem.IO.Stream = New System.IO.MemoryStream
s.Write(b, 0, b.Length)
        Dim bt As New System.Drawing.Bitmap(s)
        Return bt
    End Function
End Module
```

Encryption.CS

```
Imports System.Security.Cryptography
Public Class Encryption
```

```
    Public Shared pass As String = "x12345678y"

    Public Shared Function EncryptBytes(ByVal b() As Byte) As
Byte()
        Dim              AES              As              New
System.Security.Cryptography.RijndaelManaged
        Dim                  Hash_AESAs                  New
System.Security.Cryptography.MD5CryptoServiceProvider
        Dim encrypted() As Byte = Nothing
        Try
            Dim hash(31) As Byte
            Dim        temp        As        Byte()        =
Hash_AES.ComputeHash(System.Text.ASCIIEncoding.ASCII.GetByt
es(pass))
Array.Copy(temp, 0, hash, 0, 16)
Array.Copy(temp, 0, hash, 15, 16)
AES.Key = hash
AES.Mode = CipherMode.ECB
            Dim
DESEncrypterAsSystem.Security.Cryptography.ICryptoTransform
= AES.CreateEncryptor
            Dim Buffer As Byte() = b
encrypted   =   DESEncrypter.TransformFinalBlock(Buffer,   0,
Buffer.Length)
            Return encrypted
        Catch ex As Exception
            Return Nothing
        End Try
    End Function

    Public Shared Function DecryptBytes(ByVal b() As Byte) As
Byte()
        Dim              AES              As              New
System.Security.Cryptography.RijndaelManaged
        Dim                  Hash_AESAs                  New
System.Security.Cryptography.MD5CryptoServiceProvider
```

```
        Dim decrypted() As Byte = Nothing
        Try
            Dim hash(31) As Byte
            Dim temp As Byte() =
Hash_AES.ComputeHash(System.Text.ASCIIEncoding.ASCII.GetByt
es(pass))
Array.Copy(temp, 0, hash, 0, 16)
Array.Copy(temp, 0, hash, 15, 16)
AES.Key = hash
AES.Mode = CipherMode.ECB
            Dim
DESDecrypterAsSystem.Security.Cryptography.ICryptoTransform
= AES.CreateDecryptor
            Dim Buffer As Byte() = b
decrypted = DESDecrypter.TransformFinalBlock(Buffer, 0,
Buffer.Length)
            Return decrypted
        Catch ex As Exception
            Return Nothing
        End Try
    End Function

    Shared Function EncryptString(input As String) As String
        Return
Convert.ToBase64String(EncryptBytes(System.Text.ASCIIEncodi
ng.ASCII.GetBytes(input)))
    End Function

    Shared Function DecryptString(input As String) As String
        Return
System.Text.Encoding.UTF8.GetString(DecryptBytes(Convert.Fr
omBase64String(input)))
    End Function

    Shared Function Compress(ByVal raw() As Byte) As Byte()
' Clean up memory with Using-statements.
```

```
        Using memory As System.IO.MemoryStream = New
System.IO.MemoryStream()
' Create compression stream.
            Using gzip As System.IO.Compression.GZipStream =
New                 System.IO.Compression.GZipStream(memory,
System.IO.Compression.CompressionMode.Compress, True)
' Write.
gzip.Write(raw, 0, raw.Length)
            End Using
' Return array.
            Return memory.ToArray()
        End Using
    End Function

    Shared Function DeCompress(ByVal raw() As Byte) As Byte()
' Clean up memory with Using-statements.
        Using ms As New IO.MemoryStream(raw)
            Using           gzs           As           New
System.IO.Compression.GZipStream(ms,
IO.Compression.CompressionMode.Decompress)

                Using rdr As New IO.StreamReader(gzs)
                    Dim s As String = rdr.ReadToEnd
                    'Return Convert.FromBase64String(s)
                    Return
System.Text.Encoding.ASCII.GetBytes(s)
                End Using 'rdr
            End Using 'gzs
        End Using 'ms
        Return Nothing
    End Function
End Class
```

Color.CS

```
Public Class Color
    Property R As Integer
```

```
    Property G As Integer
    Property B As Integer
    Property A As Integer

    Sub New()

    End Sub

    Sub New(c AsDrawing.Color)
        R = c.R
        G = c.G
        B = c.B
        A = c.A
    End Sub

    Sub New(R, G, B, A)
Me.R = R
Me.G = G
Me.B = B
Me.A = A
    End Sub

    Function clone() As Color
        Return Me.MemberwiseClone()
    End Function
EndClass
```

STEGANOGRAPHY CODE ANALYSIS

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;

namespace Text2Image
{
    public partial class FrmSteganography : Form
```

```
    {
        public FrmSteganography()
        {
            InitializeComponent();
        }

        //public values:
        string loadedTrueImagePath, loadedFilePath,
saveToImage,DLoadImagePath,DSaveFilePath;
        int height, width;
        long fileSize, fileNameSize;
        Image loadedTrueImage, DecryptedImage
,AfterEncryption;
        Bitmap loadedTrueBitmap, DecryptedBitmap;
        Rectangle previewImage = new
Rectangle(20,160,490,470);
        bool canPaint = false, EncriptionDone = false;
        byte[] fileContainer;

        private void EnImageBrowse_btn_Click(object sender,
EventArgs e)
        {
            if (openFileDialog1.ShowDialog() ==
DialogResult.OK)
            {
                loadedTrueImagePath =
openFileDialog1.FileName;
                EnImage_tbx.Text = loadedTrueImagePath;
                loadedTrueImage =
Image.FromFile(loadedTrueImagePath);
                height = loadedTrueImage.Height;
                width = loadedTrueImage.Width;
                loadedTrueBitmap = new
Bitmap(loadedTrueImage);
```

```
                FileInfo imginf = new
FileInfo(loadedTrueImagePath);
                float fs = (float)imginf.Length / 1024;
                ImageSize_lbl.Text =
smalldecimal(fs.ToString(), 2) + " KB";
                ImageHeight_lbl.Text =
loadedTrueImage.Height.ToString() + " Pixel";
                ImageWidth_lbl.Text =
loadedTrueImage.Width.ToString() + " Pixel";
                double cansave = (8.0 * ((height * (width /
3) * 3) / 3 - 1)) / 1024;
                CanSave_lbl.Text =
smalldecimal(cansave.ToString(), 2) + " KB";

                canPaint = true;
                this.Invalidate();
            }
        }

        private string smalldecimal(string inp, int dec)
        {
            int i;
            for (i = inp.Length - 1; i > 0; i--)
                if (inp[i] == '.')
                    break;
            try
            {
                return inp.Substring(0, i + dec + 1);
            }
            catch
            {
                return inp;
            }
        }
```

```
        private void EnFileBrowse_btn_Click(object sender,
EventArgs e)
        {
            if (openFileDialog2.ShowDialog() ==
DialogResult.OK)
            {
                loadedFilePath = openFileDialog2.FileName;
                EnFile_tbx.Text = loadedFilePath;
                FileInfo finfo = new
FileInfo(loadedFilePath);
                fileSize = finfo.Length;
                fileNameSize =
justFName(loadedFilePath).Length;
            }
        }

        private void Encrypt_btn_Click(object sender,
EventArgs e)
        {
            if (saveFileDialog1.ShowDialog() ==
DialogResult.OK)
            {
                saveToImage = saveFileDialog1.FileName;
            }
            else
                return;
            if (EnImage_tbx.Text == String.Empty ||
EnFile_tbx.Text == String.Empty)
            {
                MessageBox.Show("Encrypton information is
incomplete!\nPlease complete them frist.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            if (8*((height * (width/3)*3)/3 - 1) < fileSize
+ fileNameSize)
            {
```

```
                MessageBox.Show("File size is too
large!\nPlease use a larger image to hide this file.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            fileContainer =
File.ReadAllBytes(loadedFilePath);
            EncryptLayer();
        }


        private void EncryptLayer()
        {
            toolStripStatusLabel1.Text ="Encrypting...
Please wait";
            Application.DoEvents();
            long FSize = fileSize;
            Bitmap changedBitmap = EncryptLayer(8,
loadedTrueBitmap, 0, (height * (width/3)*3) / 3 -
fileNameSize - 1, true);
            FSize -= (height * (width / 3) * 3) / 3 -
fileNameSize - 1;
            if(FSize > 0)
            {
                for (int i = 7; i >= 0 && FSize > 0; i--)
                {
                    changedBitmap = EncryptLayer(i,
changedBitmap, (((8 - i) * height * (width / 3) * 3) / 3 -
fileNameSize - (8 - i)), (((9 - i) * height * (width / 3) *
3) / 3 - fileNameSize - (9 - i)), false);
                    FSize -= (height * (width / 3) * 3) / 3
- 1;
                }
            }
            changedBitmap.Save(saveToImage);
```

```
            toolStripStatusLabel1.Text = "Encrypted image
has been successfully saved.";
            EncriptionDone = true;
            AfterEncryption = Image.FromFile(saveToImage);
            this.Invalidate();
        }


        private Bitmap EncryptLayer(int layer, Bitmap
inputBitmap, long startPosition, long endPosition, bool
writeFileName)
        {
            Bitmap outputBitmap = inputBitmap;
            layer--;
            int i = 0, j = 0;
            long FNSize = 0;
            bool[] t = new bool[8];
            bool[] rb = new bool[8];
            bool[] gb = new bool[8];
            bool[] bb = new bool[8];
            Color pixel = new Color();
            byte r, g, b;

            if (writeFileName)
            {
                FNSize = fileNameSize;
                string fileName =
justFName(loadedFilePath);

                //write fileName:
                for (i = 0; i < height && i * (height / 3)
< fileNameSize; i++)
                    for (j = 0; j < (width / 3) * 3 && i *
(height / 3) + (j / 3) < fileNameSize; j++)
                    {
```

```
                        byte2bool((byte)fileName[i *
(height / 3) + j / 3], ref t);
                        pixel = inputBitmap.GetPixel(j, i);
                        r = pixel.R;
                        g = pixel.G;
                        b = pixel.B;
                        byte2bool(r, ref rb);
                        byte2bool(g, ref gb);
                        byte2bool(b, ref bb);
                        if (j % 3 == 0)
                        {
                            rb[7] = t[0];
                            gb[7] = t[1];
                            bb[7] = t[2];
                        }
                        else if (j % 3 == 1)
                        {
                            rb[7] = t[3];
                            gb[7] = t[4];
                            bb[7] = t[5];
                        }
                        else
                        {
                            rb[7] = t[6];
                            gb[7] = t[7];
                        }
                        Color result =
Color.FromArgb((int)bool2byte(rb), (int)bool2byte(gb),
(int)bool2byte(bb));
                        outputBitmap.SetPixel(j, i,
result);
                    }
                i--;
            }
            //write file (after file name):
            int tempj = j;
```

```
            for (; i < height && i * (height / 3) <
endPosition - startPosition + FNSize && startPosition + i *
(height / 3) < fileSize + FNSize; i++)
                for (j = 0; j < (width / 3) * 3 && i *
(height / 3) + (j / 3) < endPosition - startPosition +
FNSize && startPosition + i * (height / 3) + (j / 3) <
fileSize + FNSize; j++)
                {
                    if (tempj != 0)
                    {
                        j = tempj;
                        tempj = 0;
                    }

byte2bool((byte)fileContainer[startPosition + i * (height /
3) + j / 3 - FNSize], ref t);
                    pixel = inputBitmap.GetPixel(j, i);
                    r = pixel.R;
                    g = pixel.G;
                    b = pixel.B;
                    byte2bool(r, ref rb);
                    byte2bool(g, ref gb);
                    byte2bool(b, ref bb);
                    if (j % 3 == 0)
                    {
                        rb[layer] = t[0];
                        gb[layer] = t[1];
                        bb[layer] = t[2];
                    }
                    else if (j % 3 == 1)
                    {
                        rb[layer] = t[3];
                        gb[layer] = t[4];
                        bb[layer] = t[5];
                    }
```

```
                    else
                    {
                        rb[layer] = t[6];
                        gb[layer] = t[7];
                    }
                    Color result =
Color.FromArgb((int)bool2byte(rb), (int)bool2byte(gb),
(int)bool2byte(bb));
                    outputBitmap.SetPixel(j, i, result);

                }
            long tempFS = fileSize, tempFNS = fileNameSize;
            r = (byte)(tempFS % 100);
            tempFS /= 100;
            g = (byte)(tempFS % 100);
            tempFS /= 100;
            b = (byte)(tempFS % 100);
            Color flenColor = Color.FromArgb(r,g,b);
            outputBitmap.SetPixel(width - 1, height - 1,
flenColor);

            r = (byte)(tempFNS % 100);
            tempFNS /= 100;
            g = (byte)(tempFNS % 100);
            tempFNS /= 100;
            b = (byte)(tempFNS % 100);
            Color fnlenColor = Color.FromArgb(r,g,b);
            outputBitmap.SetPixel(width - 2, height - 1,
fnlenColor);

            return outputBitmap;
        }


        private void DecryptLayer()
        {
```

```
            toolStripStatusLabel1.Text = "Decrypting...
Please wait";
            Application.DoEvents();
            int i, j = 0;
            bool[] t = new bool[8];
            bool[] rb = new bool[8];
            bool[] gb = new bool[8];
            bool[] bb = new bool[8];
            Color pixel = new Color();
            byte r, g, b;
            pixel = DecryptedBitmap.GetPixel(width - 1,
height - 1);
            long fSize = pixel.R + pixel.G * 100 + pixel.B
* 10000;
            pixel = DecryptedBitmap.GetPixel(width - 2,
height - 1);
            long fNameSize = pixel.R + pixel.G * 100 +
pixel.B * 10000;
            byte[] res = new byte[fSize];
            string resFName = "";
            byte temp;

            //Read file name:
            for (i = 0; i < height && i * (height / 3) <
fNameSize; i++)
                for (j = 0; j < (width / 3) * 3 && i *
(height / 3) + (j / 3) < fNameSize; j++)
                {
                    pixel = DecryptedBitmap.GetPixel(j, i);
                    r = pixel.R;
                    g = pixel.G;
                    b = pixel.B;
                    byte2bool(r, ref rb);
                    byte2bool(g, ref gb);
                    byte2bool(b, ref bb);
                    if (j % 3 == 0)
```

```
                    {
                        t[0] = rb[7];
                        t[1] = gb[7];
                        t[2] = bb[7];
                    }
                    else if (j % 3 == 1)
                    {
                        t[3] = rb[7];
                        t[4] = gb[7];
                        t[5] = bb[7];
                    }
                    else
                    {
                        t[6] = rb[7];
                        t[7] = gb[7];
                        temp = bool2byte(t);
                        resFName += (char)temp;
                    }
                }

            //Read file on layer 8 (after file name):
            int tempj = j;
            i--;

            for (; i < height && i * (height / 3) < fSize +
fNameSize; i++)
                for (j = 0; j < (width / 3) * 3 && i *
(height / 3) + (j / 3) < (height * (width / 3) * 3) / 3 - 1
&& i * (height / 3) + (j / 3) < fSize + fNameSize; j++)
                {
                    if (tempj != 0)
                    {
                        j = tempj;
                        tempj = 0;
                    }
                    pixel = DecryptedBitmap.GetPixel(j, i);
```

```
                    r = pixel.R;
                    g = pixel.G;
                    b = pixel.B;
                    byte2bool(r, ref rb);
                    byte2bool(g, ref gb);
                    byte2bool(b, ref bb);
                    if (j % 3 == 0)
                    {
                        t[0] = rb[7];
                        t[1] = gb[7];
                        t[2] = bb[7];
                    }
                    else if (j % 3 == 1)
                    {
                        t[3] = rb[7];
                        t[4] = gb[7];
                        t[5] = bb[7];
                    }
                    else
                    {
                        t[6] = rb[7];
                        t[7] = gb[7];
                        temp = bool2byte(t);
                        res[i * (height / 3) + j / 3 -
fNameSize] = temp;
                    }
                }

            //Read file on other layers:
            long readedOnL8 = (height * (width/3)*3) /3 -
fNameSize - 1;

            for (int layer = 6; layer >= 0 && readedOnL8 +
(6 - layer) * ((height * (width / 3) * 3) / 3 - 1) < fSize;
layer--)
```

```
                for (i = 0; i < height && i * (height / 3)
+ readedOnL8 + (6 - layer) * ((height * (width / 3) * 3) /
3 - 1) < fSize; i++)
                    for (j = 0; j < (width / 3) * 3 && i *
(height / 3) + (j / 3) + readedOnL8 + (6 - layer) *
((height * (width / 3) * 3) / 3 - 1) < fSize; j++)
                    {
                        pixel = DecryptedBitmap.GetPixel(j,
i);
                        r = pixel.R;
                        g = pixel.G;
                        b = pixel.B;
                        byte2bool(r, ref rb);
                        byte2bool(g, ref gb);
                        byte2bool(b, ref bb);
                        if (j % 3 == 0)
                        {
                            t[0] = rb[layer];
                            t[1] = gb[layer];
                            t[2] = bb[layer];
                        }
                        else if (j % 3 == 1)
                        {
                            t[3] = rb[layer];
                            t[4] = gb[layer];
                            t[5] = bb[layer];
                        }
                        else
                        {
                            t[6] = rb[layer];
                            t[7] = gb[layer];
                            temp = bool2byte(t);
                            res[i * (height / 3) + j / 3 +
(6 - layer) * ((height * (width / 3) * 3) / 3 - 1) +
readedOnL8] = temp;
                        }
```

```
                    }

            if (File.Exists(DSaveFilePath + "\\" +
resFName))
            {
                MessageBox.Show("File \"" + resFName + "\"
already exist please choose another path to save file",
"Error",MessageBoxButtons.OK,MessageBoxIcon.Error);
                return;
            }
            else
                File.WriteAllBytes(DSaveFilePath + "\\" +
resFName, res);
            toolStripStatusLabel1.Text = "Decrypted file
has been successfully saved.";
            Application.DoEvents();
        }

        private void byte2bool(byte inp, ref bool[] outp)
        {
            if(inp>=0 && inp<=255)
                for (short i = 7; i >= 0; i--)
                {
                    if (inp % 2 == 1)
                        outp[i] = true;
                    else
                        outp[i] = false;
                    inp /= 2;
                }
            else
                throw new Exception("Input number is
illegal.");
        }

        private byte bool2byte(bool[] inp)
        {
```

```
            byte outp = 0;
            for (short i = 7; i >= 0; i--)
            {
                if (inp[i])
                    outp += (byte)Math.Pow(2.0, (double)(7-
i));
            }
            return outp;
        }

        private void Decrypt_btn_Click(object sender,
EventArgs e)
        {


            if (DeSaveFile_tbx.Text == String.Empty ||
DeLoadImage_tbx.Text == String.Empty)
            {
                MessageBox.Show("Text boxes must not be
empty!", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);

                return;
            }

            if (System.IO.File.Exists(DeLoadImage_tbx.Text)
== false)
            {
                MessageBox.Show("Select image file.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                DeLoadImage_tbx.Focus();
                return;
            }
```

```
            DecryptLayer();
        }

        private void DeLoadImageBrowse_btn_Click(object
sender, EventArgs e)
        {
            if (openFileDialog3.ShowDialog() ==
DialogResult.OK)
            {
                DLoadImagePath = openFileDialog3.FileName;
                DeLoadImage_tbx.Text = DLoadImagePath;
                DecryptedImage =
Image.FromFile(DLoadImagePath);
                height = DecryptedImage.Height;
                width = DecryptedImage.Width;
                DecryptedBitmap = new
Bitmap(DecryptedImage);

                FileInfo imginf = new
FileInfo(DLoadImagePath);
                float fs = (float)imginf.Length / 1024;
                ImageSize_lbl.Text =
smalldecimal(fs.ToString(), 2) + " KB";
                ImageHeight_lbl.Text =
DecryptedImage.Height.ToString() + " Pixel";
                ImageWidth_lbl.Text =
DecryptedImage.Width.ToString() + " Pixel";
                double cansave = (8.0 * ((height * (width /
3) * 3) / 3 - 1)) / 1024;
                CanSave_lbl.Text =
smalldecimal(cansave.ToString(), 2) + " KB";

                canPaint = true;
                this.Invalidate();
            }
        }
```

```
        private void DeSaveFileBrowse_btn_Click(object
sender, EventArgs e)
        {
            if (folderBrowserDialog1.ShowDialog() ==
DialogResult.OK)
            {
                DSaveFilePath =
folderBrowserDialog1.SelectedPath;
                DeSaveFile_tbx.Text = DSaveFilePath;
            }
        }

        private void Form1_Paint(object sender,
PaintEventArgs e)
        {
            if(canPaint)
                try
                {
                    if (!EncriptionDone)

e.Graphics.DrawImage(loadedTrueImage, previewImage);
                    else

e.Graphics.DrawImage(AfterEncryption, previewImage);
                }
                catch
                {
                    e.Graphics.DrawImage(DecryptedImage,
previewImage);
                }
        }

        private string justFName(string path)
        {
            string output;
```

```
            int i;
            if (path.Length == 3)   // i.e: "C:\\"
                return path.Substring(0, 1);
            for (i = path.Length - 1; i > 0; i--)
                if (path[i] == '\\')
                    break;
            output = path.Substring(i + 1);
            return output;
        }


        private string justEx(string fName)
        {
            string output;
            int i;
            for (i = fName.Length - 1; i > 0; i--)
                if (fName[i] == '.')
                    break;
            output = fName.Substring(i + 1);
            return output;
        }

        private void Close_btn_Click(object sender,
EventArgs e)
        {
            this.Close();
        }

        private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {

System.Diagnostics.Process.Start("http:\\\\www.programmer2p
rogrammer.net");
     }
          }
                }
```