IMPLEMENTATION OF TEXT STEGANOGRAPHY USING SYNONYM SUBSTITUTION BASED ALGORITHM IN LETTER OF CREDIT (LC)

ME ME KHAING

M.C.Sc. (THESIS)

AUGUST, 2016

IMPLEMENTATION OF TEXT STEGANOGRAPHY USING SYNONYM SUBSTITUTION BASED ALGORITHM IN LETTER OF CREDIT (LC)

ME ME KHAING

M.C.Sc. (THESIS)

August, 2016

IMPLEMENTATION OF TEXT STEGANOGRAPHY USING SYNONYM SUBSTITUTION BASED ALGORITHM IN LETTER OF CREDIT (LC)

BY

ME ME KHAING

B.C.Sc. (Hons:)

Dissertation submitted in partial fulfillment of the requirements

for the degree of

Master of Computer Science (M.C.Sc.)

of

COMPUTER UNIVERSITY (MANDALAY)

August, 2016

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude and sincere thanks to all persons who contributed directly or indirectly towards the success of this thesis.

I owe my respectful thanks to **Dr. Win Aye**, the Rector, Computer University (Mandalay), for her kind permission to make this thesis in this university.

I am deeply thankful to **Dr. Thi Thi Soe**, Associate Professor, Department of Software Technology, Computer University (Mandalay), for her continuous, sinscere, and kind guidance during the period of completing this thesis.

I am greatly indebated to my supervisor, **Dr. Aye Aye Chaw**, Associate Professor, Head of Department of Software Technology, Computer University (Mandalay), for her valuable supervision, good advice, detailed guidance, and helpful suggestions throughout the terms of finishing the thesis.

I would like to extend my deepest gratitude to **U Thaung Kyaw**, Associate Professor, Head of Department of English at Computer University (Mandalay), for editing my thesis.

Finally, I also thank all the members of Computer University (Mandaly), who attended the seminars on this thesis, for their support, valuable suggestions, helpful hints, and fair criticisms.

Moreover, I also thank all my mentors and friends, for their invaluable suggestions and helpful contributions to success this thesis.

Especially, my deepest thanks go to my parents for giving me moral and material support, sympathy and empathy, care and kindness throughout my studies and my efforts to complete this thesis without any trouble.

ABSTRACT

Text steganography uses text files as carrier files to hide confidential information. The goal of text steganography is to make the hidden data undetectable except the sender and receiver. This thesis focuses on semantic based text steganography. It hides confidential information by replacing words with their synonyms (same meanings). The main advantage of semantic based text steganography is that the encrypted information is in human readable text format. So, the encrypted information cannot be suspected by the other parties who are not related with the information. One of the major transformations used in semantic based text steganography is synonym substitution that hides the confidential information by substituting the words of cover text with the words that are in same meaning. This system hides the confidential information by applying Synonym Substitution-Based Algorithm. There are three sub-algorithms in Synonym Substitution-Based Algorithm: Synonym Retrieval Algorithm, Obfuscation Algorithm and Debfuscation Algorithm. The Synonym Retrieval Algorithm retrieves synonyms (same meaning of cover words) from WordNet Dictionary and it uses British National Corpus (BNC) frequency data. The Obfuscation Algorithm encodes the message and it uses American National Corpus (ANC) frequency data. The Debfuscation Algorithm decrypts the message. These days, more and more banks focus on the security of the customer's confidential information such as main balances and account numbers. This thesis implements Synonym Substitution-Based Algorithm to make bank's Letter of Credit (LC) information (balances and account numbers) more secure. This system is implemented by using Java software development kit (J2SE 7).

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

CHAPTER 1

INTRODUCTION

Steganography, the art of information hiding, has been around for thousands of years. Steganography is concerned with hiding the existence of data by encapsulating it within some cover-text. The goal is to make the hidden data undetectable, by man or machine [3].

Nowadays, more and more banks are connecting not only locally but also globally. Likewise, more and more trading companies sell and buy products internationally. In international trading processes, products are imported and exported between foreign countries, in this case, the sellers and the buyers do not know each other personally. Therefore, the seller has a number of risks such as credit risk and legal risk caused by the distance. Letter of credit (LC) in International Banking process helps to solve these risks.

Letter of Credit (LC) is a letter from a bank guaranteeing that a buyer's payment to a seller will be received on time and for the correct amount [9]. To use LC, both the seller and the buyer of the product open bank account and deposit amounts at Banks in their countries. The two banks from seller's country and buyer's country are connected each other. Then, the bank from buyer's country guarantees that the buyer has sufficient amount to buy the product. This system makes safe the LC information between the bank from buyer's country called issuing bank and the bank from seller's country called advising bank by using the Synonym Substitution Based Algorithm. This system includes two banks the issuing bank which is the sender of the message and the advising bank which is the receiver of the message. The sender encodes the message by applying Synonym Retrieval Algorithm and Obfuscation Algorithm. The receiver decodes the message by applying Deobfuscation Algorithm.

1.1 Motivation of the Thesis

This system implements Synonym Substitution Based Algorithm to make LC information safe. Text file, which is the most popular and useful media, is used as carrier file to hide information in this system. The sending side needs the information to hide and the carrier text file to encode the information. The receiving side needs the stego text file and synset file that is received from the sending side, to decode the original information.

1.2 Objectives of the Thesis

The objectives of the system are:

- To understand the details of text steganography
- To apply synonym substitution-based algorithm for text steganography
- To hide the sensitive information without arising the suspicion by human or machine
- To secure the LC information between the issuing bank and advising bank by using synonym substitution-based algorithm

1.3 Organization of the Thesis

This thesis is organized in five chapters.

Chapter 1 introduces text steganography. This chapter also mentions objectives of the thesis. This chapter concludes by describing the organization of thesis.

Chapter 2 describes the theory about the steganography, history of steganography, uses of steganography, types of steganography.

Chapter 3 discusses linguistic-based text steganography, ASCII, Synonym Retrieval Algorithm, WordNet Dictionary, British National Corpus,

American National Corpus, Obfuscation Algorithm and Deobfuscation Algorithm.

Chapter 4 explains the design, implementation and result of the system.

Chapter 5 points out conclusion, limitations and further extensions of the system.

CHAPTER 2

THEORETICAL BACKGROUND

This chapter explains the theoretical background of Steganography. It also describes the history of steganography, uses of steganography, types of steganography and the detailed theory of text steganography.

2.1 Steganography

Steganography is the art and science of hiding information technique such that its presence cannot be detected and a communication is happening. Secret information is encoding in a manner such that the very existence of the information is concealed. Paired with existing communication methods, steganography can be used to carry out hidden exchanges.

2.2 History of Steganography

The word steganography combines the Greek words *steganos* which mean "covered, concealed, or protected", and *graphein* which means "writing". The term "Steganography" comes from word "Steganographia" which is the title of a book written by the German abbot, whose name is Johannes Trithemius. There are three series in that book and contained hidden information on cryptography and steganography [11].

Another modern example is from the Second World War. Germans developed the Microdots (small round images around 1mm in diameter), which would then be hidden. Information, especially photographs, was reduced in size until it was the size of a typed period. Extremely difficult to detect, to the naked eye, the information is invisible, and can only be viewed using magnification [3].

2.3 Uses of Steganography

The historical primary use of steganography is for message hiding. When methods such as cryptography are used, the output of these methods, such as the cipher text, is often very noticeable. The reason for this is that these methods are designed to ensure that the message is unreadable to anyone other than the intended receiver, rather than hide its existence.

Steganographic techniques, however, are designed to ensure that the message (or data) is hidden and so observers do not know that the message exists. The hidden message or data may be encrypted before it is hidden for an added layer of security. The reason why this is useful is that in some countries it is illegal to even possess encrypted material. By using steganography this material can be hidden. This process is however vulnerable.

Steganography is increasingly being found to be used by terrorists and criminals to hide data and exchange information. For example, a child pornography distributor could hide images in the photos on a legitimate ebay listing to distribute them to their customers. There has also been evidence of al-Qaeda storing information contained in text files hidden in videos.

Steganography is not just used by criminals and organizations. There are a large number of freely available steganography programs which can be used by home users to store sensitive information more securely, for example bank account details could be hidden in an image so when they are required they can be retrieved, but if someone was to gain access to the computer they would not be able to identify them (as they would be able to recognize, and possibly break, an encrypted file).

There is evidence of governments using steganography for both legal and illegal communication. There has been court cases in the US where accused Russian spies were found to have been using steganographic communication channels (specifically images) to communicate with their handlers.

There have even been proposals of using steganography as a means for communication within a botnet. Nagaraja proposed Stegobot, a botnet which uses steganography as the basis for its command and control (C& C) network. It uses JPEG steganography to hide collected data (such as credit card details and passwords) in images that are uploaded to Facebook. Bots are connected to each other via users on the social network. Once the images are uploaded, they are visible to all users connected to the uploading user. Bots on infected machines intercept any images that a user on that machine views, and extracts any information. This bot then hides the data in an image upload by a user on its computer, where it will be visible to all of that user's connections. The data eventually reaches the bot master via restrictive flooding. The estimated bitrate is around 20mb/month, which while not sounding like much can represent a large amount of personal information [4].

One particularly interesting case is that printer manufacturers secretly printing microscopic dots onto all pages they print. These dots are arranged in a pattern which can be decoded into the date and time the page was printed, along with the printer's serial number. It is claimed that this data is used by governments in criminal counterfeit investigations. This is an example of steganography where the data has been used to generate a pattern, and then this pattern has been printed at a microscopic level, following a similar (but much smaller) principle to microdots. There are a number of pieces of software that can be used to perform steganography such as OpenStego. It is an open source program for performing image steganography. It can hide any type of file within the image, and files are encrypted before they are hidden [10].

2.4 Types of Steganography

In the modern age, the important confidential information are mainly sent and received over the internet. To make our information more secure, modern steganography can transmit and receive our information by embedding four main different types of media: Text, Image, Audio and Video. Steganography uses these four types of media. Therefore, there are four types of steganography: Image, Audio, Video and Text [3].

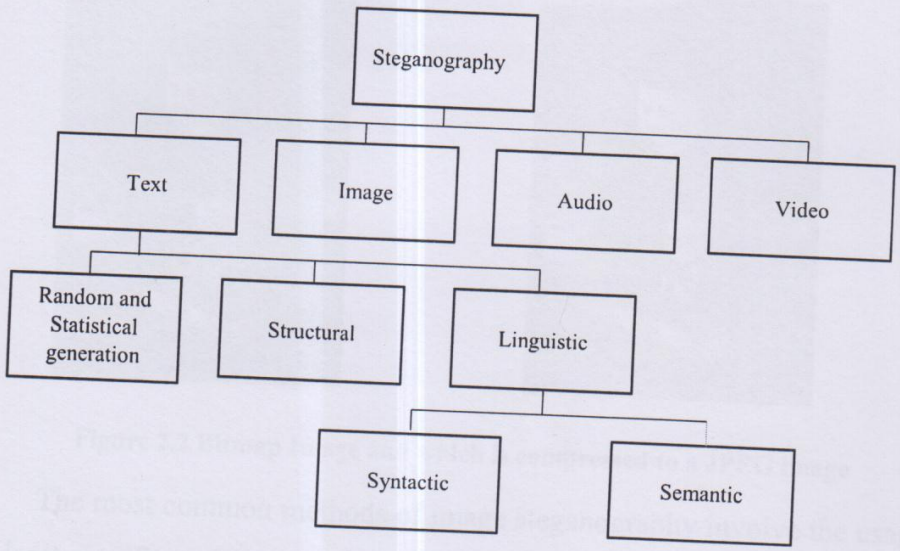

Figure 2.1 Types of Steganography

2.4.1 Image Steganography

Image steganography has gotten more popular press in recent years than other kinds of steganography, possibly because of the flood of electronic image information available with the advent of digital cameras and high-speed internet distribution. To hide a message inside an image without changing its visible properties, the cover source can be altered in "noisy" areas with many color

variations, so less attention will be drawn to the modifications. Most kinds of information contain some kind of noise. Noise can be described as unwanted distortion of information within the signal. Within an audio signal, the concept of noise is obvious. For images, however, noise generally refers to the imperfections inherent in the process of rendering an analog picture as a digital image. For example, the values of colors in the palette for a digital image will not only not be the exact colors in the real image, and the distribution of these colors will be also be imperfect. The image below is a bitmap image which is compressed to a JPEG image by the tool as the secret message is embedded.

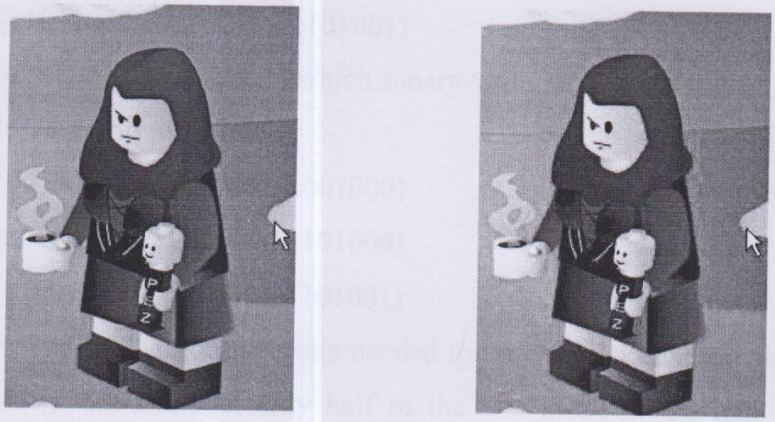

Figure 2.2 Bitmap Image and which is compressed to a JPEG image

The most common methods of image steganography involve the usage of the least-significant bit or LSB, masking, filtering and transformations on the cover image. The following techniques can be used with varying degrees of success on different types of image files [5].

- **Least Significant Bits:** A simple approach for embedding information in cover image is using Least Significant Bits (LSB). 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. To hide a secret message inside an image, a proper cover image

is needed. Because this method uses bits of each pixel in the image, it is necessary to use a lossless compression format, otherwise the hidden information will get lost in the transformations of a lossy compression algorithm. When using a 24-bit color image, a bit of each of the red, green and blue color components can be used, so a total of 3 bits can be stored in each pixel. For example, the following grid can be considered as 3 pixels of a 24-bit color image, using 9 bytes of memory:

(00100111 11101001 11001000)

(00100111 11001000 11101001)

(11001000 00100111 11101001)

When the character A, which binary value equals 01000001, is inserted, the following grid results:

(00100111 11101000 11001000)

(00100110 11001000 11101000)

(11001000 00100111 11101001)

In this case, only three bits needed to be changed to insert the character successfully. On average, only half of the bits in an image will need to be modified to hide a secret message using the maximal cover size. The result changes that are made to the least significant bits are too small to be recognized by the human visual system (HVS), so the message is effectively hidden. The least significant bit of third color is remained without any changes. It can be used for checking the correctness of 8 bits which are embedded in these 3 pixels. In other words, it could be used as "parity bit".

- **Masking and filtering:** Masking and filtering techniques, usually restricted to 24 bits or grayscale images, take a different approach to hiding a message. These methods are effectively similar to paper watermarks, creating markings in an image. This can be achieved by modifying the luminance of parts of the image. While masking does change the visible properties of an image, it can be done in such a way that the human eye will not notice the

anomalies. Since masking uses visible aspects of the image, it is more robust than LSB modification with respect to compression, cropping and different kinds of image processing. The information is not hidden at the "noise" level but is inside the visible part of the image, which makes it more suitable than LSB modifications in case a lossy compression algorithm like JPEG is being used.

- **Transformations:** A more complex way of hiding a secret inside an image comes with the use and modifications of discrete cosine transformations. Discrete cosine transformations (DST)), are used by the JPEG compression algorithm to transform successive 8 x 8 pixel blocks of the image, into 64 DCT coefficients each. Each DCT coefficient *F(u, v)* of an 8 x 8 block of image pixels *f(x, y)* is given by:

$$F(u,v)=\frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) * cos\frac{(2x+1)u\pi}{16} cos\frac{(2y+1)v\pi}{16}\right], \quad (2.1)$$

$$C(x)=f(x)=\begin{cases}\frac{1}{\sqrt{2}} & x=0\\ 1 & else.\end{cases} \quad (2.2)$$

After calculating the coefficients, the following quantizing operation is performed:

$$F^{Q}(u,v)=\left\lfloor\frac{F(u,v)}{Q(u,v)}\right\rfloor \quad (2.3)$$

Where *Q(u, v)* is a 64-element quantization table. A simple pseudo-code algorithm to hide a message inside a JPEG image could look like this [4]:

Input: message, cover image
Output: steganographic image containing message
while data left to embed do
get next DCT coefficient from cover image
if DCT 6= 0 and DCT 6= 1 then

get next LSB from message
replace DCT LSB with message bit
end if
insert DCT into steganographic image
end while

2.4.2 Audio Steganography

Audio steganography, the hiding of messages in audio "noise" (and in frequencies which humans can't hear), is another area of information hiding that relies on using an existing source as a space in which to hide information. Audio steganography can be problematic, however, since musicians, audiophiles, and sound engineers have been reported to be able to detect the "high-pitched whine" associated with extra high-frequency information encoded in messages. In addition to storing information in non-audible frequencies or by distorting the audible signal to include additional noise, an echo is introduced into the signal, and the size of the echo displacement from the original signal can be used to indicate 1's or 0's, depending on the size. Regardless of the kind of signal modification used, as with many steganographic techniques applied to images, changing an existing signal modifies its statistical profile in addition to potentially changing the audible qualities of the signal. Making such steganography less detectable depends on making the changes look like legitimately occurring noise or signal distortion [3].

The following is a short visual sample that happens when a 13k audio file is embedded into a 168k.wav file using Steganos the audio of the steganographic file sounds fuzzy, like a radio that is not well-tuned (the original sound file is quite clear). The representation captured below shows the end of the sound trailing off into silence. While both images (Figure 2.3 and Figure 2.4) are similar, an examination of the steganographic representation shows that

there is additional noise (this is made particularly obvious by the fact that line at the end of the right sample is much thicker than that of the left sample):

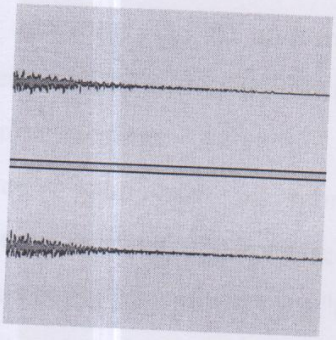

Figure 2.3 Clean.wav File (Cover Object)

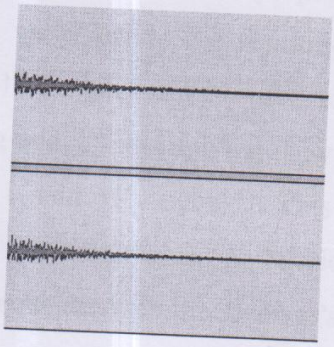

Figure 2.4 Stego.wav File

Among several methods of audio steganography, the following methods are some methods of audio steganography.

- **LSB Coding:** Sampling technique followed by Quantization converts analog audio signal to digital binary sequence. In this technique LSB of binary sequence of each sample of digitized audio file is replaced with binary equivalent of secret message.
- **Phase Coding:** Human Auditory System (HAS) can't recognize the phase change in audio signal as easy it can recognize noise in the signal. The phase coding method exploits this fact. This technique encodes the secret

message bits as phase shifts in the phase spectrum of a digital signal, achieving an inaudible encoding in terms of signal-to- noise ratio.

- **Spread Spectrum:** There are two approaches are used in this technique: the direct sequence spread spectrum (DSSS) and frequency hopping spread spectrum (FHSS). Direct-sequence spread spectrum (DSSS) is a modulation technique used in telecommunication. As with other spread spectrum technologies, the transmitted signal takes up more bandwidth than the information signal that is being modulated. Direct-sequence spread-spectrum transmissions multiply the data being transmitted by a "noise" signal. This noise signal is a pseudorandom sequence of 1 and −1 values, at a frequency much higher than that of the original signal, thereby spreading the energy of the original signal into a much wider band. The resulting signal resembles white noise. However, this noise-like signal can be used to exactly reconstruct the original data at the receiving end, by multiplying it by the same pseudorandom sequence (because $1 \times 1 = 1$, and $-1 \times -1 = 1$). This process, known as "de-spreading", mathematically constitutes a correlation of the transmitted Pseudorandom Noise (PN) sequence with the receiver's assumed sequence. For de-spreading to work correctly, transmit and receive sequences must be synchronized. This requires the receiver to synchronize its sequence with the transmitter's sequence via some sort of timing search process. In contrast, frequency-hopping spread spectrum pseudo-randomly retunes the carrier, instead of adding pseudo-random noise to the data, which results in a uniform frequency distribution whose width is determined by the output range of the pseudo-random number generator.
- **Echo Hiding:** In this method the secret message is embedded into cover audio signal as an echo. Three parameters of the echo of the cover signal namely amplitude, decay rate and offset from original signal are varied to represent encoded secret binary message. They are set below to the threshold of Human Auditory System (HAS) so that echo can't be easily resolved. Video

files are generally consists of images and sounds, so most of the relevant techniques for hiding data into images and audio are also applicable to video media. In the case of Video steganography sender sends the secret message to the recipient using a video sequence as cover media. Optional secret key 'K' can also be used during embedding the secret message to the cover media to produce 'stego-video'. After that the stego-video is communicated over public channel to the receiver. At the receiving end, receiver uses the secret key along with the extracting algorithm to extract the secret message from the stego-object.

The original cover video consists of frames represented by Ck(m,n) where 1 £ k £ N. 'N' is the total number of frame and m,n are the row and column indices of the pixels, respectively. The binary secret message denoted by Mk (m, n) is embedded into the cover video media by modulating it into a signal [4]. Mk (m, n) is defined over the same domain as the host Ck(m,n).The stego-video signal is represented by the equation:

$$Sk(m, n) = Ck(m, n)+ak (m, n) Mk(m, n) , k = 1, 2, 3 \ldots N \qquad (2.4)$$

Where ak (m, n) is a scaling factor. For simplicity ak (m, n) can be considered to be constant over all the pixels and frames. So the equation becomes:

$$Sk(m, n) = Ck(m, n)+a (m, n) Mk(m, n) , k = 1, 2, 3 \ldots N \qquad (2.5)$$

2.4.3 Video Steganography

A video can be viewed as a sequence of still images. Data embedding in videos seems very similar to images. However, there are many differences between data hiding in images and videos, where the first important difference is the size of the host media. Since videos contain more sample number of pixels or the number of transform domain coefficients, a video has higher

capacity than a still image and more data can be embedded in the video. Also, there are some characteristics in videos which cannot be found in images as perceptual redundancy in videos is due to their temporal features. Here data hiding operations are executed entirely in the compressed domain. On the other hand, when really higher amount of data must be embedded in the case of video sequences, there is a more demanding constraint on real-time effectiveness of the system. The method utilizes the characteristic of the human vision's sensitivity to color value variations. The aim is to offer safe exchange of color stego video across the internet that is resistant to all the steganalysis methods like statistical and visual analysis. Image based and video based steganographic techniques are mainly classified into spatial domain and frequency domain based methods. The former embedding techniques are LSB, matrix embedding etc. Two important parameters for evaluating the performance of a steganographic system are capacity and imperceptibility. Capacity refers to the amount of data that can be hidden in the cover medium so that no perceptible distortion is introduced. Imperceptibility or transparency represents the invisibility of the hidden data in the cover media without degrading the perceptual quality by data embedding. Security is the other parameter in the steganographic systems, which refers to an unauthorized person's inability to detect hidden data. In this Steganographic scheme secret data are embedded the I frame with maximum scene change and macro blocks of P and B frames based on motion vectors with large magnitude. To enlarge the capacity of the hidden secret information and to provide an imperceptible stego-image for human vision, tri-way pixel-value differencing (TPVD) algorithm is used for embedding.

Steganography in video can be divided into two main classes. One is embedding data in uncompressed raw video, which is compressed later. The other tries to embed data directly in compressed video stream. The problem of the former is how to make the embedded message resist video compression. But

because the video basically exists in the format of compression, the research of the latter is more significative [7].

2.4.4 Text Steganography

Text steganography involves anything like changing the format of an existing text, changing words within a text, generating random character sequences. Due to deficiency of redundant information which is present in image, audio or a video file, text steganography is believed to be the trickiest technique. In text documents, the information can be hidden by introducing changes in the structure of the document without making a notable change in the concerned output. Unperceivable changes can be made to an image or an audio file, but, in text files, even an additional letter or punctuation can be marked by a casual reader. Storing text file requires less memory and its faster as well as easier communication makes it preferable to other types of steganographic methods [1].

2.5 Types of Text Steganography

Text steganography can be broadly classified into three types:

1. Structural
2. Random and statistical generation,
3 Linguistic methods.

2.5.1 Structural

Structural-based text steganography involves manipulating the structure of the text in order to hide data. The structure includes the line spacing, word spacing, font size and anything similar. For example, one could use double word spacing to hide a 1, single to hide a zero. At a first glance, this will not be noticeable to the human eye. Margaret Thatcher reportedly used this method to aid in the prevention of press leakages of government documents. A certain number of white spaces were placed in documents, the number being related to

the cabinet minister to receive the document, so she could identify the owner of any leaked documents.

2.5.2 Random and Statistical Generation

In order to avoid comparison with a known plaintext, steganographers often resort to generating their own cover texts. While this often resolves the problem of a known cover attack, the properties of the generated text may still give rise to suspicions that the text is not legitimate. Such generation generally tries to simulate some property of normal text, usually by approximating some arbitrary statistical distribution found in real text. The non-linguistic text generation methods are:

- **Character sequences:** The hiding of information within character sequences is appealing because so much character-based information is available and transmitted over networks. One approach to text steganography might hide information in what appears to be a random sequence of characters. Of course, to both the person sending and receiving the message, this sequence is far from random, but it must appear to be random to anyone who intercepts the message. Not only must it appear to be random, however, but since we are also concerned with detection of the fact that this is a steganographic text, it must not appear to be suspicious. Random sets of characters which all fall within one character set but have no apparent meaning might indeed raise red flags. Because of this, a second approach to character generation is to take the statistical properties of word-length and letter frequency in order to create "words" (with no lexical value) which will appear to have the same statistical properties as actual words in a given language. These words might convince a computer which is only doing statistical analysis (and this is much less likely now that we are in an age where enormous dictionaries can be used to check the validity of words), but has clear problems with modern computer systems in terms of appearing suspicious.

- **Word sequences:** To solve the problem of detection of non-lexical sequences, actual dictionary items can be used to encode one or more bits of information per word. This might involve a code-book of mappings between lexical items and bit sequences, or the words themselves (length, letters, etc.) might encode the hidden information. However, this too has problems. A string of words with no semantic structure and no discernable semantic relationship can be detected by both humans and computers alike. Even in situations where computers cannot do complex syntactic analysis, simply classifying words and noticing extremely unlikely patterns (too many verbs or determiners in a row, no prepositions, etc.) within sequences of a certain length may be enough to alert the attacker to anomalous behavior. Even grammar-checkers used by modern word processors may be helpful tools in discovering ungrammatical texts. While legitimate ungrammatical texts certainly exist, given a certain context and threshold, such methods could be used to flag texts with no syntactic structure for further attention.
- **Statistical generation of sequences – text mimicking:** In addition to using the statistical frequency of letters or words in order to generate cover text, Wayner proposed a clever method of generating text which can be fairly convincing lexically and syntactically (and often semantically). This is not done by linguistic generation; rather, all 15 possible strings of length n are taken from a text or set of texts which are used as a sort of generator for the cover text. These strings are then organized in a table by how often they occur in the text. One string is picked at random to start the steganographic text, and the next letter in the text is chosen by looking at a window of the last $n - 1$ characters in the steganographic text and finding all of the strings in the table which start with those characters. The next letter is chosen from the last letter of these strings by using the data structures from the Huffman compression algorithm in reverse; the statistical frequency of all of the possible next letters that end the strings that start with the desired $n - 1$ characters is used to generate a tree which uses the

frequency of each of the selected strings to organize the last letters into an encoding tree. Letters which occur more frequently encode less information; this results in more frequently-occurring letters having to be used more often in the text to carry the same amount of information as one occurrence of a letter which occurs less frequently. For example, the data structure Wayner showed for the choice between the letters *e*, *a*, and *o* as the next letter in the text, where *e* occurred most frequently at the end of the strings we're examining, and *o* occurred least frequently, looks like Figure 2.5.

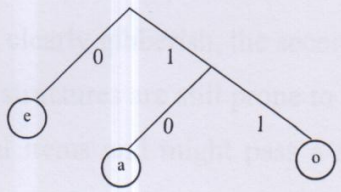

Figure 2.5 Text Mimicking

The next letter chosen in this example, then, depends on the bits we need to encode. If a "0" needs to be encoded, we use an *e* as the next letter. If a "1" needs to be encoded, we use either an *a* or an *o*. The choice depends upon whether or not a "0" or a "1" follows the "1" we need to encode. *a* encodes two bits, "10", and *o* encodes "11". Thus, *a* and *o* provide more information in a shorter space and need to be used less frequently. This, then, generates a profile not unlike the real text from which it was derived.

One of these data structures needs to exist for each set of strings with the same first $n - 1$ characters in the data set. Practically speaking, the larger n becomes, the better the results, but also the larger the amount of intermediate data structures which must be created for generation. For different sizes of n, Wayner gives examples of the algorithm's output using the text of a book chapter as the steganographic text generation seed. Each corresponding text is labeled as an *n-th order text*, where n is the string size mentioned above.

First order text:

islhne[hry saeeooisnre uo 'w nala al coehhs pebl e to agboean ceed cshcenapch nt sibPah ea m n [tmsteoia lahid egnndl y et r yfarleo awe l eo rttnntnnhtohwiseoa a dri 6oc7teit2t...

Fifth order text:

The letter compression or video is only to generate a verbatim>followed by 12 whiter 'H' wouldn't design a perfective reconomic data. This to simple hardware. These worked with encodes of...

While the first text is clearly gibberish, the second text is almost readable. The syntactic and semantic structures are still prone to serious error, but most of the words are actual lexical items and might pass a simple computer scan or even an uninterested human reader. More comprehensive linguistic details, of course, could cause detection problems. Additionally, it is possible that given the same generation "seed text", an attacker could figure out where the text was generated from. If that were to happen, given the right n, the attacker could generate the tables himself and figure out both that the text was generated by this scheme and what the original message was [4].

2.5.3 Linguistic Methods – Syntax and Semantics

Computational power is increasingly able to analyze more and more complex linguistic structures. While the texts generated in previous examples may approximate some of the appearance of legitimate text, "simulating the statistical properties of text" must increasingly handle linguistic properties as well as lexical and orthographic distribution. Linguistic steganography specifically considers the linguistic properties of generated and modified text, and in many cases, uses linguistic structure as the space in which messages are hidden. This section describes existing methods of linguistic steganography as background for the linguistic issues to be discussed in section 3 and the

proposed solution space in section 4. Because syntactic analysis can be used to detect ungrammatical sequences of lexical items, one solution to detection by syntactic steganalysis is to ensure that structures are syntactically correct from the start. In fact, steganographic data can be hidden within the syntactic structure itself. As such, Wayner proposed that context-free grammars (CFGs) be used as a basis for generation of steganographic texts. Because the text is generated directly from the grammar, unless the grammar is syntactically flawed, the text is guaranteed to be syntactically correct. Furthermore, the tree structure created by a CFG is a natural device to use for encoding bits. Tree structures are used as optimizing data structures in many areas of computer science, from compilers to sorting algorithms. In the simplest scheme, at any point where there is a branch in the syntax tree, the left branch might mean '0' and the right branch might mean '1'. This section describes existing methods of using CFGs in order to generate syntactically correct steganographic text.

The easiest way to generate syntactically correct texts is to generate them from the syntax itself. This seems obvious, but finding a way to hide information within such text may not necessarily be. Wayner proposes use of a custom-made CFG in which any choice branch in a production indicates a bit or a sequence of bits. We give a very simple grammar below, and explain how data is encoded. Wayner's method assumes that the grammar is in Greibach Normal Form (GNF) in order to prevent left recursion in parsing; that is, non-terminals come only at the end of productions. Given the following grammar:

Start := subject predicate

subject := propernoun || The noun

predicate := is adjective || can't be adjective

adjective := asleep || crusty

noun := moose || ball of cheese

propernoun := Oscar the Grouch || Trogdor the Burninator

By deciding that for every stage in the grammar where there is a choice, the first choice is a 0 and the second choice is a 1, we can encode bit sequences, and when parsing text created from this grammar, retrieve them. Note that if this grammar is expressed as a tree, bits are encoded according to a pre-order traversal. Thus, when a decision is made at a non-terminal node, the bit from that decision is encoded, followed by all of the decisions from the subtree of its left child, and then its right child(ren). In order to encode the bit string 0110, we do a pre-order traversal of the grammar. The Start symbol is processed first, from left to right; the first decision requires a 0 bit, so we choose a **proper noun** as a **subject** and encode a 0. We then need a 1, so when choosing a **proper noun**, we choose the second option, *Trogdor the Burninator*. Since all subtrees of **subject** have been processed, we now process **predicate** and its subtrees. Since the next bit we need in the sequence is a 1, we choose the second option for **predicate**, which is *can't be* **adjective.** Finally, we need a 0, so we choose the first option for **adjective**, which is *asleep*. So to encode the bit string 0110 with this grammar, we output the string "Trogdor the Burninator can't be asleep". On the other hand, given the phrase "The ball of cheese is crusty", we get "1101". The small grammar we've presented is impractical, of course, since only $2^4 = 16$ sentence possibilities can be made from it, and there would be too much repetition to be a real text. As an illustration, the bit string 1011011111100000 would produce the following text:

> *The moose can't be crusty. Trogdor the Burninator can't be crusty. The ball of cheese can't be asleep. Oscar the Grouch is asleep.*

Creating grammars of this size can be a gargantuan task. Even if the sentences generated by these grammars are readable by humans, the "writing style" that comes out of such generation may be so strange that both humans and computers may be able to detect the anomalies in vocabulary usage, register, syntactic constructions, and so forth. None of the methods so far has

any concept of the semantics of words within a syntactic structure. One step toward such an approach comes from Chapman and Davida, who created the NICETEXT system in order to do two things: first of all, they aimed to create "interesting sequences of parts-of speech", and secondly, they aimed to categorize such parts-of-speech by "types" (which are essentially semantic categories) and to incorporate these types into the syntactic structures used for generation in order to make the text more believable to a human reader (Chapman and Davida, 1997). This was done by compiling large code dictionaries which categorized words by these types, and by creating style sources which acted as syntactic templates in which the words could be inserted. Dictionaries are compatible with style sources if they contain all of the types needed by the style source templates. These code dictionaries, in addition to having words categorized by type, also contain the bit values which correspond to each word for the encoding of steganographic data. Style sources begin with a *sentence model*, which is a syntactic (typed) template with additional information for formatting the sentence (punctuation and capitalization, for example). When generating text, a syntactic structure is chosen from a *sentence model table*, which is a set of syntactic sentence structures (which have the same semantic categories as the vocabulary words in the created dictionary), and one of the words which matches the current part of speech and semantic type with the desired bit value is inserted accordingly (1998:34). Such texts have a distinct "style" which is derived by creating the sentence model table through large corpus analysis. Information is not encoded in the grammar itself, then, but by the choice of word which is inserted into the current sentence model (sentence models are chosen at random from the sentence model table). When the corpora used as sources for sentence model tables come from technical or obscure language sources (medical terms, legal proceedings, government 20 policies, archaic literature, etc.), there is a higher chance that the generated text will fool even human readers because the style

of these problems with both NICETEXT and simple CFG systems, came up with an improved system. The revised NICETEXT approach involves what the authors call "extensible contextual templates" combined with synonym replacement (2001). Known texts were again used to produce the sentence models; however, the sentence models themselves were extended into "full-blown contextual templates". These no longer randomly choose sentence models at random from a sentence model table, but rather have a large contextual template of sentence models which are derived from the corpus text (161-163). Furthermore, what were "types" in the templates of the original NICETEXT are now synonym categories based upon the words used in the original text. The code dictionaries, then, are now synonym dictionaries with bit values assigned to synonym choices for each synonym type. This provides a text which has a semantic coherence which previous methods of generating cover texts have lacked. Of course, sometimes synonym replacement causes problems; even when words are synonymous, they are not necessarily interchangeable within the same semantic structure. The classic example of such a set of synonyms is "eat" and "devour". One can *eat lunch*, and one can *devour lunch*. However, while one may have *eaten in the morning*, he cannot have *devoured in the morning*. The fact that *eat* may omit the direct object while *devour* requires one means that using them interchangeably will create semantic problems. The authors recognize such issues in the paper, and suggest culling the dictionary of such problem words; however, for a fully automated system with many large corpora and huge dictionaries, this may be an unscalable task. Additionally, if the original text is available to the adversary, semantic analysis would quickly uncover the fact that the stego object was generated from a template derived from the original text [4].

CHAPTER 3

LINGUISTIC-BASED TEXT STEGANOGRAPHY

Linguistic Steganography is concerned with hiding information in natural language text. One of the major transformations used in Linguistic Steganography is synonym substitution. Linguistic steganography comes in two forms: syntactic and semantic. Syntactic text steganography involves altering the structure of the text without significantly altering the meaning or tone. Semantic-based steganography is the focus of this paper. This method involves replacing words with their synonyms in order to hide data [9].

This thesis implements semantic-based text steganography by using Synonym Substitution-Based Algorithm. There are three sub-algorithms in Synonym Substitution-Based Algorithm: Synonym Retrieval Algorithm, Obfuscation Algorithm, and Deobfuscation Algorithm. The sending side uses Synonym Retrieval Algorithm and Obfuscation Algorithm to encode data, and then the receiving side uses Deobfuscation Algorithm.

3.1 ASCII

ASCII stands for American standard code for information interchange, which was developed by American National Standards Institute (ANSI). It is the most common code used by computers to translate text (letters, numbers, and symbols) into a form that can be sent to, and understood by, other computers and devices such as modems and printers [8]. Table 3.1 shows some of the standard ASCII characters and codes.

Table 3.1 Some Of the Standard ASCII Characters and Codes

Char	Binary	Decimal
...	...	...
A	01000001	065
B	01000010	066
C	01000011	067
D	01000100	068
E	01000101	069
F	01000110	070
...	...	...

3.2 Synonym Retrieval Algorithm

The Synonym Retrieval Algorithm is used to retrieve the same or similar meanings of the cover text from WordNet dictionary. This algorithm uses BNC to decide which of the four word type (noun, adjective, verb, adverb) is used and ANC to sort the retrieved synonym list (synset).

```
Data: word

Result: synset
synset = getsynonymsfromdictionary(word);
for i ← 0 to synsetsize do
synset.addall(getSynonymsfromdictionary(synset[i]);
end
for i ← 0 to synsetsize do
synset.setfrequency(getfrequency(synset[i-1], synset[i]);
end
synset.sort();
return synset;
```

Figure 3.1 Synonym Retrieval Algorithm

3.3 WordNet Dictionary

The dictionary that is used for Synonym Retrieval algorithm is the WordNet dictionary, produced by Princeton University in the US. WordNet is a lexical database for the English language. It groups English words into sets of synonyms called synsets, provides short, general definitions, and records the various semantic relations between these synonym sets. The purpose is twofold: to produce a combination of dictionary and thesaurus that is more intuitively usable, and to support automatic text and artificial intelligence applications. The database can be downloaded and used freely. The database can also be browsed online.

The dictionary contains around 150,000 words, organized into sets of synonyms, called synsets. A synset consists of the set of synonyms, a brief description (definition) of that sense, and in some cases one or two example sentences showing a typical usage of one of the words in the synset. There are around 115,000 synsets, some containing just a single word and some containing more than 8 synonyms. A search for a word in the dictionary database will return all synsets which contain that word, separated for the different word types (noun, verb, adverb and adjectives), and sorted by the frequency that they appear in some sample texts so the most common sense of a word is displayed first. Searches for words that appear in the same synset return the same synset [12].

When retrieving the synonyms for a word from the WordNet dictionary, in some cases this is simple as the word can only be of one type. For example searching the word "happy" shows only one type, "adjective". So we don't need to choose the word type. Figure 3.2 shows querying synset of the word "happy". The query shows only one word type (adjective). The word type noun has four senses and the synset is happy, felicitous, well-chosen.

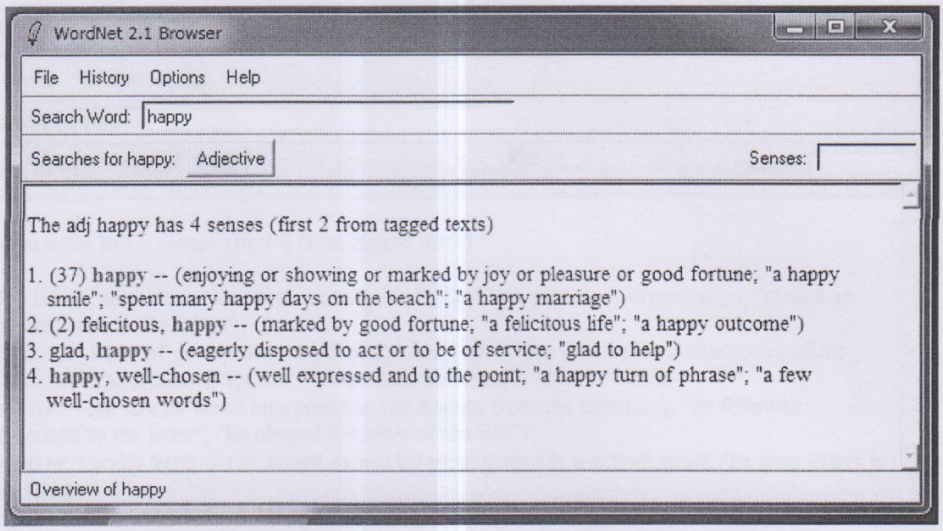

Figure 3.2 : Screenshot of WordNet Application after Querying the Word "happy"

But in many cases, the word can be of two or more of the possible types as shown in Figure 3.3. Figure (3.3) shows querying synset of the word "letter". The query shows two different word types (noun and verb). The word type noun has five senses and the verb type has three senses. The noun synset for the noun word type are letter, missive, letter of the alphabet, alphabetic character, varsity letter. The synset for the verb word type is letter.

If the wrong type is chosen, then the synonym chosen may not make sense. To choose the most likely word type, the word frequency data from the BNC (British National Corpus) is used.

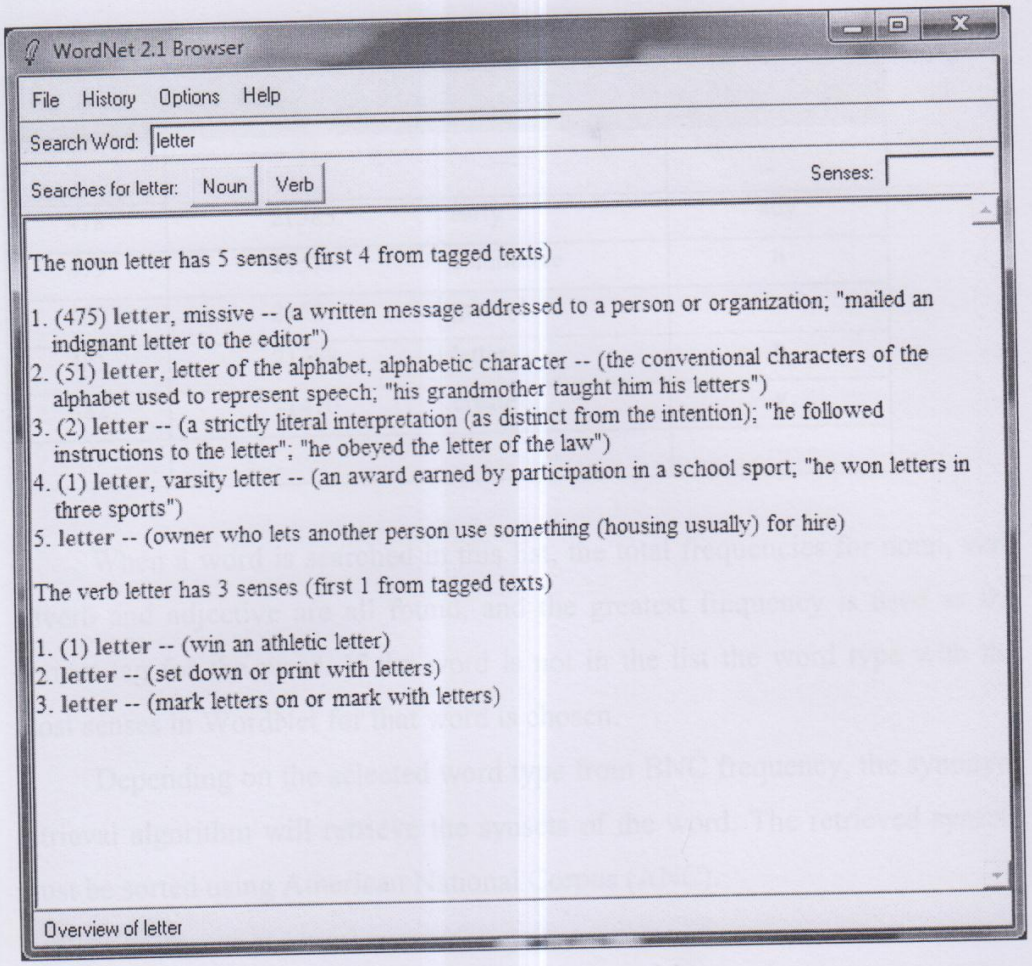

Figure 3.3: Screenshot of WordNet Application after Querying the Word "letter"

3.4 British National Corpus (BNC)

The British National Corpus (BNC) is a 100 million-word collection of samples from a wide range of sources such as regional and national newspapers, specialist periodicals and journals for all ages and interests, academic books and popular fiction, school and university essays, and many other kinds of text. There are four columns in BNC: No (sort order), frequency, word, word type (noun, verb, adverb, adjective, etc.,). Table 3.2 shows some of the BNC frequency data.

Table 3.2 Some of the BNC Frequency Data

No.	BNC Frequency	Word	Word Type
…	…	…	….
478	21585	early	adv
479	21575	committee	n
480	21504	ground	n
481	21488	letter	n
482	21470	create	v
…	…	…	….

When a word is searched in this list, the total frequencies for noun, verb adverb and adjective are all found, and the greatest frequency is used as the correct tag for the word. If the word is not in the list the word type with the most senses in WordNet for that word is chosen.

Depending on the selected word type from BNC frequency, the synonym retrieval algorithm will retrieve the synsets of the word. The retrieved synsets must be sorted using American National Corpus (ANC).

3.5 American National Corpus (ANC)

The American National Corpus (ANC) is a text corpus of American English containing 22 million words written and spoken data. The American National Corpus (ANC) is a massive electronic collection of American English produced from 1990 onward. There are three columns in ANC: word, ANC frequency and ratio (the frequency ratio for the word) as shown in Table 3.3.

Table 3.3 Some of the ANC Frequency Data

Word	ANC Frequency	Ratio
…	…	…
jobs	1739	0.0000784571
reference	1738	0.0000784120
meaning	1736	0.0000782317
saving	1734	0.0000782315
negative	1731	0.0000780962
…	…	…

ANC contains the lists of bigrams of words and their frequencies. The bigram data is limited to bigrams which have a frequency of more than 4, as below this the frequency is too low and also it limits the amount of data that needs to be searched (the full set is over 2 million in size, with this limit it is approximately 250 thousand).

3.6 Obfuscation Algorithm

Obfuscation is the term used to represent hiding data in the text. To obfuscate, two components are required, the synset that is available from Synonym Retrieval Algorithm and the bits to hide (the list of binary data that need to be hidden). This algorithm is used by the sender of the message. Figure 3.4 shows the obfuscation algorithm.

Data: Synset, bit to hide
Result: Chosen Synonym
if *synset size less than bit* **then**
return synset element at position bit;
end

Figure 3.4 Obfuscation Algorithm

3.7 Deobfuscation Algorithm

This algorithm produces the hidden bits by matching the synset and each word of the stego text. This algorithm is used by the receiver of the message. Figure 3.5 shows the Deobfuscation Algorithm.

Data: Synset, word

Result: Hidden bit(s)

for $i \leftarrow 0$ **to** *synsetsize* **do**

if *synset[i] = word* **then**

return i;

end

end

return null;

Figure 3.5 Deobfuscation Algorithm

CHAPTER 4

DESING, IMPLEMENTATION AND RESULT

This chapter presents the design, implementation and results conducted in this thesis. This thesis implements text steganography by applying Synonym Substitution-Based Algorithm in letter of credit application. At sending side, the issuing bank uses ASCII table to get equivalent binary data of message, synonym retrieval algorithm to retrieve the synonyms of the message from WordNet dictionary, obfuscation algorithm to encode the LC information. While retrieving the synonyms from WordNet dictionary, and synonym retrieval algorithm uses BNC frequency data to choose the possible word type and ANC frequency data to sort the retrieved synonyms. At receiving side, the advising bank uses deobfuscation algorithm to decode the LC information. Figure 4.1 shows the architecture of the system.

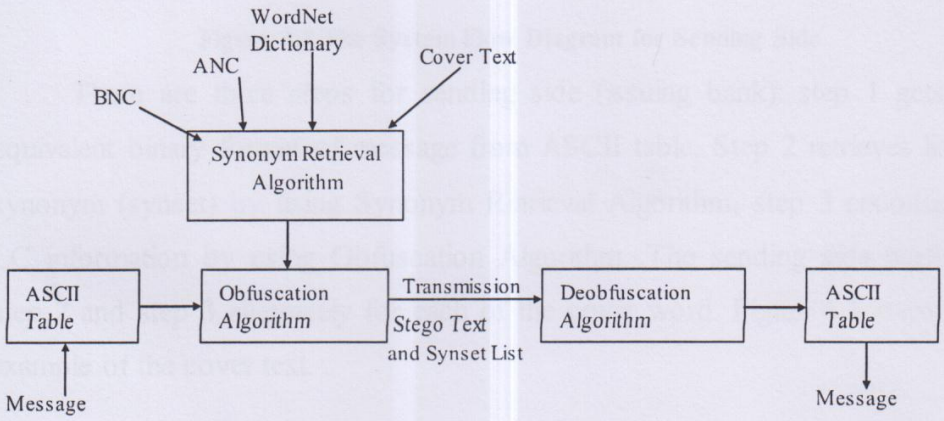

Figure 4.1 System Architecture

4.1 Sending Side

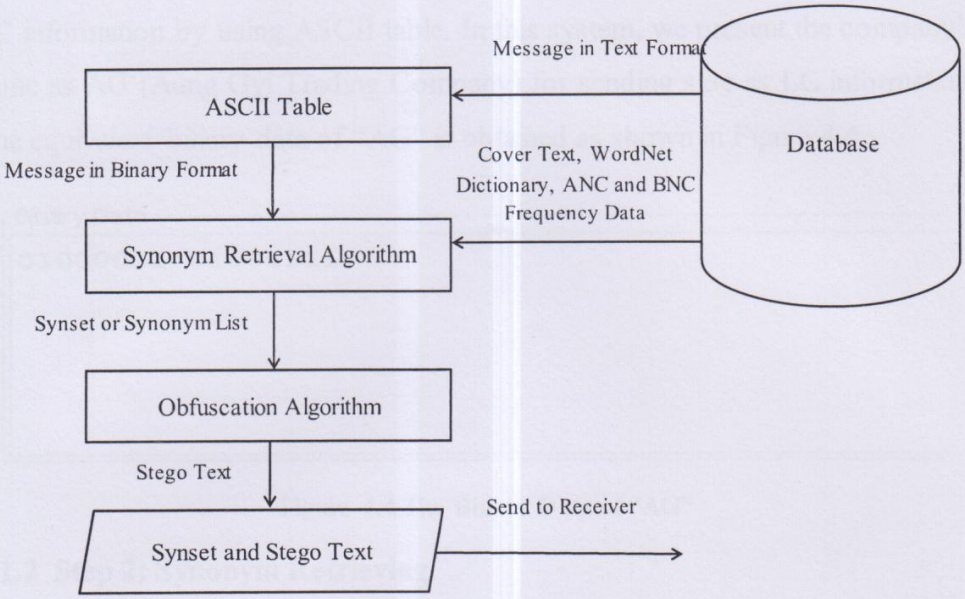

Figure 4.2 The System Flow Diagram for Sending Side

There are three steps for sending side (issuing bank): step 1 gets the equivalent binary format of message from ASCII table. Step 2 retrieves list of synonym (synset) by using Synonym Retrieval Algorithm, step 3 encodes the LC information by using Obfuscation Algorithm. The sending side performs step 2 and step 3 alternately for each of the cover word. Figure 4.3 shows the example of the cover text.

This Letter of Credit is automatically renewable without amendment for additional one year periods from the present expiry date or on future expiration date, unless 30/60/90 days prior to such expiration date we shall notify you in writing by registered or courier mail that we elect not to renew this Letter of Credit.

Figure 4.3 The Example Of Cover Text

4.1.1 Step 1: Getting the Equivalent Binary Format of Message from ASCII Table

Firstly, the issuing bank gets the equivalent binary form of the buyer's LC information by using ASCII table. In this system, we present the company's name as AG (Aung Gyi Trading Company) for sending side as LC information. The equivalent binary data of "AG" is obtained as shown in Figure 4.4.

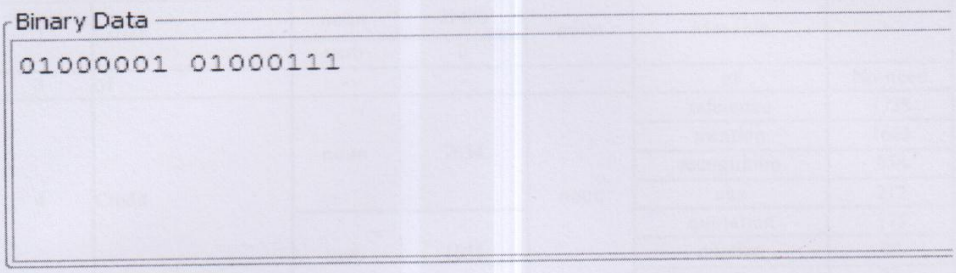

Figure 4.4 The Binary Data of "AG"

4.1.2 Step 2: Synonym Retrieving

Synonym retrieving is processed by synonym retrieval algorithm. In Figure 4.5, the processing with the first 12 words of the cover text is shown as the detailed processing of the sending side. Table 4.1 shows the example of cover text.

The first task in retrieving the synonyms from the WordNet dictionary for a cover word is to decide which of the four word types (noun, verb, adverb or adjective) the word is. In some cases, this is simple because the word can only be of one type (No 6, 7, and 9, 12 of Figure 4.4) but in many cases, the word can be two or more of the possible types (No 2, 4, and 5 of Figure 4.4). To choose the most likely word type, the word frequency data from the BNC is used. When a word is searched for BNC, the total frequencies for noun, verb, adverb and adjective are all found, and the greatest frequency is used as the correct word type for the word.

In Figure 4.5 there are two word types for the word "letter", noun and verb (No 2 of Figure 4.4). BNC frequency for "noun" word type is "21488" and for verb word type is "0". Therefore, the "noun" word type is selected.

Table 4.1 The Detailed Processing of Synonym Retrieval Algorithm

No	Word	All Word Types	BNC Frequency	Selected Word Type	Synset	ANC Frequency
1	This	-	-	-	This	-
2	Letter	noun	21488	noun	Massive	-
		verb	0			
3	of	-	-	-	of	No need
4	Credit	noun	7634	noun	reference	1738
					mention	1663
					recognition	658
					cite	212
		verb	1035		quotation	175
					citation	94
					acknowledgement	51
5	is	noun	0	-	is	No need
		verb	0			
6	automatically	adverb	-	adverb	mechanically	No need
7	renewable	adjective	-	adjective	renewable	No need
8	without	-	-	-	without	No need
9	amendment	noun	-	noun	amendment	No need
10	for	-	-	-	for	No need
11	additional	-	-	-	additional	No need
12	one	noun	-	noun	single	4759
					unity	191
					ace	162
...	...	...	...	...	...	...

Synonyms are retrieved from WordNet dictionary by using this selected word type. During retrieving process, synonyms that are composed of two or more words are ignored because it is impossible to know that they present one synonym when deobfuscating. In Figure 3.3 of Chapter 3, synonym of "letter" in "noun" word type is "massive" because all other synonyms are composed of two words. In some cases, there are only one synonym is retrieved, (No 2 of Table 4.1), and in some cases, there may be more than one synonym (No 4 and 12 of Table 4.1). The synsets that are more than one synonyms are sorted by using ANC frequency data to get the correct position when deobfuscating. The

final synset list is shown in Figure 4.5. This list is taken from synset column of Table 4.1.

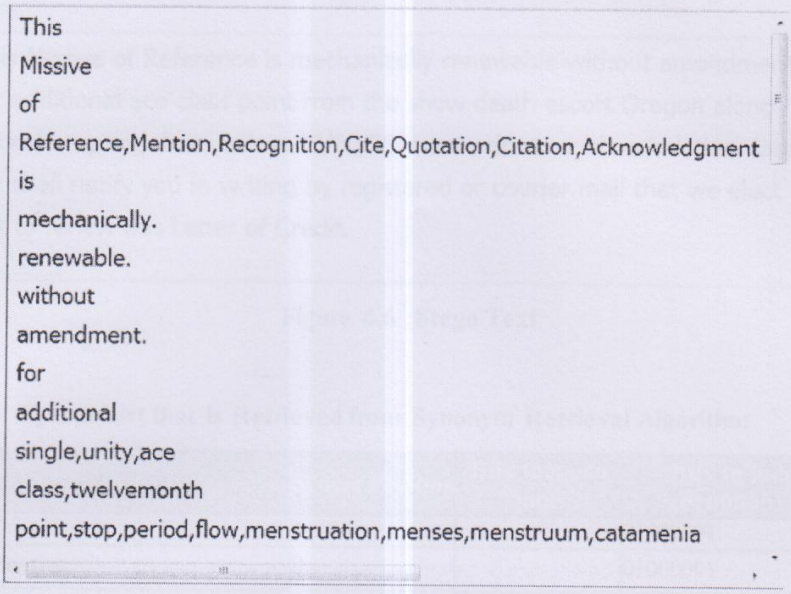

Figure 4.5 Synset List Retrieved from Synonym Retrieval Algorithm

4.1.3 Step 3: Obfuscation

Obfuscation is the term used to represent hiding data in the text. To obfuscate, two components are required: the synset that is available from Synonym Retrieval Algorithm and the bits to hide (the list of binary data that need to be hidden). Table 4.2 shows the detailed processing of Obfuscation algorithm.

If there is only one synonym in synset, no bits can be hidden (No 1, 2, 3, 5, 6, 7, 8, 9, 10, 11 of Table 4.2) and it is added to a list known as stego text. Else, if the first bit in the bit string is a 0, the first element at the first position is returned (No 4 of Table 4.2), if it is a 1 the element at the second position: if the first two elements are 10 (No 12 of Table 4.2), the element at the third position is returned, if they are 11 then the element at the forth position is returned. After all, the returned element is added to stego text. Finally, the synset list retrieved

from synonym retrieval algorithm and the stego text available from obfuscation algorithm are sent to the advising bank at the receiving side. Figure 4.6 shows the stego text.

This Missive of Reference is mechanically renewable without amendment for additional ace class point from the show death escort Oregon along hereafter going date, unless 30/60/90 years prior to such exhalation date we shall notify you in writing by registered or courier mail that we elect not to renew this Letter of Credit.

Figure 4.6 Stego Text

Table 4.2 Synset List that is Retrieved from Synonym Retrieval Algorithm

No	Synset	Synset Position	Bits that can hide	Bits to hide (01000001)
1	This	0	-	01000001
2	Massive	0	-	01000001
3	of	0	-	01000001
4	reference	0	0	1000001
	mention	1	1	
	recognition	2	10	
	cite	3	11	
	quotation	4	100	
	citation	5	101	
	acknowledgment	6	110	
5	is	0	-	1000001
6	mechanically	0	-	1000001
7	renewable	0	-	1000001
8	without	0	-	1000001
9	amendment	0	-	1000001
10	for	0	-	1000001
11	additional	0	-	1000001
12	single	0	0	00001
	unity	1	1	
	ace	2	10	
...	...	...	...	...

4.2 Receiving Side

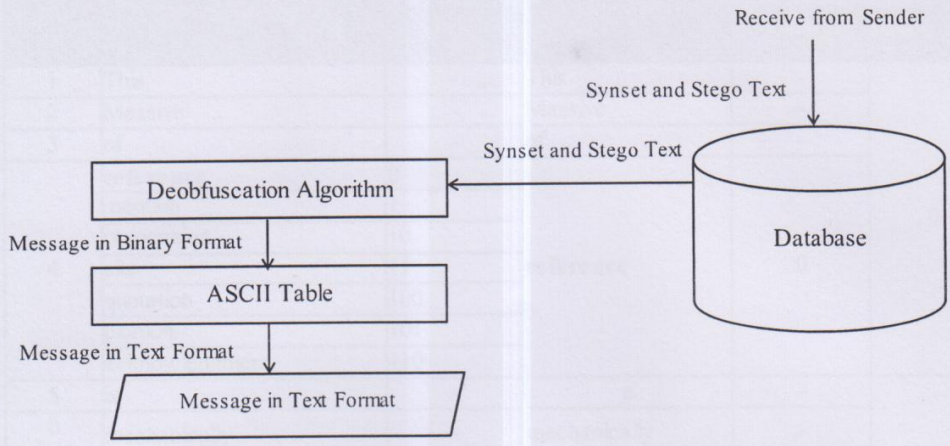

Figure 4.7 System Flow Chart for Receiving Side

There are two steps for receiving side (advising bank): step 1 decodes the LC information by using deobfuscation algorithm and step 2 gets the equivalent string format of the available bit string from the obfuscation algorithm by using ASCII table.

4.2.1 Step 1: Deobfuscation

Deobfuscation refers to extracting hidden data from the text. To deobfuscate, two components are required: the stego text and the synset list.

The algorithm matches each word of the stego text and the synset list. If the stego word is the same with the synset in synset list, then the position of the synset is returned (0 for the first position, 1 for the second position, 10 for the third position, and 11 for the fourth position) and so on. The algorithm goes through each synset list and stego text. Finally, a list of binary string is returned. Table 4.3 shows the detailed processing of deobfuscation.

Table 4.3 The Detailed Processing of Deobfuscation

No	Synset	Bits that can hide	Stego Text	Selected Binary	Current Binary Data
1	This	-	This	-	
2	Massive	-	Massive	-	
3	of	-	of	-	
4	**reference**	**0**	**reference**	0	0
	mention	1			
	recognition	10			
	cite	11			
	quotation	100			
	citation	101			
	acknowledgment	110			
5	is	-	is	-	
6	mechanically	-	mechanically	-	
7	renewable	-	renewable	-	
8	without	-	without	-	
9	amendment	-	amendment	-	010
10	for	-	for	-	
11	additional	-	additional	-	
12	single	0	**ace**	10	
	unity	1			
	ace	**10**			
...	...	...	...	...	...

4.2.2 Step 2 : Getting the Equivalent String Format of Message from ASCII Table

Finally, the equivalent string format of the binary data that is output deobfuscation algorithm is obtained from ASCII table. Then, the message "AG" is obtained.

4.3 Implementation

When the program is started to run, the Main Window Form of the system appears as shown in Figure 4.8.

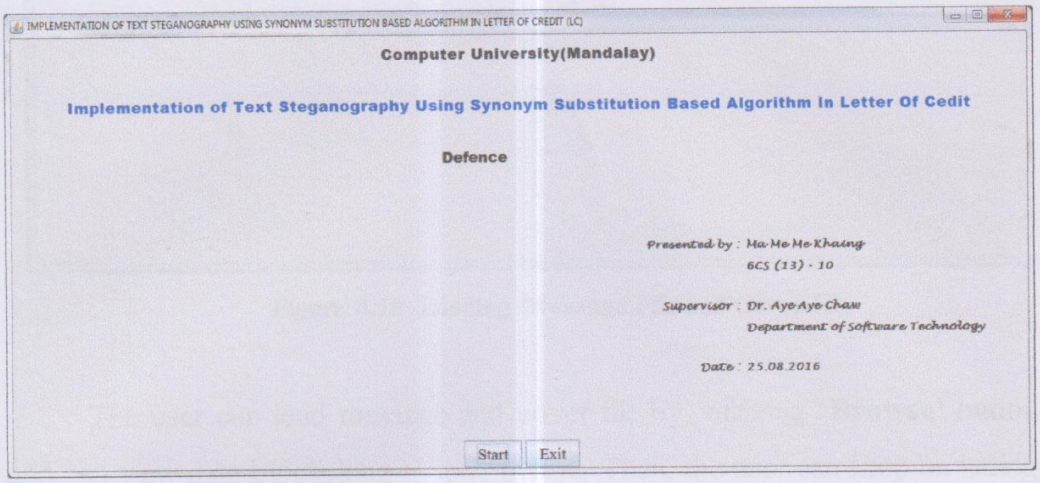

Figure 4.8 Main Window Form of the System

Sending Side

By clicking 'Start' button, the sender can see the following screen.

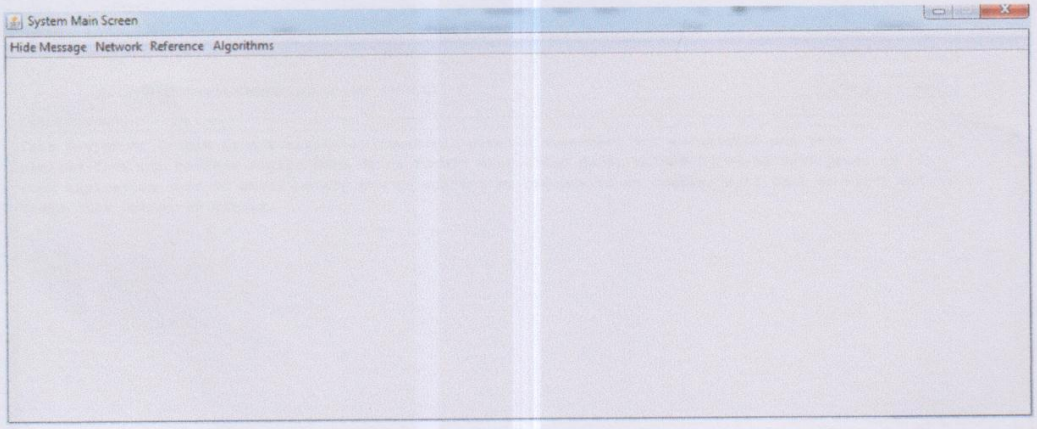

Figure 4.9 System Main Screen for Sending Side

To hide message, the user must load message file and cover file as shown in Figure (4.9) by clicking 'Hide Message' Menu. To load Message and Cover file, the user must select 'Load Message and Cover' as shown in Figure 4.10.

4.3 Implementation

When the program is started to run, the Main Window Form of the system appears as shown in Figure 4.8.

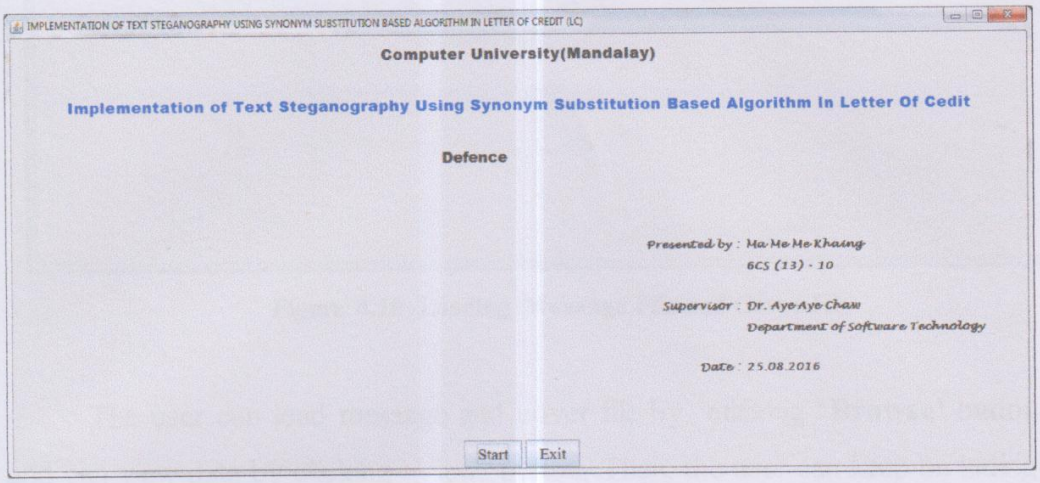

Figure 4.8 Main Window Form of the System

Sending Side

By clicking 'Start' button, the sender can see the following screen.

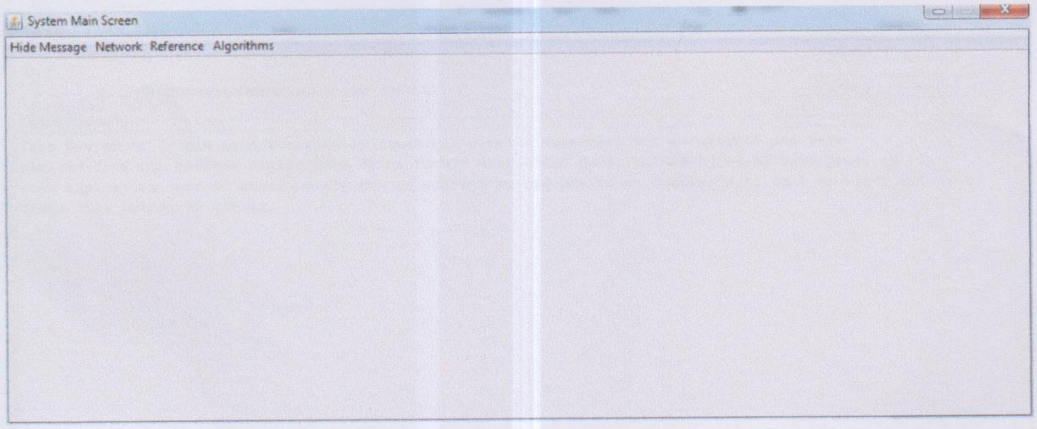

Figure 4.9 System Main Screen for Sending Side

To hide message, the user must load message file and cover file as shown in Figure (4.9) by clicking 'Hide Message' Menu. To load Message and Cover file, the user must select 'Load Message and Cover' as shown in Figure 4.10.

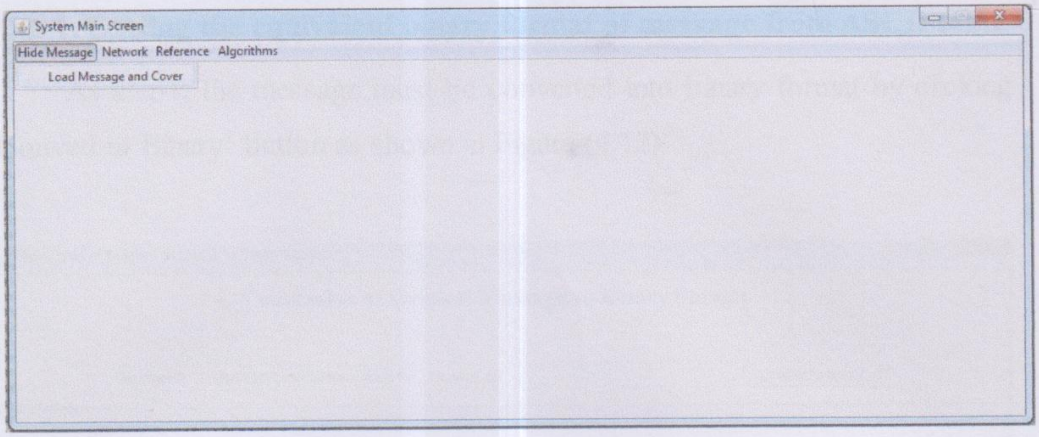

Figure 4.10 Loading Message File and Cover File

The user can load message and cover file by clicking '**Browse**' button and can view data by clicking 'View' button. Then, the user can keep on hiding process by clicking 'Auto Hide' button as shown in Figure (4.11).

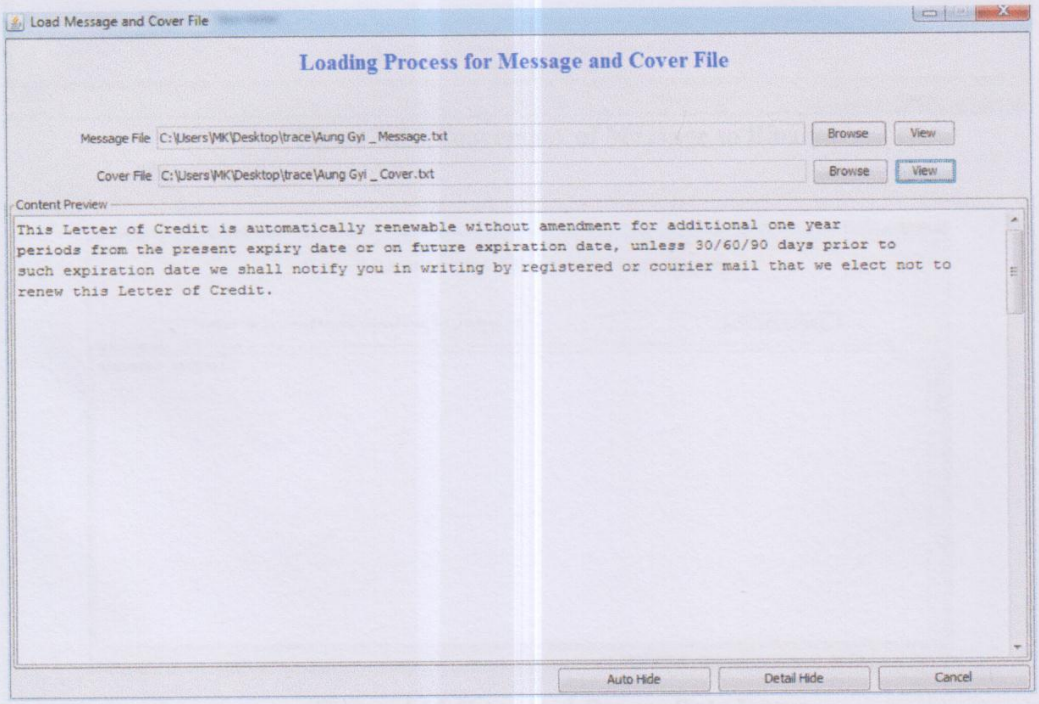

Figure 4.11 Loading Process for Message File and Cover File

Step 1: Getting the equivalent binary format of message from ASCII table

As step1, the message must be converted into binary format by clicking 'Convert to Binary' button as shown in Figure (4.12).

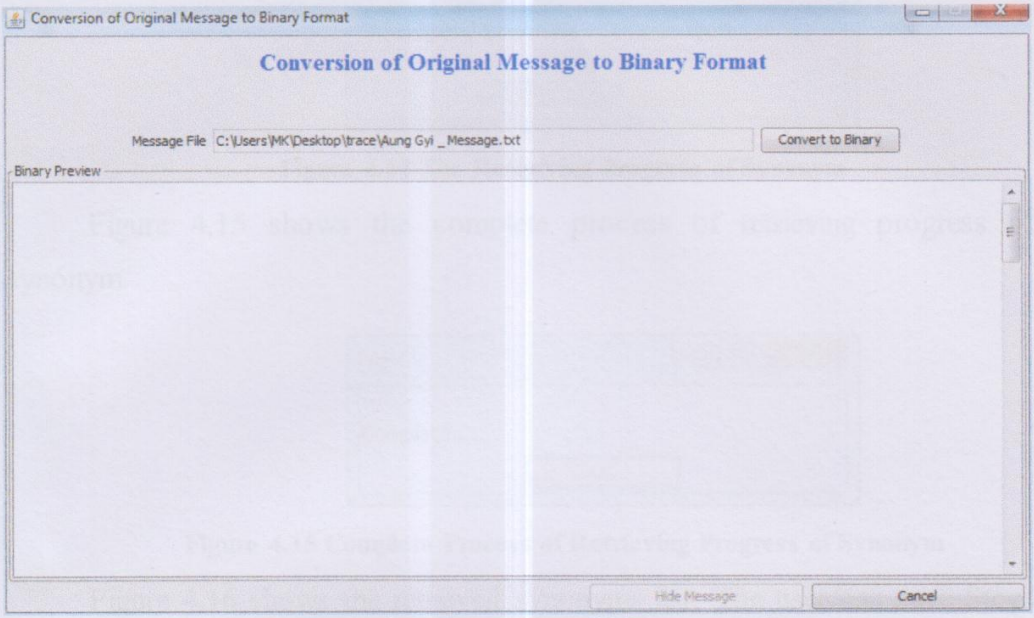

Figure 4.12 Conversion of Message to Binary

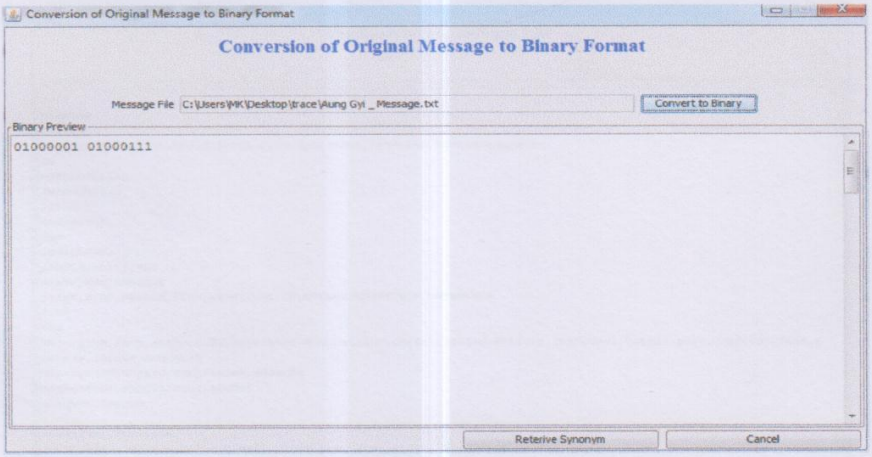

Figure 4.13 Converted Binary Data Frame

Figure 4.13 shows converted binary data of message.

Step 2: Synonym Retrieving

The user can retrieve synonym by clicking 'Retrieve Synonym' button from Figure (4.13). Figure 4.14 shows the retrieving progress of synonym.

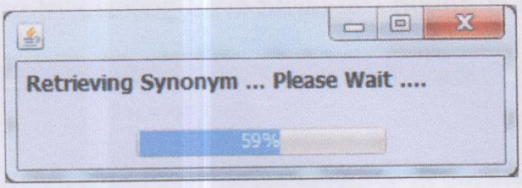

Figure 4.14 The Reteriving Progress of Synonym

Figure 4.15 shows the complete process of retrieving progress of synonym.

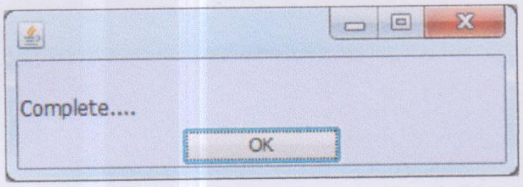

Figure 4.15 Complete Process of Retrieving Progress of Synonym

Figure 4.16 shows the retrieved synonyms list. The user can save Stego text by clicking Save Stego text button in Figure 4.16.

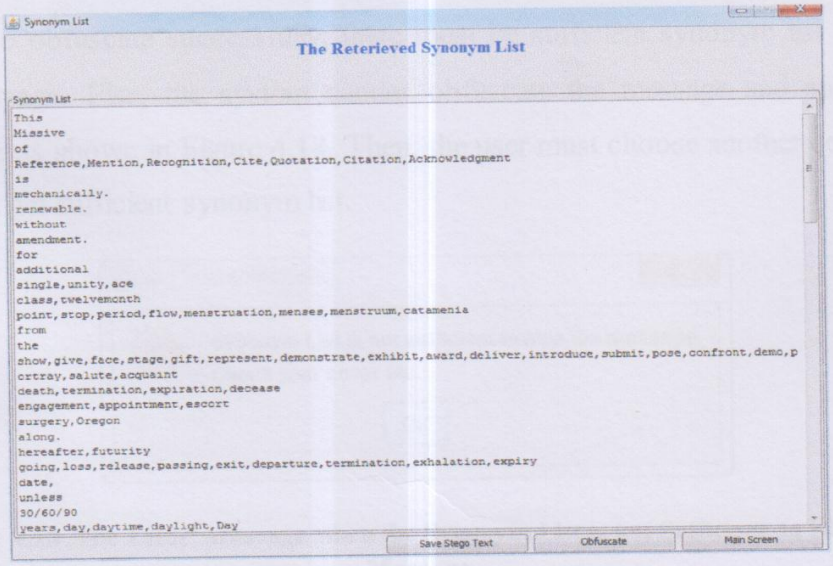

Figure 4.16 The Retrieved Synonyms List

Step 3: Obfuscation

The user can obfuscate or hide message by clicking Obfuscate button of Figure 4.16. After obfuscation, Stego text is obtained as shown in Figure 4.17. The user can save Stego text by clicking Save Stego text button in Figure 4.17.

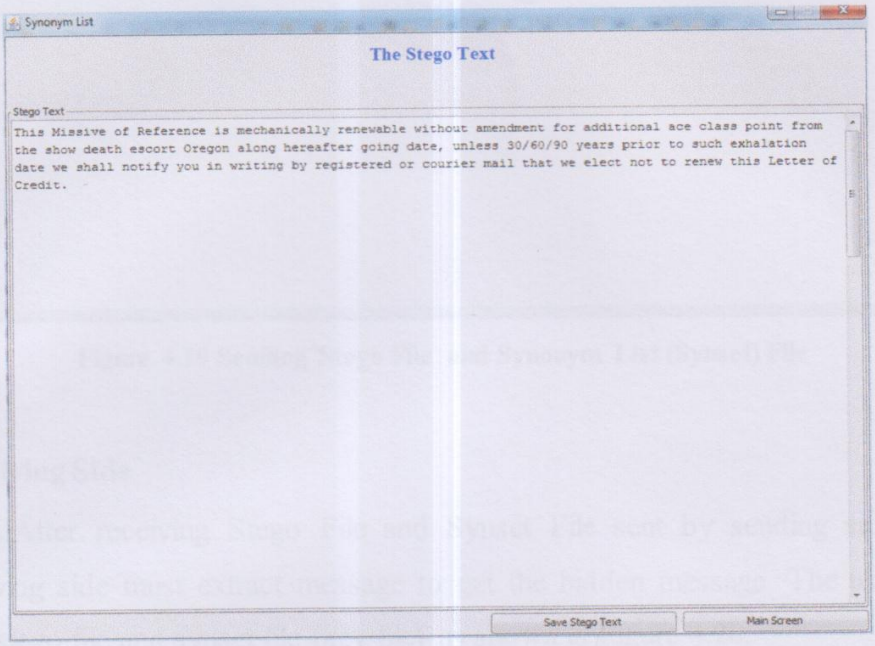

Figure 4.17 The Stego Text

To obfuscate successfully, there must be sufficient synonym list to hide the message. Else, the system cannot obfuscate the message and show the message as shown in Figure 4.18. Then, the user must choose another cover file that can get sufficient synonym list.

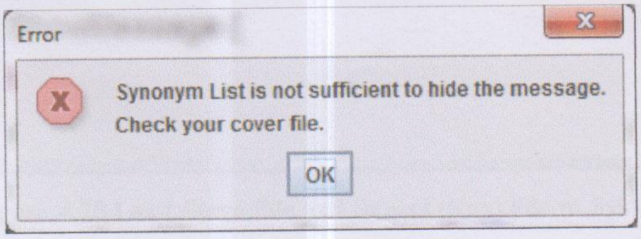

Figure 4.18 The Error Message when the Synonym List is not Sufficient to Hide the Message

Afterall, the user can send Stego file and Synonym list (Synset) file as shown in Figure 4.19.

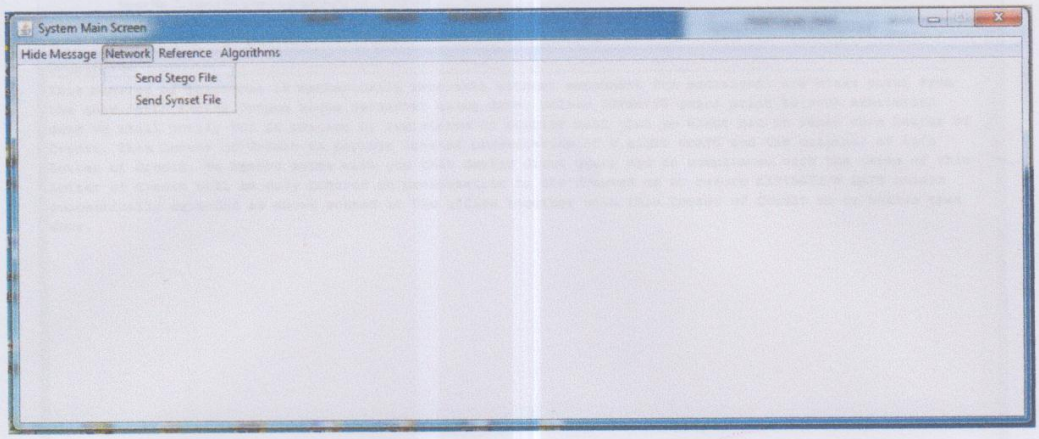

Figure 4.19 Sending Stego File and Synonym List (Synset) File

Receiving Side

After receiving Stego File and Synset File sent by sending side, the receiving side must extract message to get the hidden message. The user can load stego file and synset file (key file) as shown in Figure 4.20.

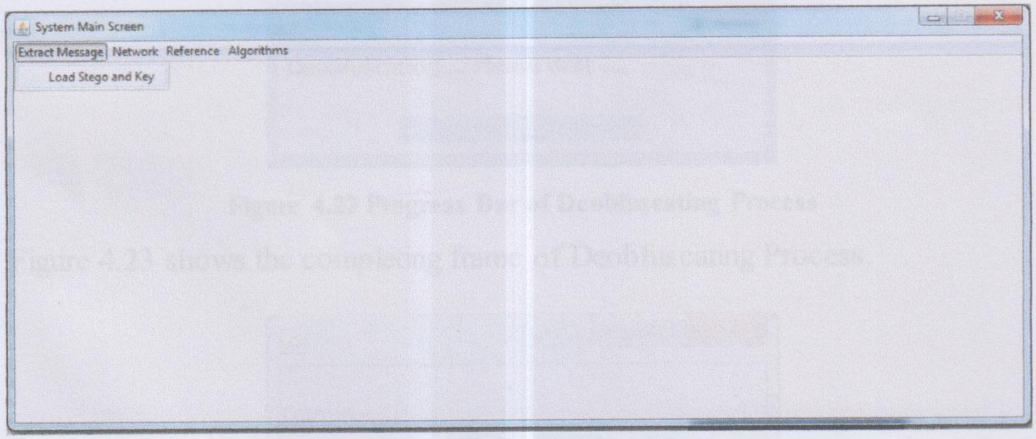

Figure 4.20 Load Stego File and Synset (Key) File to System

Then, the user can load stego file and synset file by clicking 'Browse' button as shown in Figure 4.21.

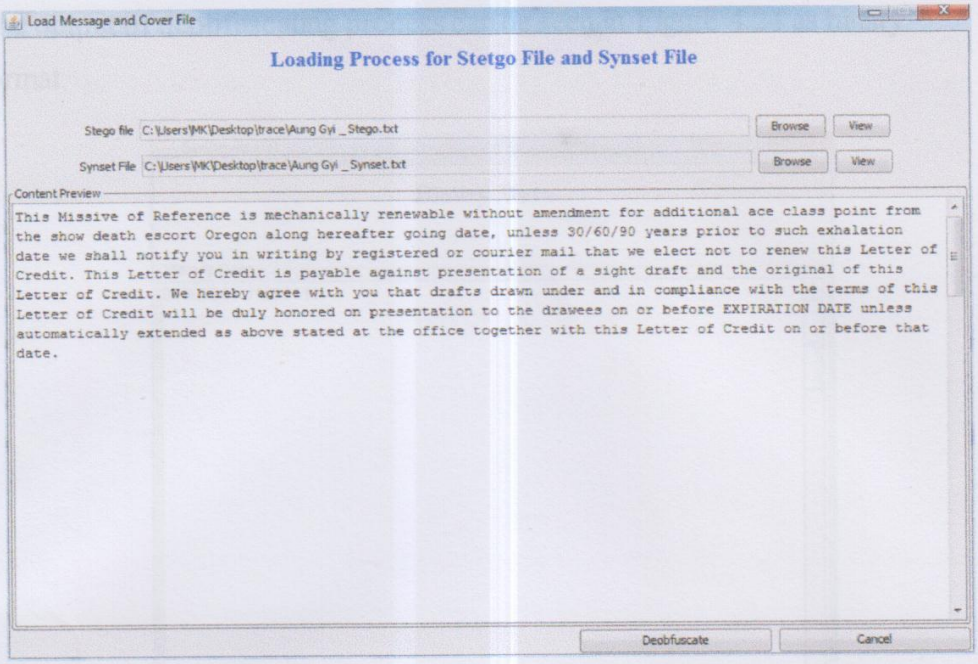

Figure 4.21 Loading Process for Stego File and Synset File to System

Step 1: Deobfuscation

The user must click 'Deobfuscate' button of Figure 4.19 to deobfuscate or extract message. Figure 4.22 shows the progress bar of deobfuscating process.

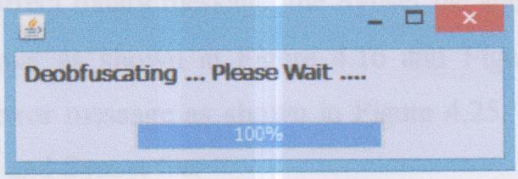

Figure 4.22 Progress Bar of Deobfuscating Process

Figure 4.23 shows the completing frame of Deobfuscating Process.

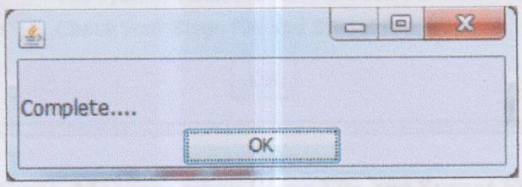

Figure 4.23 Completing Frame of Deobfuscating Process

The output of deobfuscating process can be seen in Figure 4.24 as binary format.

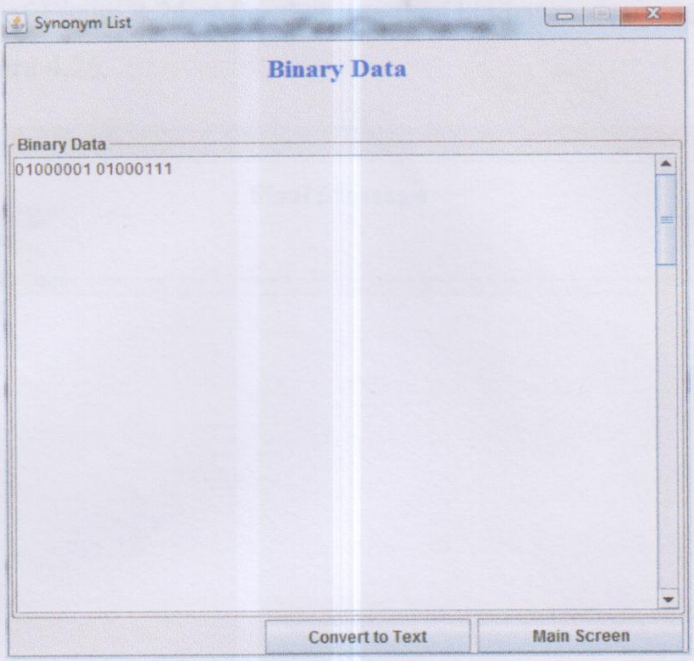

Figure 4.24 The Equivalent Binary Data of Message

To get the correct binary message, the Synset file and the Stego file must be in the right format as shown in Figure 4.16 and Figure 4.17. If not, the system shows the error message as shown in Figure 4.25. Then, the user must check the Stego file and Synset file.

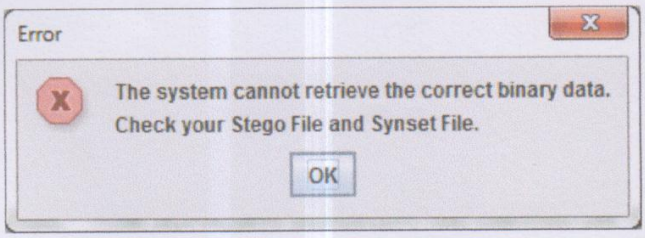

Figure 4.25 The Error Message when Stego File and Synset File are not Correct

Step 2: Getting the Equivalent String Format Of Message

The hidden message can be seen in string format by clicking 'Convert to Text' button of Figure 4.24. After that, the final hidden message can be seen as shown in Figure 4.26.

Figure 4.26 The Hidden Message

CHAPTER 5

CONCLUSION, LIMITATIONS AND FURTHER EXTENSION

This chapter concludes the system, and points out the limitations and further extensions of the system.

5.1 Conclusion

This system implements Synonym Substitution based Algorithm to send and receive LC information between issuing bank and advising bank. There are two sides in this system, the sending side and the receiving side. The issuing bank which encodes the LC information is in sending side, and the advising bank which decodes the LC information is in receiving side.

To encode the LC information, the issuing bank needs the LC information and the carrier text file (also called cover file). There are three steps for the issuing bank: Step1 converts the string format of LC information into binary format by using ASCII table. Step2 retrieves synonyms (same meaning) of the cover text from WordNet Dictionary by using Synonym Retrieval Algorithm. The Synset or Synonym List is obtained as output of Synonym Retrieval Algorithm. Then, step3 encodes the LC information by using Obfuscation Algorithm. The Stego Text is obtained from Obfuscation Algorithm. Then, the Synset and the Stego Text are sent to the issuing bank. To decode the LC information, the advising bank needs the Synset or Synonym List and the Stego Text that are received from the issuing bank. There are two steps for issuing bank to decode the LC information. Step1 decode the LC information by using Deobfuscation Algorithm and the LC information in binary format is obtained. Step2 converts the obtained binary format into text format by using ASCII table. After all, the LC information is decoded by the advising bank safely.

Text is still the primary form of communication not only online but also in the world where the computers and internet are not widespread. In this

system, the encoded LC information is in human readable text, therefore it is difficult to suspect that whether the text file hides information or not. So, the sender and the receiver can safely send and receive the LC information.

5.2 Limitations

This system works well when cover text file contains only text. If the hidden message amount is greater, the cover text needs more text. This system only works in text file and does not work in other types of files such as image, audio and video.

5.3 Further Extension

This system extracts synonym list from WordNet dictionary. It is extensible to extract synonym list from any other dictionary that provides synonym list. The user can also get synonym list from printed dictionary. This system is implemented as window-based application. It can be extensible to web application or mobile phone application.

REFERENCES

[1] Aditya K, Dr. Suresh P and Neela M, "Advance Text Steganography Algorithms : An Overview", International Journal of Research and Applications Jan-Mar 2014.

[2] Ching-Yun C, Stephen C " Practical Linguistic Steganography using Contextual Synonym Substitution and Vertex Colour Coding", URL : http://www.aclweb.org/anthology/D10-1116.

[3] Joseph G, "StegChat: A Synonym-Substitution Based Algorithm for Text Steganography", 2012.

[4] Krista B, "Linguistic steganography: survey, analysis, and robustness concerns for hiding information", Center for Education and Research in Information Assurance and Security, Purdue University, West Lafayette, IN 47907-2086.

[5] Masoud N, Ronak K, Mehdi H, "An introduction to steganography methods", World Applied Programming, Vol (1), No(3), August 2011. 191-195.

[6] Salomon D, "Data Privacy and Security", ISBN: 978-0-387-00311-5.

[7] Sherly A P and Amritha P P, "A Compressed Video Steganography using TPVD", International Journal of Database Management Systems (IJDMS) Vol.2, No.3, August 2010.

[8] ASCII, URL : https://en.wikipedia.org/wiki/ASCII.

[9] Letter of Credit, URL:http://www.investopedia.com/terms/l/ letterof credit.asp.

[10] OpenStego, URL:http://openstego.sourceforge.net.

[11] Steganography, URL : https://en.wikipedia.org/wiki/Steganography.

[12] WordNet Dictionary, URL : https://en.wikipedia.org/wiki/WordNet.