AN EMBEDDING TRAID-BIT METHOD TO IMPROVE THE PERFORMANCE OF ARABIC TEXT STEGANOGRAPHY

REEMA AHMED ABDALLA BIN THABIT

SCHOOL OF COMPUTING
UUM COLLEGE OF ARTS AND SCIENCES
UNIVERSITI UTARA MALAYSIA
2016

Permission to Use

In presenting this thesis in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the Universiti Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to:

Dean of Awang Had Salleh Graduate School of Arts and Sciences
UUM College of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok

Abstract

The enormous development in the utilization of the Internet has driven by continuous improvements in the region of security. The enhanced security techniques are applied to save the intellectual property. There are numerous sorts of security mechanisms. Steganography is the art and science of concealing secret information inside a cover media without drawing any suspicion to the eavesdropper so that the secret information can only be detected by its proposed recipient. This is done along with the other steganography methods such as image, audio, video, various text steganography methods that are being presented. The text is ideal for steganography due to its ubiquity. There are many steganography methods used several languages such as English, Chines and Arabic language to embed the hidden message in the cover text. Kashida, shifting point and sharp_edges are Arabic steganography methods with high capacity. However, kashida, shifting point and sharp_edges techniques have lack of capability to embed the hidden message into the cover text .This study proposed new method called Traid-bit method by integrating three several types of methods such us kashida, shifting point and sharp_edges to evaluate the proposed method in improving the performance of embedding process. The study presents the process design of proposed method including the algorithms and the system design. The study found that the evaluation of the proposed method provides good knowledge to steganographer to improve the performance of embedding process when the Arabic text steganography method is developed.

Keywords: Arabic Language, Text Steganography, Embedding Performance, Information Security.

Abstrak

Perkembangan pesat dalam penggunaan Internet telah didorong oleh peningkatan berterusan dalam perihal keselamatan di alam maya. Teknik-teknik keselamatan yang sedia ada dipertingkatkan lagi yang bertujuan untuk menjaga dan menyelamatkan harta intelek. Terdapat pelbagai jenis mekanisme keselamatan, dan Steganografi adalah salah satu daripadanya. Steganografi adalah suatu seni dan sains yang bertindak sebagai mekanisme untuk menyembunyikan maklumat rahsia di dalam sesebuah media tanpa menimbulkan sebarang kekeliruan supaya maklumat rahsia hanya dapat dikesan oleh penerima. Perkara ini dilakukan bersama-sama dengan kaedah steganografi yang lain seperti imej, audio, video, dan pelbagai kaedah teks steganografi yang ada. Teks sesuai untuk digunakan untuk steganografi kerana keleluasaannya. Terdapat banyak teknik steganografi digunakan beberapa bahasa seperti Inggeris, China dan Bahasa Arab untuk menyamarkan mesej yang tersembunyi dalam teks penutup. *Kashida*, *shifting point* dan *sharp_edges* adalah teknik steganografi Arab dengan kapasiti tinggi. Walau bagaimanapun, *kashida*, *shifting point* dan *sharp_edges* teknik mempunyai kekurangan prestasi untuk membenamkan mesej yang tersembunyi ke dalam teks penutup. Kajian ini mencadangkan kaedah baru dengan mengintegrasikan teknik teks steganografi Bahasa Arab untuk menilai kaedah yang dicadangkan dalam meningkatkan prestasi proses penerapan. Kemudian, kajian ini membentangkan reka bentuk proses kaedah yang dicadangkan termasuk algoritma dan reka bentuk sistem. Kajian ini mendapati bahawa penilaian kaedah yang dicadangkan menyediakan pengetahuan yang baik untuk steganographer dalam meningkatkan prestasi proses penerapan di dalam teknik teks steganografi Arabic dibangunkan.

Kajian ini membentangkan reka bentuk proses kaedah yang dicadangkan termasuk algoritma dan reka bentuk sistem. Kajian mendapati bahawa penilaian kaedah yang dicadangkan menyediakan pengetahuan yang baik untuk steganographer untuk meningkatkan prestasi proses menerapkan apabila kaedah teks steganografi Arab dibangunkan.

Keywords: Bahasa Arab, Teks Steganographi, Menerapkan Prestasi, Maklumat Keselamat.

Table of Contents

List of Tables

List of Figures

CHAPTER ONE
INTRODUCTION

1.1 Background of Study

Nowadays, our communications are hardly private anymore. Intruders in communication and technology can get information in a readable and understandable form of system. The information disclosed by Intruders to others, may be used to lunch attack or altered against the person or organization's want (Bhattacharyya, Banerjee, & Sanyal, 2011). Unauthorized people are increasingly interested in other people's conversations, especially with the Internet being an open system. Hence, added counter measures must be taken to ensure privacy rights. One of the solutions used to tackle this problem is Steganography.

Steganography is the art and science that deals with hiding of secret messages in order to protect the existence of the message being detected by human senses. It is also a sub-discipline of information hiding. It is often mistaken for cryptography, even though both are used to protect valuable information. The difference between them is that steganography is the study of hiding information to conceal its existence, while Cryptography is the art of cryptogram or secret writing, involving various methods or embedded technique to ensure the protection of message contents (Computing, Kumar, & Singh, 2015). This depicts that the use of steganography makes anyone looking at the object storing secret information not to expect the existence of a message in the object, thereby dismissing thoughts of decrypting the object.

Steganography originated from a Greek word which comprises of Steganos, meaning 'to cover' or 'secret', and Graptos which means writing or drawing. Hence, steganography literally means 'covered writing'. The art has been used centuries ago before it was known as steganography. An example of its ancient use: "*Greek history* *tells the tale of how Demeratus from Xerxes used the art to write a warning message to the Spartans against the invasion, using a wooden base of wax tablet which was carved and covered with a fresh layer of wax. During World War 2, information were written on a sheet of paper using the invisible ink, so that it seems like a plain sheet of paper if it happens to fall in the hands of the other party*" (Por, Ang, & Delina, 2008).From the examples above, the advantage of steganography is seen and could be related to archives whereby the existence of information will not be suspected, especially when the archives look like other normal files.

Steganography consist of two main focus categories; the linguistic steganography and technical steganography (Malik & Mitra, 2015). Technical steganography is a method of hiding information in digitally invisible codes and other mediums such as image, video, and audio. However linguistic steganography also known as natural language steganography. It is the art of using the natural language to conceal secret message that divided to be two main focus, those are linguistic steganography and text steganography. In Addition, the linguistic steganography entails encoding messages based on order linguistically altered or modified cover document. Meanwhile text steganography is based on manipulating lines, characters, spaces, or other features of the message.

Text steganography is said to be the most challenging approach in steganography. This is due to the fact that text file as the cover media normally possesses redundant data for hiding information in a very small quantity (Nasab & Shafiei, 2011). Thus, a method from text steganography category, is the main focus of the study. Text steganography methods are mostly applied to English texts. However, a few text steganography methods are applied to other languages (Khairullah, 2009). Only a few researches have been conducted on information hiding in Arabic texts (Odeh, Alzubi, Hani, & Elleithy, 2012). Therefore, the focus of this study will concentrate on Arabic text steganography and its method developed by researchers.

1.2 Motivation

There are several motivations in this study:

1. Arabic text has potential as can text manipulation as other languages such as Jawi, Persian or Urdu because they have same text scripting.
2. The features of the Arabic character set and the multiple character of Arabic letter can be exploited in term of sending secret messages by using steganography embedded techniques.
3. There is little research on Arabic language compared to English. Therefore, there is a lot of potential in utilization Arabic language in text steganography domain.
4. Arabic is the official language of Middle East countries. Therefore, developing steganography embedded techniques on Arabic text will enable those interested in sending and receiving the secret or private messages such as governments, organizations and individuals in secret and protection way against eavesdropping, modification and destruction.

1.3 Problem Statement

Most current embedded techniques used in Arabic text steganography have problems in security and robustness compared to capacity. Despite many researchers conducted in Arabic text steganography to enhance the capacity, the security and robustness problems are still occurring. For example, in the shifting point embedded technique the capacity is improved, but the performance issues such as retyping destruction and fixed output format are still in existence (; Al-Alwani, Mahfooz, & Gutub, 2007; Odeh, Alzubi, Hani, & Elleithy, 2012).

The current researches in kashida embedding embedded technique have improved the capacity of Arabic text steganography although the performance issue such as an increase in the number of kashida added per word that leads to ease the detected of the hidden message still exists (Gutub, Ghouti, & Amin, 2007 ; Al-Alwani et al., 2007 ; Al-Haidari et al., 2009 ; Al-Nazer & Gutub, 2009 ; Gutub et al., 2010; Alginahi, Kabir, & Tayan, 2012 ; Odeh, Elleithy, & Faezipour, 2013). Also, in sharp edges embedded technique the researchers' concern is to enhance the capacity by using little characters to embed the hidden message. However, the performance issues such as needing secret key to create sequence cod that used to embed the hidden bits, is not concerned in this matter (Roslan, Mahmod, & Udzir, 2011; Mersal, 2014).

As explained above, the main problem with these embedded techniques which have been discovered by previous researchers in the Arabic text embedded technique is the lack of performance in developing the text steganography embedded technique (Reddy et al., 2014; Sun et al., 2010). This problem can be formulated in the lack performance of the embedded technique used during embedding process. The previous researchers focus on the capacity enhancement while other researchers display their embedded technique that shows improvement and more useful than the previous embedded technique. It is also suggested by Sharma et al. (2013) that the steganography system requires an evaluation. This suggestion anticipates figuring out and reducing the problem of lack performance. In order to overcome the problem of lack of performance in embedding the hidden message this, study proposed new method based on integration of current embedded methods (kashida, shifting point and sharp_edges) in Arabic domain. Then the study will also evaluate the performance of proposed method on embedding process.

1.4 Research Questions

1. How to integrate the embedding methods based on kashida, shifting point and sharp edges embedded techniques in Arabic text domain?
2. How to evaluate the embedding performance of proposed method in Arabic text domain?

1.5 Research Objectives

1. To integrate the embedding methods based on kashida, shifting point and sharp edges embedded techniques in Arabic text domain.
2. To evaluate the embedding performance of a proposed method in Arabic text domain.

1.6 Research Scope

This study will only focus on three current methods (sharp_edges, shifting points and kashida) in Arabic text steganography. These Arabic text steganography methods has strengths in terms of capacity but the luck of performance still exist text steganography.

1.7 Significant of Study

The strength of this research is in the integration of several Arabic steganography methods to overcome the lack performance of these methods .The significant of this research is in the evaluation results of embedding process by using five evaluation metric to improve the performance of proposed method. Extracting knowledge from this document was beneficial for the Arabic steganography domain.

1.8 Outline

This chapter presented basis of area steganography in covering hidden message in text domain by explaining the background of study, problem statement, research question, research objectives, and research scope of this study. In general, this chapter illustrated about steganography area from what kind of area, categories, domain, until methods of this system which has been focused in text steganography that use Arabic text steganography embedded techniques. Then, there are phase researches in order to present the delivered objectives of the study.

- **Chapter One**: The background of study is introduced in this chapter. Then, the problem of study is represented to formulate the research question and research objective.

- **Chapter Two:** In this chapter, the current embedded methods of Arabic text steganography are reviewed. Then, analysis the kashid, shifting points and sharp_edges to identify the limitation and criteria of embedded methods used for improvement. It delivers summarization of literature and documented of problem statement.
- **Chapter Three:** In this chapter, the research methodology phases of the study are described in the concept and methods used in order to meet the objectives of the research mentioned in Chapter One. These include problems of Arabic text steganography in using the proposed method in order to integrate the embedded methods in the current Arabic text methods. The contribution of this research is to develop the Arabic text steganography method by using the integrated embedded methods of Arabic text steganography.
- **Chapter Four:** This chapter presents the integrated embedding embedded methods called triad-bit method. It introduced the process design in details by explain the flow process and the algorithms if embedding and extracting process. It, also figure out the system design of triad-bit method.
- **Chapter Five:** The result of experiment is evaluated based on parameter matrices. The evaluation justified result of performance of stego text from embedding process to become output data in this environment. This phase can be accorded the standard evaluation in Arabic text steganography.
- **Chapter Six:** This chapter revisits the objectives of this study with the related contribution to the steganography on the Arabic text domain. The chapter continues discussion with summarizes the main contribution of this

thesis by referring to the research objectives. This chapter finishes up the thesis by highlighting the limitation and future works of this research that gives the significance to the steganography community.

The phase that employed of this research based on domain, category, algorithm, evaluation, and any other requirement of Arabic text embedded technique in steganography. Then, for the following input in categories that used in steganography on text domain will be presented in chapter two.

CHAPTER TWO
LITERATURE REVIEW

2.1 Introduction

The study of information covering in any medium of data, and the concealing of all evidence so they cannot be detected by human sense is known as steganography. It is an important area in information hiding for securing of information to acquire privacy. Steganography can be used to hide information in mediums such as images, video, audio, and text. Text steganography is the most hard and best sort of steganography (Alla & Prasad, 2009).

This chapter presents survey of text steganography methods and its classification in previous researches. This chapter also reviews categories of natural language steganography based on method has been used by previous researchers focusing on Arabic language. The main objective of this chapter is about a survey of current methods on Arabic text steganography domain.

2.2 Related Work

In comparison to other mediums, text steganography is mostly utilized as it encrypts plaintexts and merges with cover text. This generates random character sequences. Due to lack of redundant information present in audio, image or video files, text steganography have come to be the trickiest embedded technique used. Text data storage is faster to read and uses less memory as well as easier communication. This makes it a preferable embedded technique as compared to the other types of steganography methods. Traditionally, three classifications can be used in describing text steganography as shown in Figure 2.1 such us random statistical generation, format based, and linguistic methods (Por & Delina, 2008).

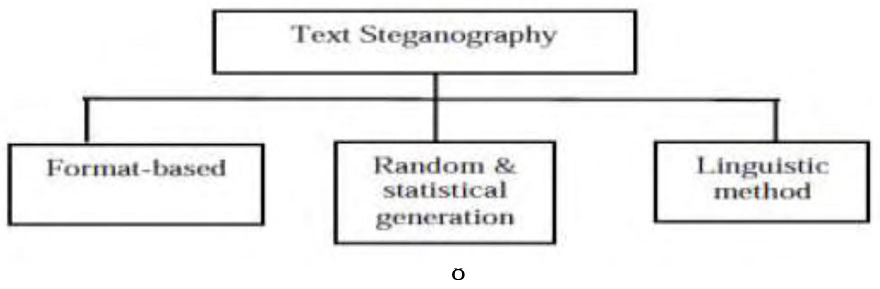


Figure 2.1: Three Basic Categories of Text Steganography (Por & Delina, 2008)

2.2.1 Format-based

This method uses physical text formatting to hide information. Generally, this method entails the modification of existing text to hide the steganographic text. Deliberate misspellings, resizing the fonts, and insertion of spaces, to mention a few, are format-based methods used in text steganography. However, according to Benneth, these format-based methods might trick the human eyes but cannot trick computer systems being used (Por & Delina, 2008).

One of the earlier embedded technique was proposed by Al-Alwani et al. (2007) which is utilize alphabets instead of pointed ones only.

Pointed alphabets with extension were used to hold the secret bit ‘one’, and un-pointed alphabets with extension were used to hold the secret bit ‘zero’. Though, it is important to note that letter extensions have no effect to the writing content.

As a matter of fact, the Arabic extension character used in electronic typing is viewed as a redundant character used only for format and arrangement purposes. The problem with the extension method is that not all alphabets can be extended using the extension character. This is as a result of the Arabic writing nature and their position in words.

The extensions can only be included in locations between connected alphabets of Arabic text; That is, an extension cannot be placed before alphabets at the beginning of words, or after alphabets at the end of words. Therefore, whenever an alphabet is found intentionally without an extension or cannot be extended, it is considered not to hold any secret bits.

Por and Delina (2008) have proposed a new method on information hiding by manipulating white spaces between paragraph and words. The proposed method provided additional capacity for hiding greater number of bits of data into a cover-text. The previous embedding strategy was applied to the space appearing between words. The major disadvantage of this approach was that it needed a large amount of space for encoding of few bits. A combination of this with inter-paragraph for hiding of secret bits, can effectively utilize most of the white spaces present in the text document. Hence, the study used inter-paragraph and inter-word spacing for information hiding.

Another method has represented by Memon et al. (2008), they proposed a embedded technique which depends on one of the features of the Arabic language known as Arabs. They features are Fatah, Kasra, and Damma. Fatha is a slash-like symbol written over the alphabet, whereas Kasra which is also a slash-like symbol, but in its case, used below the alphabet. Damma is a symbol resembling the number nine placed over the alphabet. These Arabs are not commonly used nowadays. They are acceptable in Urdu, but rarely used. In addition, they are applicable on all single characters of the Arabic language. In general Arabs are noteworthy in Urdu or the Arabic text.

Alla and Prasad (2009) have represented an enhanced embedded technique for Short Message Service (SMS) steganography, using SMS-texting language. The embedded technique is implemented by the removal of the static nature of abbreviated word lists or ‘word-abbreviation list’, and the introduction of computationally light weighted Exclusive-OR encryption known as XoR encryption for short. The dynamic positioning of ‘word-abbreviation list’, provides security of moderate level if used alone making it difficult for instant extractions of ones’ or zeros’ out of an SMS-text using a known algorithm, by an adversary.

Meanwhile and Khairullah (2009) has suggested another embedded technique of information hiding in Microsoft Word documents. He used the setting of foreground colors for invisible characters as like other visible characters. Hence, secret information can be hidden in invisible characters in the form of RGB values which bears no risk of exposure. This method has a very high capacity. Three bytes of data are hidden in every invisible character.

Again, the study by Roy and Manasmita (2011) introduced a method for information steganography in the English texts. This method uses the embedding of binary values into character texts. The secret message can be embedded bitwise by using an existing algorithm. The proposed algorithm embeds a single bit indicating the value of a secret message block which contains two bits. Hence, they concluded that by using the proposed algorithm, there was an increase in the storage capacity of a cover object having a fixed length in compares to the existing algorithm. This method can also be used for the prevention of illegal duplication and distribution of texts common with electronic texts, in addition to initiating secret communication. Other than electronic texts, this method can be applied to hard copy documents where the document is printed after hiding data in it. The hard copy document will have to be scanned to extract data and to reveal the embedded data using a computer.

Recently, Singh et al. (2012) have presented approach to text steganography through application of null spaces in cover messages for information hiding. In this method, extra null spaces are added to the plaintext of the cover file to hide bits of secret messages. The null spaces are used when the binary bit of the secret message equals 1 in the plaintext of each cover file. For secret messages with binary bit of 0, the null space remains unchanged.

2.2.1.1 Random and Statistical Generation

Random and statistical generationare generating cover text based on statistical properties. This method depends on words and character sequences. Information hiding within the character sequences is obtainable by embedding the information to appear in a random sequence of characters. The sequence appears to be random to a person who intercepts the message. Another approach to character generation is by taking statistical properties of letter frequency and word-length to create “words” (without lexical value) that will appear to possess same statistical properties as could be found in actual words in a given language. In information hiding within word sequences, bits of information per word, can be encoded using the actual dictionary items and a codebook of mappings in between bit sequences and lexical items, or the words themselves can be encoded to hide the information.

Furthermore, the study by Kataria et al. (2013) have proposed a embedded technique for text-based steganography applied in the English language text. This approach used simple X-OR Operation for enciphering of hidden messages. It also reorders the text using 8-bit random key for greater security. The proposed method is immune to reformatting ad retyping of text. The embedded technique generates a nonsense message that can be utilized in cloud computing, where it is only needed to store data in a large quantity. Hence, the method takes less time to encrypt the data. It also occupies less memory for data storage on the servers.

2.2.1.2 Linguistic Method

Linguistic steganography entails concealing secret information concerned with language of words and order linguistically modified words for message encoding. This section explains linguistic steganography that uses steganography embedded technique.

The linguistic steganography framework shown in Figure 2.2. Firstly, secret messages represented as a sequence of bits are hidden in a cover text.

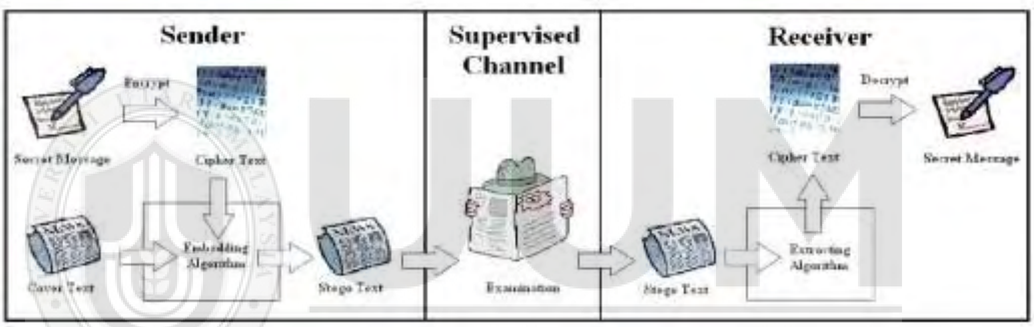


Figure 2.2: The Linguistic Steganography Framework (Chang & Clark,

This is done by using the embedding algorithm that results in the stego text. Next, the stego text proceeds to the human observer. The observer who happily passes the innocuous messages between the sender and the receiver, will examine the message content or text for suspicious looking contents. The algorithm is used to recover the hidden message when the stego text arrives at the receiver. All steganography systems have a fundamental tradeoff which is especially obvious in the Linguistic Steganography framework. This tradeoff is between payload and imperceptibility. The number of bits which can be encoded for every unit of cover medium (e.g. for every sentence in the linguistic case), is called payload. The tradeoff arises due to attempts to hide more information in the cover text via the application of additional linguistic transformations. This introduction of anomalies into the text is most likely to promote the chances of raising the observer's suspicion (Chang & Clark, 2010).

Linguistic methods consist of two groups. The first is the syntactic method, which depends on the use of punctuation signs for information hiding. The second category produces a synonym dictionary which is used in place of the interactive word (by some carrier file words), for the purpose of passing the hidden bits (Ammar et al, 2014).

i. Semantic (Synonym Substitution)

Most of embedded techniques that had been used were synonym substitution. Synonym substitution can be used in any language as long as the language the text of language has synonym word.

The research by Shirali-shahreza and Shirali-Shahreza (2006) has introduced the earlier embedded technique for steganography of information in Arabic and Persian texts. This embedded technique is founded on the existence of point in most of the Persian and Arabic alphabets. Based on this, changing of places of points was used in hiding information. This approach can be utilized in hidden exchange of information via text watermarking and text documents. The points can be displaced in vertical and horizontal directions, thus, hiding two bits of information in each letter. The amount of hidden information can be increased by combining the above embedded technique with other embedded techniques such as word shifting and line shifting. By utilizing a font editing software, a dynamical enabling of the program can be done producing the necessary fonts for information hiding to ensure that the output form of the characters or text is not homogenous. It also ensures that the input from conforms to the output form of the text.

Later Shirali-shahreza (2007) presented an approach to develop and privatize the abbreviation of text steganography method through SMS-Texting. The study examines the secret exchange of information through Internet chats. This embedded technique is not limited to a particular and special platform due to the fact that it is written in the Java programming language, and is capable of being implemented on any platform. For instance, medical texts can use the abbreviated equivalents of medical terminologies. Other than the in English, Internet chatting can be performed in other languages such as Greek, Arabic, and so on. Hence, the abbreviation text steganography approach, can also be used for information hiding in these languages.

Simultaneously, another similar approach was presented by Yuling et al. (2007). The algorithm comprises of three input sources. They are the secret message, natural language, and the key- and one output, the Stego-object. After the first scan, each word was recognized by the system and the set to which they belonged by generating lexical substitutions set, as well as variant forms of the same words. The lexical analyzer was then implemented on the Chinese language for the purpose of embedding the correct words in the carrier files, and also to take the context of each word into consideration.

Rafat (2009) has proposed an enhanced embedded technique for SMS steganography. The embedded technique used SMS-texting language, removing word-abbreviation lists of static nature, and introducing computationally light weighted XoR encryption. The dynamic arrangement of these 'word-abbreviation list' provides a moderate level of security which makes it difficult for adversaries to extract zeros' or ones' (if used alone), from SMS-text by having knowledge of the algorithm.

Furthermore, the study by Chang and Clark (2010) developed a embedded technique in which words are used as vertices in a graph. Here also, synonyms are joined or linked by edges, and vertex coding algorithms are used in determining bits assigned to words. In addition, the embedded technique considers words in a synonym transitive closure chain, and scores are assigned to each word by utilizing the proposed substitution checker. Low-score words are eliminated by applying a score threshold, i.e the rest of the words are in the synonym transitive closure chain, and are also acceptable to the context. A synonym graph is then constructed consisting of a vertex for each word remaining, and for every pair of synonyms, an undirected edge is constructed. After the synonym graph has been constructed, vertex coding method which was inspired by vertex coloring, was used for code assignment to each word in the graph. This approach ensures that every word stands for a unique sequence of bits, and ensures that a large number of synonym is not cut down. Hence, maintaining a reasonable embedding capacity.

Afterward, Prasad and Alla (2011) has presented an algorithm for synonyms used for information hiding in Bahasa Malaya language. In their study, the hidden algorithm consisted of two phases. The initial step converted hidden message to binary code, specifically using the ASCII codes. After that, synonym files were created with the sender and recipient having same word list for encryption and decryption of the message. A sender does not replace the word, if he intends to use a zero (0) in the text. Otherwise, the synonyms file is used to replace the word. The same strategy was repeated and applied to the whole of the secret message. The recipient decrypts the message by using an inverting strategy and then comparing to see if a replacement exists. In such cases, the secret code is 1.

Liu et al. (2013) proposed an information hiding algorithm used for text based on a substituted conception. The defect of this approach is that changing the meaning or context of the original semantics, is very easy. This causes the results from practical applications of the algorithm, to be of little significance. Hence, algorithms for synonym replacement must first of all, process an automatic segmentation. To create an algorithm for hidden information, the first step is by performing a word segmentation of the text containing the hidden information. The user determines the existence of the words in the thesaurus base, after traversing each word. If it exists in the thesaurus base, the location and number of the group of words are then embedded in the algorithm coding and decoding. This is done to obtain a result of binary strings, and the steps are repeated until the end of the algorithm.

Eventually, Odeh (2015) proposed an algorithms which employs text file as a carrier file. The (proposed) model conceals secret data in text files by applying varying properties and special symbols into the file font format or text file. In addition, the model can be applied using both ASCII code and Unicode, regardless of the text file format.

ii. Syntactic Transformations

The second which is the most commonly used manipulation method for linguistic steganography is syntactic transformation. This approach is based on the fact that a transformation can be applied to a sentence in more than one semantically equivalent syntactic structure, via transformations such as clefting, passivation, and topicalization. The first or initial syntactic transformation method was presented by (Atallah et al., 2000).

Shirali-Shahreza (2008) have created a text steganography embedded technique for hiding data in the English texts. This embedded technique is based on substituting UK and US spellings of words. US and UK have different spellings for some words in English. For example the word “program” has different form of spelling in US (program) and UK (programme). By using this characteristic, they proposed a embedded technique for hiding data in an English text. The embedded technique conceals data in a text by substituting or exchanging such words. The embedded technique can be applied on electronic documents as well as on printed texts. Printing the electronic document does not destroy the hidden data. They argued that their method is new, hence, the possibility of breaking it is low. The proposed embedded technique can also be utilized in other devices such as PDAs, Pocket PC, and mobile phones. For instance, the approach can effectively be applied on Short Message Service (SMS). SMS is a very popular service on mobile phones that entails the exchange and transfer of short text messages (i.e SMS), between mobile phones.

The study by Changder et al. (2010) introduced a embedded technique for text steganography via the Indian Language which considers the features of a sentence such as number of characters, number of vowels, number of words, etc. The presence of redundant feature code able characters present in the Indian Language, could be used in concealing the message into a cover file containing some Indian texts. For this to be done, first the messages are divided into a number of blocks with sizes equal to the number of properties. Next the secret message is hidden by applying a comparison of properties represented by the bits, to properties followed by the sentence. This method also confers the extraction of message or information from generated cover files by employing the reverse method of hiding. The application of the proposed embedded technique to topics in the daily newspaper in Indian Languages like Bengali, satisfactory results were obtained with unnoticeable variance between the program generated cover file and the original text file.

At that time, Chang and Clark (2010) has developed an automatic system utilized in the checking of fluency and grammaticality of paraphrases in context. They also proposed using paraphrases as an efficient transformation for Linguistic Steganography. The merits of this approach is that it is domain and language independent, hence, needs only a Google n-gram corpus and paraphrase dictionary, both of which might be available in a variety of languages in the future.

Kamel and Banawan (2012) proposed a strategy for hiding of secret messages in the common English text. The strategy exploits the redundancy or repetition existing in the different locations of maneuverable words, ensuring not to change the meaning and grammar (or the integrity) of the sentence. Their study demonstrated the feasibility of automating the variant transforms. The study has also shown that these transforms are commonly found in English text, using Google Books Corpus. Since the transforms are conducted at different levels of granularity and on varying sentence parts, they are independent, thereby, encoding each sentence by more than one transform.

Another study by Wilson et al. (2014), presented a embedded technique for information hiding in tweets. It consists of two types of manipulation in a novel combination. One of the types of manipulations is by the use of language models, which is computed automatically. The other is human involvement for selection of the `optimal' stego tweet from a list or record of possible paraphrases of the cover. Their study reported to have found the NL transformations through unilingual translation, using a large, multi-word paraphrase database. The combination has proven to be more sophisticated than previously associated in the NL steganography literature. This allowed 4 bits to be hidden in usable tweets, where usable tweets stood for approximately 40% of their corpus. This represents a small amount of information, but is comparable to other existing NL steganography embedded techniques that typically manages one or two bits per sentence.

Simultaneously, Ammar et al. (2014) conducted a study and introduced an algorithm that hides data in document files of any and many languages using word symbols. The method employed Remarks (Left Remark, Right Remark, ZWJ, and ZWNJ) symbols, which can be utilized in any language. It can also be utilized at any position within the words. These can enable a user to pass 4 bits in any two letters or alphabets. In addition, the algorithm was enhanced using the stego key to produce symbols table representation.

Recently, Gaur and Sharma (2015) proposed an approach which consists of simple addition, multiplication, and subtraction of ASCII code digits on each character of the cover text. The newly produced numeric values are utilized in the encrypting of ASCII values of plain text. The approach consists of three basic arithmetic operations which are performed on each and every character of cover text. Hence, each character generated three numeric values after the arithmetic calculations were performed, such that every numeric value encrypted two ASCII values of plain text parallel. The first from the beginning of array of ASCII values of plain text, and the other, from the ending of the same array. Hence, one character of cover text conceals at most six (6) characters of plain text. Therefore, the execution time and the memory allocation problem common with cover text are reduced. This enhanced the used of parallelism performance in their approach.

Most of studies divided the Natural Language steganography into two views; Linguistic Steganography and Text Steganography. The Linguistic Steganography entails encoding messages based on order linguistically altered or modified cover document. While Text Steganography is based on manipulating lines, characters, spaces, or other features of the message (Malik & Mitra, 2015).

2.2.2 Natural Language steganography

This section presents a categorization of natural language steganography for information hiding. Natural language steganography for concealing of messages in medium of text, can be generally categorized into two main categories: The categories are text steganography and linguistic steganography. Text steganography conceals information by manipulating components in text which comprises of feature in word, line, space, and others characters in a sentence of text. Whereas linguistic steganography conceals the information by modifying the information or encoding the massage based on linguistic order (Din & Samsudin, 2011). The categories of natural language steganography are presented in Figure 2.3.

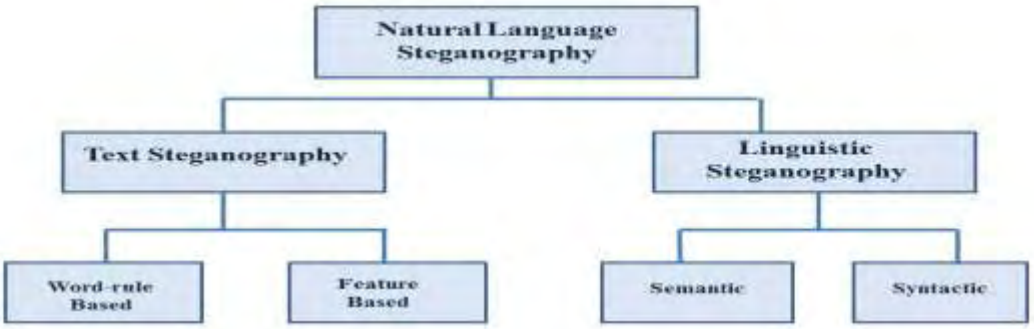


Figure 2.3. Natural Language Steganography Categories Classification

Text steganography embedded techniques which are classified into two categories. The first category explains word-rules consisting of two sections: word shift-coding and line-shift coding. The second category is known as feature-based embedded technique in text steganography. This category is divided into sections based on unique characteristics of the embedded technique, that is, language based and letter based.

a. Word-Rule Based

Word-rule based is divided into two parts or methods to hide information. The first part is line shift coding, which is a unique form of concealing embedded message in the text line by shifted vertical route.

b. Feature Based

Feature based approach alters the features by manipulating the shape, size, and position which relates to the features and structures of the text font. This embedded technique prevents the reader from recognizing the secret message or information in the text (Roy & Manamisti, 2011). The characteristics of this embedded technique encourage the use by many researchers studying languages all over the world. The feature based embedded technique is also used on website text. Figure 2.4 illustrates the classification of the feature based embedded technique.

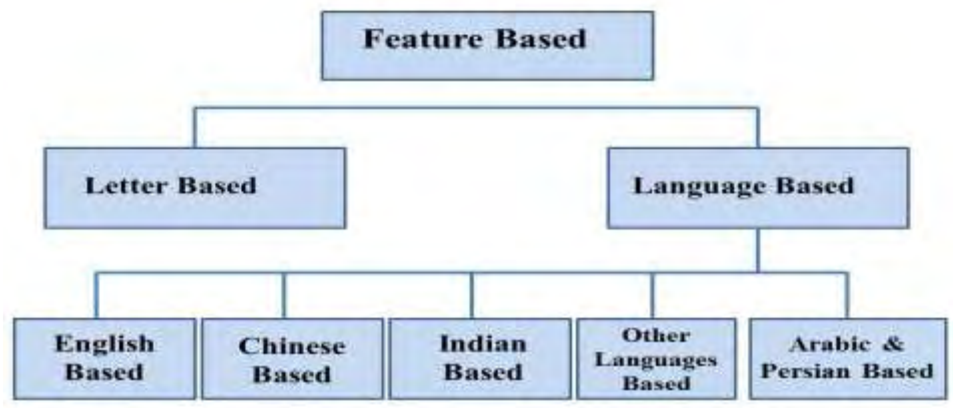


Figure 2.4: Feature Based Categories

i. Letter Based

Letter that uses the alphabets A to Z can be used in other languages in this world. Hence, the feature based embedded technique can hide information in the alphabets A to Z known as letter based. Some researches on the feature based are:

Battacharya et al. (2011) proposed a method of changing alphabet letter patterns, in order to hide information by embedding them in some characters of the English alphabet. Hidden messages are embedded based on the binary bit sequence with alphabets such as i and j. The alphabet I and j have the pointed features and are embedded in the 0 bits, while the 1 bits are embedded in characters like a, c and A. This approach generates bits for hidden messages by inserting message bits based on those characters using the embedding algorithm as the stego text. Batthacarya et al. (2011) suggested that the approach can be extended for use in other Indian languages.

Meanwhile, Dulera et al. (2011) developed a method for concealing of characters which combines feature based and random character sequences based on characteristics of the English language that converts the message into binary bits.

The approach constitutes three (3) methods that divide binary bits into groups. The first group is those based on curves of the alphabets. Here, alphabets with curves are used for hiding the 0 bit, and the alphabets without curve, for hiding the 1 bit. The second group is based on vertical lines in alphabets. Alphabets with vertical lines hide the 1 bit and the other alphabets hide the 0 bit. The last method is the quadruple which further divides the alphabets into four (4) groups such as, curved alphabets used for 00 bit, alphabets with middle horizontal line for 01bit, alphabets with one vertical line for 10 bit, and the alphabets with diagonal line for 11 bit. These methods cannot be decoded and are hardly noticed by the user. The method also has resistance to retyping and reformatting of the text. The drawback of this approach is that it is easily attacked if the application is detected. Formal security analysis, especially from this embedded technique takes care of syntax, sequence sentence, and paragraph. Future analysis can use this approach in linguistic steganography.

Letter based can also be utilized in web based for the purpose of hiding secret information in text. Mahato et al. (2013) proposed a embedded technique for text steganography which conceals the information in back end interface web page, from the source code based on attributes and HTML tags. To hide the information, the embedded technique used binary bits 1 for two sequence attributes of the same tags, and binary bit 0 for two sequence attributes of varying tags. The aim of the embedded technique is for web pages to be transmitted easily and safely via the Internet to the destination. This is possible because the hidden messages exist in the source code. Another advantage of this embedded technique is the ability to achieve the highest capacity. For future work, probably the embedded technique can be extended to others similar format such as XML, CSS, etc. In addition, a decrease in the time complexity can be of concern in a future research.

One of the recent study by Kumar et al. (2014) developed a text steganography algorithm based on email sender image address. The approach increases the concealing volume with compression ratio, which combines Burrows Wheeler Transform (BMT) + Move to Front (MTF) encoding + LWZ coding. The approach also includes numerous random alphabets before the '@' symbol of email ids in order to magnify randomness. The method is divided into two steps. The first step is the embedding step were construction of other matrix D in sequence is performed to choose vital text from hidden messages. Here, the BMT reorganizes the alphabet (A-Z), MTF strengthens the correlation between the elements, and LWZ updates the dictionary in each and every single string or symbol. The second step is the extraction step, where the stego text is extracted from four (4) binary bits, and transformed or decoded using the MTF and the BMT for obtaining the original text. The strategy performance is quite good in concealing information of a large capacity in email environments. Though, the randomness of this embedded technique is of concern because random elements in email addresses, influences the security of the approach.

ii. Language Based

The unique characteristics of the feature based embedded technique makes it easy to be used in any language, either with figures or alphabets. Feature base embedded technique can be used in several languages such as:

A. English Based

Previously, Sui and Luo (2004) introduced the modification of written status of the markup letter. This was used to analyze the hidden secret information or message in hypertext. Markup letters determine the secret bits used to reveal distance of hidden messages. The embedded technique embedded the files without unwanted symbols in the secret information. However, steganography based on hypertext is more difficult than other steganography embedded techniques (image, video, etc.). Also, the theoretical research and application of this embedded technique is still limited. Therefore, the future role of system in information security is of concern to researchers.

The study by Stutsman et al., (2006), proposed an approach for feature text-based steganography employing machine translation for the hiding of messages in natural language text. This embedded technique translates the transmitted text, allowing the source to be kept in its original form. In the process of generating this embedded technique, key ideas are used by both the sender and the receiver to conceal information behind translation-based steganography in the natural language translation. The prototype used in this embedded technique, attained roughly 0.33% in protocol steganographic. Hence, there is a need to increase the capacity for the modification encoding scheme.

Later, Banerjee et al. (2011) produced a new embedded technique for code representation known as secret steganography code for embedding. The approach positioned vowels and consonants according to grammatical sequence for cover text. The embedded technique also puts hidden messages in indefinite articles for embedding and mapping information from non-specific English language. Two binary bits were used per word, and then divided into two side windows: the sender side and the receiver side. On the sender side, the cover message and the text message were selected to be embedded in text encryption. The cover text selected, will be hidden in the encrypted message and generated to be the stego text. The receiver side should be able to extract stego text messages from the sender side to reveal the cover information. This embedded technique generates a stego text that cannot be revealed by the people and lowest degradation. Also, the embedded technique has a high security level, because the encoding of hidden messages increases security levels. The embedding capacity of this approach is also increased because it uses two bits per words, even though the embedding capacity is based on sequence and cover text patterns.

Odeh et al. (2013) proposed a text steganography embedded technique that uses the Right-to-Left Remark and the Left-to-Right remark, to conceal information. The approach didn’t change the text format and applied it to other languages based on the ASCII or UNICODE coding. Using this embedded technique, information is hidden by converting into binary bits. While 00 bits add nothing to the text, 01 bits add to Right-to Left Remark (U200F). Likewise, 10 bits are used for Left-to-Right Remark (U200E), and 11 bits for the combination of U200F and U200E. This embedded technique hides the secret data or message, without changing the file’s information. It also avoids the retyping problem by converting the file into PDF format. The optimization embedded techniques suggested by Odeh et al. (2013)

utilizes a very large amount of data and also reduces the file size. Feature based embedded technique s that use ASCII characters, were also utilized by other researchers.

Kataria et al. (2013) produced a text steganography approach for Encryption with Cover Text and Reordering (ECR) using ExOR operation which can be generated very fast for the encryption and decryption process.

It also merges two characters which are hard to fetch when enciphered in text, or the original message. The embedded technique reorders eight (8) bit random keys, with the 0 bits describing the cover text, and the 1 bit describing encrypted text. According to process encrypted and reordered, the embedded technique hides much number of bytes, with small cover text and lesser time is required when generating the process.

The embedded technique creates the non-sense message that requires less time to encrypt, hence, large data that can be utilized in cloud computing

Odeh et al. (2014) introduced an algorithm that utilizes several invisible character symbols for covering of four bits between alphabets in word symbols such as left remark, right remark, and zero width joiner. Zero width joiner inserts table variation in steganography carrier file data. The algorithm utilized the stego key and input carrier in creating symbol tables, and binary representation to embed hidden bits in the text. The advantage of using this embedded technique is that the algorithm is not only suitable for Unicode languages, but also for ASCII codes. Other advantages are that it does not require modification of file format, and does not need special equipment to conceal data or information. The algorithm can only be applied to specific languages; hence, there is a need to extend the configuration embedded techniques to enable application in any language.

Again, Reddy et al. (2014), presented a Deoxyribonucleic Acid (DNA) steganography, consisting of four alphabets represented by binary bit 00, C for binary bits 10, G for binary bits 01, and T for binary bits 11. The approach generated DNA structures, which is transformed into DNA sequence, and then into chipper text using look-up tables. Hidden messages are decrypted using step received text, rule sequence table, lookup tables, and DNA sequences. Hidden messages from DNA sequence are then embedded in binary bits according to represented binary on lookup tables. The limitation of this embedded technique is that only ASCII characters can be used. Therefore, it requires some work for Unicode characters to be used.

B. Chinese Based

Zhang et al. (2006) has proposed a embedded technique in the Chinese alphabets by rearranging the sizes of the components of the rectangular region, to embed signals into characters. The embedded technique uses a simple algorithm based on the content, and explores the embedded technique according to mathematical expressions and automatic generation.

The process of embedding information is without distortion of the integrity and aesthetic feeling of the Chinese alphabets. The advantage of this approach is in the transparency and robustness to secure the Chinese alphabet. It can also be applied in copyright protection, secret transmissions, and other forms of information hiding.

In other words, the approach plays an important role in securing text for the Chinese alphabets.

This embedded technique has proven to be more efficient than others in concealing of information. Although, its drawback is having a low information watermarking, thereby, requiring a secret key to extract watermarking. Hence, deleting the watermarking without the key, can be a difficult process.

The study by Sun et al. (2010), introduced a text steganography approach in the Chinese text alphabet. They presented two embedded technique s in process embedding feature in the Chinese text. The embedded technique s are the high efficient substitution embedded technique (HESM), and the simple substitution embedded technique (SSM). SSM embeds the hidden message by encrypting the binary bit sequence in the hidden message.

The features of the Chinese text correspond in bit sequence, which can be changed to traditional form corresponding to the bit 1, and the remaining unchanged features correspond to the bit 0. Given the stego message based on the bit, the features of the 0 bit and 1 bit are extracted if they appear in the substitution dictionary. HESM also used characteristics in the substitution dictionary like SSM embedded technique which can embed 2 or more bits in each substitution.

However, the security level is still in the simple phase to avoid traditional attacks such as rewrite text and semantic analysis. Hidden messages need to be encrypted before embedding and SD only chose traditional form and simply characters to avoid.

C. Indian Based

Changder et al. (2009) proposed an approach in Hindi text which considers the structure words by shifting the specific matra (media vowel representation) according to free spaces to left and right, in order to hide the secret information or message in the text. The embedded technique compresses the secret or hidden message, and translates the length of the compressed message to binary bits containing the matra. Even though retyping will lose the hidden information, this embedded technique can be used in a large amount to hide information. Also, it is not easily detected or revealed by recognition program (OCR). This steganography embedded technique, may be applied to writing styles which are similar to the Hindi script.

Furthermore, Phatak (2010) proposed Numerical Code Text Steganography, also in Hindi character or other similar Indian languages. This approach creates all the vowel and consonant alphabets in binary bits made up of four numbers or bits. All the vowels and consonants of the Hindi alphabets are encoded to a specific Numerical code based on the Indian theology. Every alphabet in this approach is encoded based on the four binary bits representation. Consideration is given to the first letter to decode or reveal the information by chipper phrases. Security of features, capacities, and robustness, are the three (3) principle steganography that can establish the secret or hidden messages and are vital in changing of the hidden information in text. However, the embedded technique takes a great deal of time to reveal the hidden information. Also, not all Hindi characters can use this embedded technique. This means that only specific word in Hindi alphabets can be used.

Although, it might be possible for other scripts which are similar to the Hindi text to utilize the embedded technique.

Ghosh and Debnath (2010) have proposed an approach that applies concept binary string to hidden message. The Indian language uses the inverse of this embedded technique to cover file. The hidden message used longest common subsequence to hide the messages with minimal modification of the alphabet features. This embedded technique (feature based) selects the longest common subsequence to extract the real message by using binary bits to substitute the matched alphabet based on format Indian language from cover file. This embedded technique is important for generating hidden messages in large volume, although, it is not easy to use because it has to deliberate the sentences.

In addition to numerical, feature based embedded technique in Hindi text uses grammar. Das and Ghosh (2010) proposed a feature based embedded technique in flexible grammar feature in Indian Languages for the hiding of messages in able character. This embedded technique covers the medium used in transforming the version after substituting the feature code. The embedded technique encodes a binary string with the Finite State Machine (FSM) in order to define transition functions as well as transformable symbols in each and every category. To encode the binary 0 bit, the next transformable symbol is selected from the same category, otherwise selection of the next transformable symbol does not same and apply the transformation. After that, the symbol is transformed by reconstructing binary strings in the decoding phase, which uses symmetric reverse of encoding outcome. For future work, there is a need to develop modification algorithms to achieve a decrease in iteration number, thereby reducing the cover file volume, as well as the time algorithm complex.

Using a language similar to the Hindi language, Alam and Naser (2013) showed features script for Bangla text using chain code. Chain code is used in translating codes into number signified contour border pixel directions, where one pixel to the next pixel, converts a part of the chain which is afterwards concatenated to form a chain. This approach represents the 50 feature vector of Bangla alphabets or characters which employed chain code by taking a bounding box (smallest frame or contour) divided into 7x7 blocks, and calculating each block into local histogram consisting of chain codes. However, the use of chain codes do not give vital variation in figuring boundary characters, because of the similarity in shape and lack of robust features in Bangla script.

D. Other Languages Based

1. Polish Based

The study by Khan and Abhijitha (2015) has proposed an unprecedented steganography approach which is applicable to the Polish language electronic inscription. This approach comes from the ascription of possessing points more than the partial size of the text alphabets. The hidden information in the form of bit 'one', is stored in the pointed alphabets, and the secret information in the form of bit 'zero' is stored in the un-pointed alphabets. The data is required to fit according to the cover-text alphabets. Hence, all the alphabets do not carry secret bits or information. Surplus Polish extension characters are utilized beside the alphabets for jotting certain alphabets clutching the cloistered secret bits. The advantage of alphabet extension is that there is no inclination towards the inscription content.

2. Thai Based

The study by Samphaiboon and Dailey (2008) has proposed a blind steganography strategy for the Thai text that exploits redundancies in the way TIS-620 signifies compound alphabets, merging vowels and diacritical symbols. They found that the alterations made when information bits are inserted in the carrier text are not noticeable by casual observers. They also found out that the embedding capacity of 0.22%, makes the strategy an effective and practical embedded technique for covert communication.

3. Czech Based

Khan et al. (2015) they presented an unaccustomed steganography approach which is applicative for the Czech language electronic superscription. This approach aids from the feature of having points more than the partial size of the text letters. The secret bit ‘0’ is stored in the un-pointed letters and the secluded information bit ‘1’ is stored in the pointed letters. The data needs to be in symmetry to the cover-text letters, ergo all the letters do not carry secret bits. Excess Czech extension characters are used nearby the letters for jotting the specific letters clenching the hidden secret bits. The advantage of letter extension is, there is no proclivity towards the inscription content. During the concealed exchange of the information by substantiate stealthy communication and text documents, and the integral aspects of the steganography are essential, such as the reliability, robustness, security.

2.2.3 Arabic and Persian Based

A few studies have been conducted on the hiding of information in the Arabic texts. The following is a list of various methods used by studies for utilizing the Arabic text, and reported so far as shown in Figure 2.5.

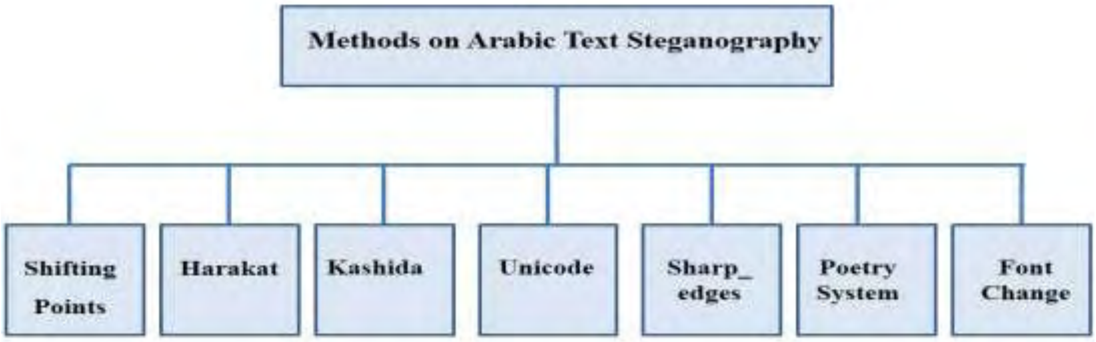


Figure 2.5 Embedded methods on Arabic Text Steganography

2.2.3.1 Shifting Points Embedded Method

In Arabic languages, the dot is very important. Fourteen (14) of twenty eight (28) Arabic alphabets have one or more dots. In this approach, data are hidden in the Arabic texts using these alphabets as will be explained later .By displacing the points vertically, it is possible to hide information in the texts as shown in Figure 2.6. Though, a special font developed solely for this purpose is used.

Shirali-shahreza and Shirali-Shahreza (2006) has introduced the earlier approach for steganography of information in Persian and Arabic texts. This approach depends on the presence of points in majority of the Persian and Arabic alphabets. According to this, the place of points were changed to hide information. This approach can be utilized in hidden exchange of information via text watermarking and text documents. In addition to using this embedded technique for electronic texts, it can be utilized on hard copy documents. Also, the points can be displaced in both horizontal and vertical direction as well, thereby allowing two bits of information to be hidden in each alphabet. By utilizing a software for font editing, the program can be supported dynamically to generate necessary fonts for information hiding so that the text output is not homogenous and conforms to the text input.

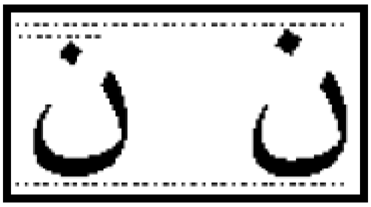
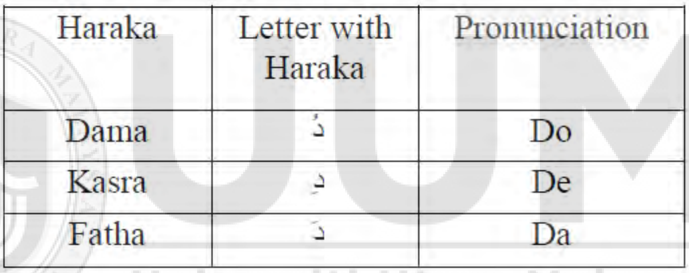


Figure 2.6: Vertical Shifting Point (Shirali-shahreza & Shirali-Shahreza, 2006)

Another embedded technique has suggested by Odeh et al. (2012) in text Steganography for Arabic multipoint letters. The algorithm deals with two (2) bits for every multipoint alphabet. Their embedded technique was combined with vertical point shifting in order to improve the amount or quantity of the hidden data. A challenging problem for algorithms similar to this is the retyping process which removes all the hidden data. Their study proposed a solution to this challenge to reduce any new font format changes. This was achieved by unifying all the data leading to a homogenous file.

2.2.3.2 Harakat Embedded Method

Arabic language uses varying marks or diacritics (Arabic extension character) known as Harakat which has an optional use. The main use of these symbols is to differentiate between words that have same alphabets (i.e., so that each word is pronounced differently). The diacritic ‘Fatha’ is used to hide the bit 1. The rest of the diacritics are used in hiding a 0 due to the fact that ‘Fatha’ stands for almost half of the diacritics in Arabic texts as explained below in Figure 2.7. The merit of this approach is that it attracts the reader’s attention.



Haraka	Letter with Haraka	Pronunciation
Dama	دُ	Do
Kasra	دِ	De
Fatha	دَ	Da

Figure 2.7: Some Letters with Mark and Their Pronunciation (Odeh, 2012)

Aabed et al. (2007) have presented an approach that makes use of eight (8) varying diacritical symbols in Arabic for the purpose of hiding binary bits in the original cover medium. The embedded information is extracted and revealed by reading the diacritics from the text, and converting it back to a binary representation. Arabic texts which are fully discretized, are utilized as cover media.

The first bit of the hidden data is compared with the initial diacritic in the cover media. For instance, if the first hidden bit is one (1) and the initial diacritic is a ‘fatha’, the diacritic remains on the cover media. Then an incrementation of the index for both the cover media and the embedded text is performed. However, if the first diacritic was not a ‘fatha’, it is taken out of the cover media. Again, an incrementation of index for the cover media is performed to explore the next diacritic. A repetition of the approach is done until the next ‘fatha’ is realised. A secret 0-bit is used or embedded in a similar approach except it uses the remaining seven diacritics with the exemption of the ‘fatha’.

Memon et al. (2008) have proposed another text steganography embedded techniques in the Arabic and Urdu alphabets based on the Arabic language characteristic of some alphabets having slash-like symbol (Fatah, Kasra, and Damma). The approach employed reversed Fatah to represent the hidden message in the text. From the article written in the Arabic language, the hidden message is read and matched character by character to the original article, and then the reverse Fatah is embedded in different lines to hide the message. The approach is a fast and good approach for deceiving the human vision because the reverse Fatah doesn’t seem clear and can be used in hiding large volumes of information. The disadvantage of this embedded technique is that the text can be lost during retyping and also, only one font possess a static frame. This embedded technique can be applied in other similar script such as Pakistan and Urdu. Perhaps its use can be considered in Asian script.

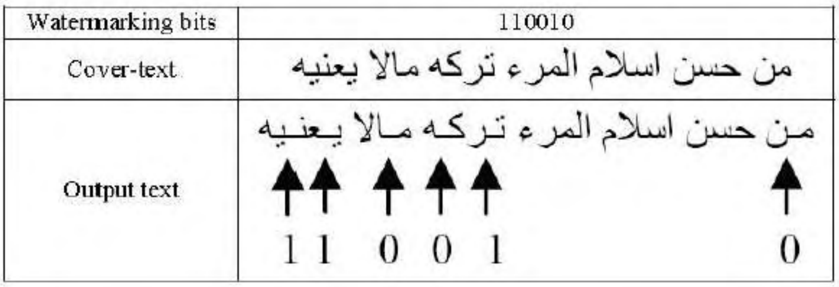
Bensaad and Yagoubi (2011) described a embedded technique for information hiding in a vocalized Arabic text. In the implementation of the approach, the first step is to ensure that all possible diacritics exist in the cover text. Next, every diacritic is sequentially matched to a bit from the secret bits. Finally, if the secret bit is 1 then the diacritic is presented as it is, otherwise (if it is 0), the diacritic is removed.

In this study, Gutub et al. (2010) proposed two steganography algorithms for the Arabic text. The algorithms were designed based on a novel concept of wasting/non-wasting property of the Arabic diacritics. In the first algorithm, a fixed size block parsing is used. A stream of binary bits is parsed into cover blocks of fixed amount of size. The second algorithm uses the variable-size content-based. Here, binary data is parsed into an integer number of blocks irrespective of the amount of bits they possess. The proposed algorithms have different properties and are thereby suited for various application types, as well as steganography requirements (i.e. robustness, file size and capacity). In contrast to the content-based algorithm, the fixed-size algorithm permits a straightforward computation of the required quantity of cover text but cannot instantly predict the file size of the output. The results reported shows that a block size of 4-bit is preferred for a small size of the corresponding message file. Conversely, the content-based algorithm is desired for minimal overhead. In regards to the percentage increase in cover size, the most preferred is 1-bit block because it acts most secretly.

Eventually, Ahmadoh (2015) designed an Arabic text steganography embedded technique by means of utilizing two (2) Diacritics (Kasrah and Fathah) for the purpose of information hiding as a novel version refining the single (only Fathah) Diacritic approach used before. This use of Diacritics (Harakat) for security purposes is conceived as an important approach for text written originally with Diacritics such as historical books or religious Arabic. The proposed algorithm is initiated by fragmenting the hidden message into two (2) arrays of binary values, forming odd and even lists. The general idea is that the odd array list are hidden in the Fathah diacritics while the even array list, are concealed in the Kasrah Diacritics. This fragmentation is done to add extra security to the proposed approach when compared to the Fathaha only or single Diacritic approach. The first odd bit of the hidden message is read by the program and then comparisms are made with the initial Fathah in the cover text. For instance, if the initial odd bit to be concealed was a ‘one’, the initial Fathah will not be touched. Else, if the initial odd bit to be concealed was not a ‘one’, the Fathah will be removed. A repetition of the process will be performed, until every hidden bit in the odd array are examined. In the same way, the even bits array is read by the program and its’ effect is applied to the Kasrah’s existence or disappearance. The algorithm is programmed using PHP language with UTF-8 Encoding. This is because of its ease in fully supporting the use of Arabic text.

2.2.3.3 Kashida Embedded Method

In this embedded technique, the extension (Kashida in Arabic) was added to words to represent secret bit ‘one’, and are not added to words without the extension (Kashida) to represent the secret bit ‘zero’. It is worth noting that alphabet extensions have no effect on the writing content or the message content as shown in Figure 2.8. The main drawback of this approach is that it captures the reader’s attention, increases file size, and changes the text apparent look.



Watermarking bits	110010
Cover-text	من حسن اسلام المرء تركه مالا يعنيه
Output text	مـن حسن اسلام المرء تـركـه مـالا يـعنـيه ↑↑ ↑ ↑↑ ↑ 1 1 0 0 1 0

Figure 2.8 Steganography Example Adding Extensions after Letters (M. Shirali-Shahreza & Shirali-Shahreza, 2008a)

Alnazer and Gutub (2009) performed a study based on characteristics of Arabic alphabets and how Kashida affect the looks of the Arabic alphabets. The results of the study answered certain questions like, “Is it appropriate to utilize Kashida whenever possible?”, and also questions like, “In an Arabic text, how many places can Kashida be embedded. A new system was developed based on the proposed approach. The embedded technique enables the embedding of a chosen secret after it has been converted to bit representation. The result is presented as a cover media of text with a secret inside it. The system is capable of reading the cover media (or medium) and extracting the secret which is represented in bits. The bits are the converted to reveal the secret in its original format.

The study by Al-Haidari et al. (2009) proposed an algorithm for text steganography in the Arabic language. Three scenarios of the proposed approach were evaluated, that is, by using at most 1, 2, or 3 Kashidas for each word, with varying cover text media sizes such as 70627, 131233, 235125 and 525271 bytes. Information and files were concealed in the cover text documents using the described algorithm. The major idea is to utilize the positions of possible extendable characters in a given word (in the cover text media), to conceal secret data bits or hide messages. The secret data are denoted within a word, by introducing Kashidas after some extendable alphabets in the word.

Meanwhile The study by Alginahi et al. (2012), used an approach which encodes the original text document using Kashida based on a particular key generated prior to the process of encoding. The key comprises of 48 bits represented by 011000101001010101010111000010100111010010010110. Using this approach, the Kashida is positioned before the following alphabets: ا، أ، إ، آ، ئ، د، ذ، ر، ز، و، ؤ. As mentioned earlier, a Kashida can not be placed after these alphabets with the exception of the alphabet ئ. Here, a Kashida is positioned before the alphabets when they are encountered. This means that the watermarking key carries a bit 1, hence, a Kashida is not positioned before these alphabets. Otherwise, a Kashida is positioned before the alphabets if the key carries a bit 0. It is water marking

2.2.3.4 Unicode Embedded Method

Arabic alphabets possess many forms based on the unicode standard. The unicode standard consists of two groups of codes for the Arabic alphabets. The first is the representative code, and the second is the code of the possible shapes of the letter. The approach utilizes the various possible unicode values of the same alphabet to conceal the bits. It is suitable for use on the internet and modern devices such as smart phones. Although, it is prone to traditional attacks. Some Unicode based steganography embedded techniques, provides low security and high capacity, or low capacity and high security.

The study by Shirali-Shahreza and Shirali-Shahreza (2008) designed a embedded technique for Steganography in the Persian and Arabic texts. The embedded technique deployed the unicode Standard and the special characteristic of the Persian and Arabic languages which is having different shapes for the alphabets. This embedded technique does not depend on any special format and saves the stego text in different formats such as Microsoft Word documents, HTML pages, or even plain text format. Each word in the text is saved using the representative alphabets. Although, the words can also be saved using codes representing the correct shapes of each letter (according to its position in the word).In concealing the bit 0 in the word, the word is saved by using the first option. Likewise in concealing the bit 1 in the word, the embedded technique uses the second option. This embedded technique conceals one bit per word in the text file. Therefore, the use of the embedded technique generates a high capacity.

Again, Shirali-Shahreza and Shirali-Shahreza, (2008) proposed a embedded technique for solving the alphabet "La" steganography embedded technique problems by improving it. The approach also used the unicode standard and the special characteristic of the Persian and Arabic languages whereby each alphabets have different shapes. The embedded technique is implemented by hiding the bit 1 using the representative form of "Lam" (its Unicode is 0644) , and "Alef' (its Unicode is 0627) characters to write the word "La", the text viewer displays the it in its special form ("u"). Likewise, to conceal the bit 0, the word "La" is written in its usual form, instead of inserting or placing the Arabic extension alphabet between "Lam" (its Unicode is FEEO) and "Alef' (its Unicode is FE8E) characters. It is written by utilizing the code that corresponds to the correct shape of each alphabet (according to its position or place in the word).

Later, Shirali-Shahreza and Shirali-Shahreza (2010) developed a embedded technique for steganography in the Persian and Arabic texts using the Unicode Standard. The embedded technique hides data in texts by altering the Persian and Arabic characters for the alphabets « ک» and « ی». Arabic and Persian alphabets « ک» and « ی», possess similar initial and medial forms, but their final and isolated forms differ. Hence, their Unicode codes also differs. In addition, the proposed embedded technique is vulnerable to a number of attacks such as retyping. Although, it is robust to digital copy-paste procedure, that means copying and pasting the text between programs preserve the hidden message. The figures of the Persian and Arabic texts are different in the Unicode Standard. The digits 4, 5, and 6 assume different shapes in Persian, as well as in Arabic. As a result of this, it can be suggested as a embedded technique for text steganography to be implemented on the remaining Persian and Arabic similar digits.

Sabir (2013) designed an information hiding approach in the formal and non-formal Arabic text. The approach uses the Unicode system characteristics and non-printing characters in hiding information in Arabic texts (formal and non-formal). The proposed embedded technique is based on replacing the alphabets in the original cover text with its Unicode standard on the Unicode standard table in range [0600-06FF], and with its specific shape on the Unicode standard table in range [FE70-FEFF]. The study proposed a shape detection algorithm which detects the shapes of alphabets (single, middle, right, left) and the way it's concocted with its neighbors having similar characters.

Another study by Mohamed (2014) also proposed a steganography algorithm for the Arabic alphabets based on the characteristics of the Arabic texts. The study utilized the isolated alphabets as hidden keys in the Arabic texts which is written in the Unicode format. The concealing program searches for these alphabets in the word presented in the carrier text. Hence, data can be hidden in the carrier text by using the isolated alphabets without being noticed in the target word. In order to simplify the algorithm's complexity, consideration is given to the isolated letters at the beginning of the words, and also at the end of the words. However, this is not applicable to all the isolated letters in the words.

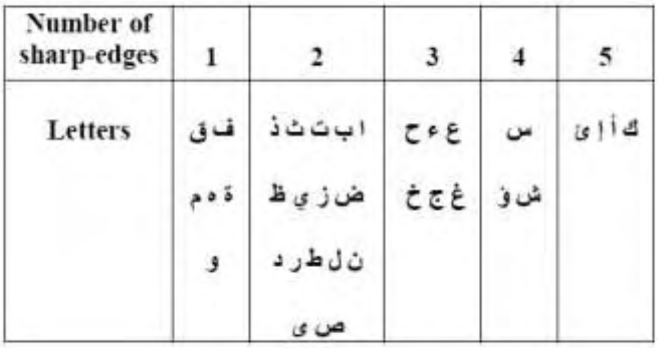
Azawi and Fadhil (2011) designed a steganography embedded technique used for Arabic language electronic writing. Arabic unicode text utilizes two special characters. The first is the zero width joiner character (JWZ) which is used in joining two alphabets, and the second is the zero width non joiner character (JWNZ), used in preventing two alphabets from joining. The benefits of the ZWJ and the ZWNJ non printed unicodes characters, is that it holds secret information without affecting the written contents. Regular expressions are used to assign positions or places suitable for inserting a block of ZWJ and ZWNJ unicodes characters into the words of a given cover text. The ZWJ conceals the secret bit ‘one’, and the ZWNJ conceals the secret bit ‘zero’. The study was evaluated with varying Arabic text steganography embedded techniques, and it was proven that the presented approach produces more security and a high capacity as compared to other Arabic text steganography embedded technique s. Thus, the embedded technique can conceal different files in the Arabic text.

2.2.3.5 Sharp-edges Embedded Method

The study by Roslan et al. (2011) took advantage of the uniqueness of the Arabic characters which is the possession of numerous sharp-edges. The varying number of sharp-edges represents the possibility to conceal the bit 1 and 0 (as secret bits). Alphabets having one (1) sharp-edge, can conceal the secret bit using two (2) conditions that is hiding bit 0 or 1 at the position of sharp-edges. Meanwhile, if the sharp-edges are two (2), the option for the position of secret bits, is in four conditions; 11, 10, 00 or 01.This approach still delivers the maximum capacity of concealed information, even though not every alphabet in the cover-text are fully utilized. The strength of this approach is presented on the reference table. The secret bit position or placement is deduced from this table. Hence, a high security layer is required to guarantee that the table is protected from eavesdroppers.

The approach used by Mersal (2014) operates on dotted and un-dotted alphabets. Random numbers are generated and used to allocate alphabets which are sufficient in hiding 104 bits of secret message. This results in the following alphabets: ر ا ر ن ر ص و ت ب ج ي ا ق ا م أ و ن ش س خ ى ل ي ب ل ي ه ه ا ى ل ص ل ر ا ض ن
ب خ ع س ل س ن ع ق.

The number of sharp-edges on the initial alphabet as shown on the Figure 2.9, determines how the number of bits will he hidden. The secret bit that corresponds to this number is included in the code sequence and the process goes on until all the bits are hidden. For instance, the character (ر) has two (2) sharp-edges. Hence, it can carry the first two (2) binary bits, that is 01, and represent them in the corresponding decimal unit, which is 1. The code sequence generated will be utilized with the stego-text in the decoding or retrieving process. The algorithm is implemented for smart phones using the Java language.



Number of sharp-edges	1	2	3	4	5
Letters	ف ق ة ه م و	ا ب ت ث ذ ض ز ي ظ ن ل ط ر د ص ى	ع ء ح غ ج خ	س ش و	ك أ إ ى

Figure 2.9: Number of Sharp-edges for Each Letter (Mersal, 2014)

2.2.3.6 Poetry System Embedded technique

Khan (2014) proposed to use the Arabic poetry system in hiding secret bits. Since there is a representation of binary units embedded in each Arabic poem, it can be utilized as a means of hiding secret bits. The main idea here is to presume that the embedded binary bits position in poems, contain secret bits. The real secret bit is either equivalent to the binary position, or equivalent to its reverse. For the purpose of increasing the capacity of the proposed embedded technique, diacritics and kashida approaches have been utilized in some cases. The study included redundant embedded technique s only in cases where the secret bit contrast the binary bit of the corresponding alphabet in the cover poem.

2.2.3.7 Changing Font Embedded technique

The study by Ardakani et al. (2015) conceals information by preparing and arranging the hidden message using font changing feature of end dots that appear in sentences of a word file. The Persian alphabets in a word file were examined to check the font letters which resulted in the selection of alphabets that are not connected to other alphabets that come before and after the character. The challenge faced in modifying font letters in a word file is that alphabets with changed fonts, are separated from the alphabet that comes before and after them. This results in a contrast in the hidden information, because the demarcation of alphabets can be noticed in text.

2.2.3.8 A brief review of Arabic Text Steganography Embedded Methods

Then, Table 2.1 shows a brief review of Arabic text steganography embedded techniques in this survey. The table explained the general embedded technique that has been proposed that divided with the list of named of section embedded technique and paper that has been published by previous authors.

Table 2.1: A Brief Review of Arabic Text Steganography Embedded Methods

No.	Embedded Method Used	Authors	Embedded Method that has been proposed	Advantages	Disadvantages
1.	Shifting points	(Shirali-shahreza & Shirali-Shahreza, 2006), (Odeh et al., 2012)	Changing the place of points vertically or horizontally and two bits of information can be hidden in each letter.	-The size the cover text will not change. -High invisibility.	-Used fixed font. -Restored by retyping. -Long execution time.
2.	Harakat	(Aabed et al., 2007), (Memon et al., 2008), (Adnan et al., 2010),(Bensaad & Yagoubi, 2011), (Ahmadoh, 2015)	Sequentially match every diacritic to a bit from the secret bits. If the secret bit is 1 then let the diacritic present as it is, but if it	- Used with different fonts. -The human can performed it manually.	-Low invisibility. -Use fully diacritical text

			is 0 then remove or reverse the diacritic.		
3.	Kashida	(Alnazer & Gutub, 2009), (Al-Haidari et al., 2009)	Putting Kashida where it is applicable and the bit representation of the secret has value of 1.	-Used with different fonts. -Resistance to retyping.	-Low invisibility. -The size the cover text will increase.
4.	Unicode	(M. Shirali-Shahreza & Shirali-Shahreza, 2008a), (Shirali-Shahreza & Shirali-Shahreza, 2010) , (Azawi & Fadhil, 2011), (Sabir, 2013), (Mohamed, 2014)	Looking for the isolated letters as hidden keys in Arabic texts written in Unicode format at the beginning of the words and at the end of the words not all the isolated letters in the words.	-Apply on numerous formats. -The size the cover text will not change. -Resistance to retyping.	
5.	Sharp_edges	(Roslan et al., 2011),(Mersal, 2014)	The character with the number of sharp-edges 1 is possible to hide the secret bit in two possible conditions, which is either hiding secret bit	-High capacity. -High invisibility. -Resistance to retyping. -Apply on numerous formats. -The size the	A high security layer is needed.

			0 or 1at the sharp-edges position.	cover text will not change.	
6.	Poetry system	(Khan, 2014)	Presuppose that the embedded binary bit in a poem position contains a secret bit. The actual secret bit either is the same as that binary position or its reverse.		
7.	Font change	(Ardakani et al., 2015)	Using font changing feature of end dots of sentence in a word file.	-Resistant against of attacks resize text attacks, change of text color, font changes or phrase in the text.	-It has not resistant against attacks by changing the font, copying and cutting attack.

As can be seen from table 2.1, the advantages and disadvantages of various Arabic text steganography embedded methods are summarized. It is found that most of researchers focused to enhance the capacity, security and robustness. However, some embedded techniques concerned to improve the capacity compared to security and robustness. For instance, in shifting point embedded techniques the researchers conducted the enhancement of capacity and security than robustness. Furthermore, the researchers in kashida embedded methods have worked to improve the capacity and robustness compared to security. Meanwhile, most researchers in sharp_edges has improved the capacity and security than robustness.

In conclusion, the study found that, in shifting point, kasida and sharp_edges embedded methods most of researchers concerned to improve the capacity issues compared to security and robustness issues .therefore, the study focused to integrate these three embedded methods toward enhance capacity, security and robustness issues. As explained above, there are two main problems in these embedded methods which have been discovered by previous researchers in the Arabic text embedded technique with the lack of robustness in developing the text steganography embedded technique (Reddy et al., 2014) and low security for covering text in order to hide hidden message (Sun et al., 2010). These two problems can be formulated in the lack performance of the embedded method used. Hence, the lack embedding performance in *kashida*, shifting-point and sharp edges embedded technique needs to be evaluated so that it is easily discovered through this Arabic text steganography embedded method. It is also suggested by Sharma, Upadhyaya, and Agarwal (2013) that the steganography system requires an evaluation measurement of the verification and validation processes. This suggestion anticipates figuring out and reducing the problem of lack effectiveness and low security.

2.2.3.9 Review on Used Evaluation of Arabic Text Steganography Embedded Methods

Table 1.2: The Used Evaluation from previous Researches.

ARABIC TEXT STEGANOGRAPHY EMBEDDED METHODS		**METRIC PARAMETERS**				
		Capacity Ratio	***Usage Ratio***	***Usability Ratio***	***Fitness Performance***	***Run Time***
KASHIDA	Steganography in Arabic Text Using Kashida Variation Algorithm (KVA)(Odeh et al., 2013)	Yes	No	No	No	No

	Improving Security and Capacity for Arabic Text Steganography Using 'Kashida' Extensions.(Al-Haidari et al., 2009)	Yes	Yes	No	No	No
	Exploit Kashida Adding to Arabic e-Text for High Capacity Steganography(Alnazer & Gutub, 2009)	Yes	No	No	No	No
SHIFTING POINTS	A New Approach to Persian/Arabic Text Steganography(Shirali-shahreza & Shirali-Shahreza, 2006)	Yes	No	NO	No	No
	A Novel Arabic Text Steganography Embedded technique Using Letter Points and Extensions(Odeh et al., 2012)	No	No	No	No	No
	Steganography by Multipoint Arabic Letters(Odeh et al., 2012)	No	No	No	NO	NO
SHARP_EDGES	SHARP-EDGES EMBEDDED TECHNIQUE IN ARABIC TEXT STEGANOGRAPHY(Roslan et al., 2011)	Yes	Yes	No	No	NO
	Arabic Text Steganography in Smartphone(Mersal, 2014)	Yes	Yes	NO	NO	NO

Based on Table 2.2, the previous researchers focus only on the capacity and usage evaluation while other researchers only display their embedded technique that shows improvement and more useful than the previous embedded technique.

In order to reduce the problems in lack of robustness in embedding the hidden message for developing the embedded technique and low security in cover text, the verification and validation processes need to be carried out in developing the embedded method. It is anticipated that it can be developed to strengthen the text steganography security and improve robustness of Arabic text steganography embedded methods. This is because the obtained results from the evaluation are expected to show a specific performance of the embedded method that will be used. Therefore, an evaluation can be recommended to overcome both problems so that the evaluation measurement can be a guidance in achieving the desired goal in the system development. Despite this study as a challenge to evaluate the performance of the previous embedded methods in Arabic text steganography embedded method, this research is looking forward to being able to evaluate and justify the embedded method by integrating the kashida, shifting points and sharp edges embedded methods in one method called the *Triad-bit method*.

2.3 Summary

In this work, proposed a survey of the current embedded method s and its embedded method in Arabic text steganography used to hide the secret information in text medium. This survey has reviewed some published paper about embedded method s in steganography in the medium of text. This work showed that the most embedded method s that had been used in Arabic text steganography can be classified into many of languages. These languages are Jawi, Persian or Urdu because they have same text scripting of Arabic language. This review documented the problem of this study, it find that the kashida, shifting point, and sharp_edges had lack performance to embed the hidden message.

CHAPTER THREE
RESEARCH METHODOLOGY

3.1 Introduction

In this chapter, the research methodology phases of the study are described d in order to meet the objectives of the research. These include problems of Arabic text steganography in using the proposed method in order to integrate the embedded techniques in the current Arabic text methods. The research methodology comprises three phases such as phase I (Investigation Problem), phase II (Process Design) and phase III (Evaluation Process) as shown in Figure 3.1.

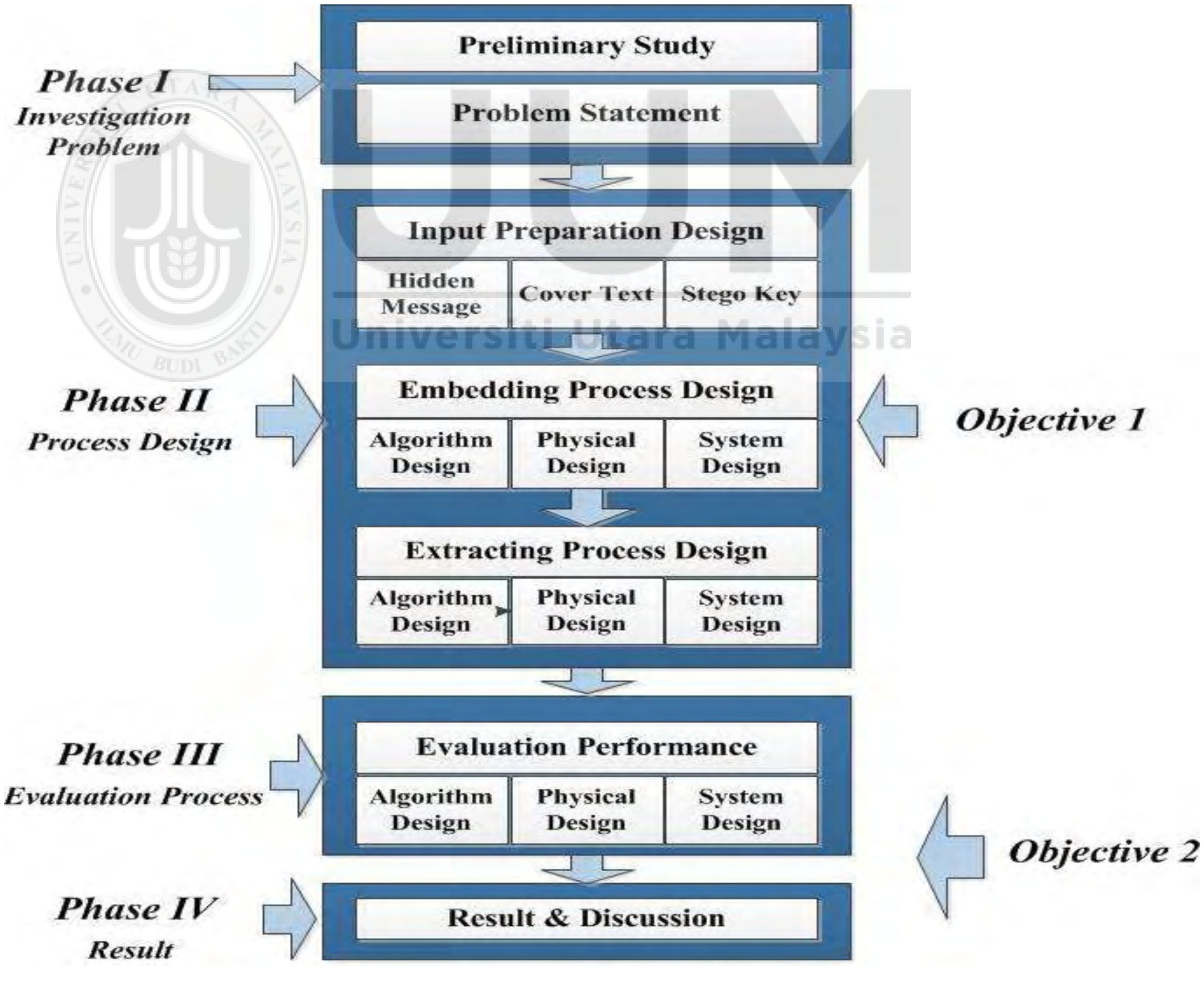


Figure 3.1: Research Methodology Design

3.2 Phase I: Investigation Problem

The initial step in this study is theoretical study. In this phase, the state of Arabic text steganography is critically analyzed in order to identify the area that needs further investigation.

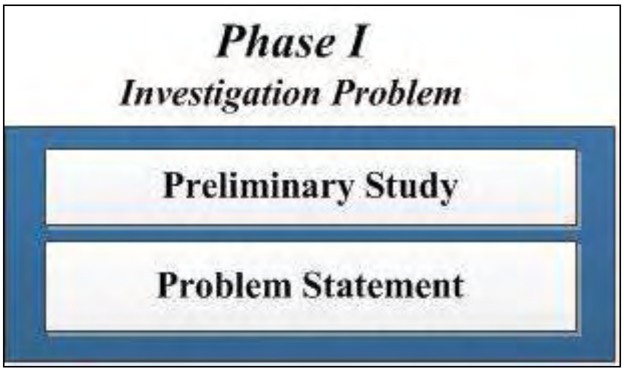


Figure 3.2: Phase I(Investgation Problem)

3.2.1Preliminary Study

In this step, the study is surveying the previous work based on two views of natural language steganography which are text steganography as one view while Linguistics is another view. Furthermore, the study focuses on text steganography based on Arabic language in order to identify the current embedded techniques used in Arabic text steganography.

After analyzing the previous literature, it has an issue of easily discovered the changes of text in steganography method that have been embedded. From the issue, a general problem of lack performance in the embedded technique arises because most of the embedded techniques in text steganography do not perform evaluation measurement.

3.2.2Problem Statement

In this step the study formulates the problem of Arabic text steganography embedded techniques based on the analysis of a previous study. The low invisibility, confusion and low security level lead to the problem of shifting points, kashida and sharp_edges embedded technique in the Arabic text domain. These problems are able to detect the embedding messages easily. That is why the study tries to integrate the embedded techniques in the selected methods in order to produce *triad-bit method* on Arabic text steganography.

3.3Phase II: Process Design

The theoretical study performed in the previous phases that formulated the problem conditional states of Arabic text steganography embedded technique becomes the basis for developing the experimental design. There are three sections of design in this phase to develop the proposed embedded techniques which are Input Design Preparation, Triad-bit method Design and Process Design.

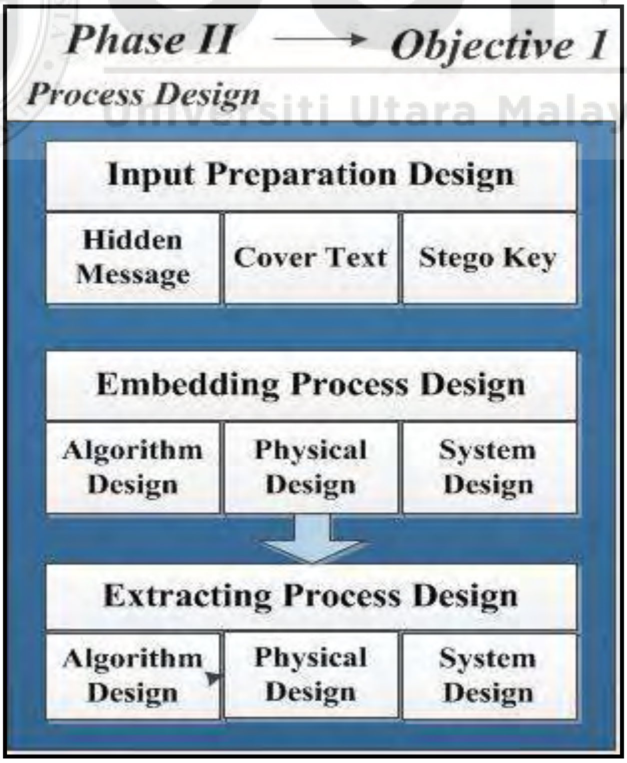


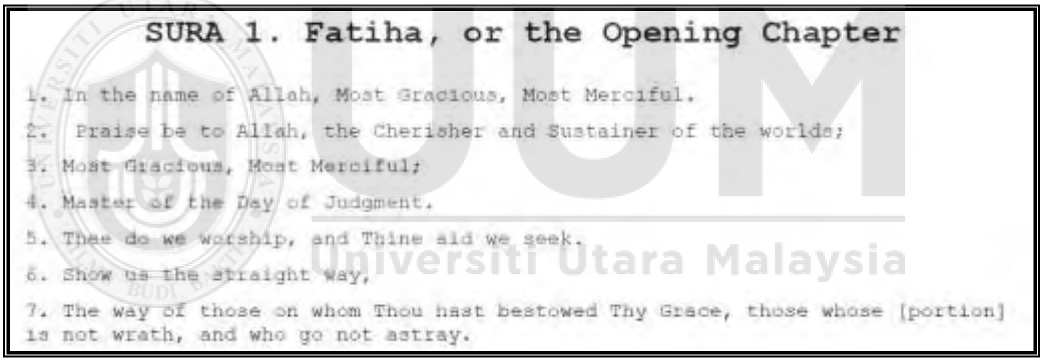
Figure 3.3: Phase II(Process Desigen)

3.3.1 Input Design Preparation

This stage is required to be the input on starting the process which consists of two; the hidden message and the cover text. The explanations are as follows:

a. Hidden Message

The hidden message is the information that are embedded in a cover text based on the Arabic text steganography embedded technique. The hidden message will be converted into a single dimension binary form (bit stream), before it is embedded into the cover text. This study applied steganography algorithm on Arabic letters using the same example in Figure 3.2.



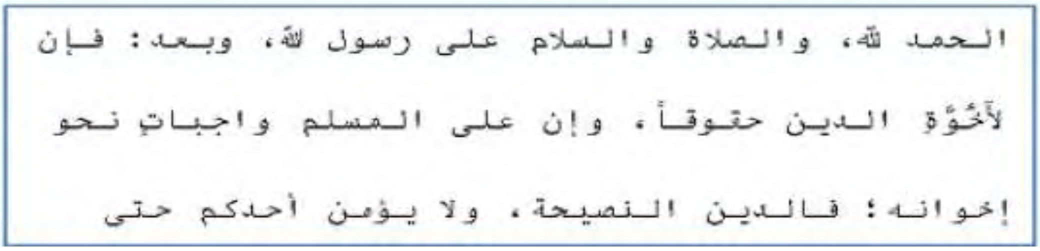
SURA 1. Fatiha, or the Opening Chapter

1. In the name of Allah, Most Gracious, Most Merciful.
2. Praise be to Allah, the Cherisher and Sustainer of the worlds;
3. Most Gracious, Most Merciful;
4. Master of the Day of Judgment.
5. Thee do we worship, and Thine aid we seek.
6. Show us the straight way,
7. The way of those on whom Thou hast bestowed Thy Grace, those whose [portion] is not wrath, and who go not astray.

Figure 3.4: Example of hidden message (Roslan et al., 2014)

b. Cover Text

Cover text is the original text that would be embedded by the hidden message in the process to hide information using Arabic text steganography embedded technique. A cover text that is used in this research is a standard common text paragraph in plain text (.txt) documents as shown in figure 3.3.



الـحمـد لله، والـصلاة والـسلام عـلـى رسول الله، وبـعـد: فـإن

لأخُوَّة الـديـن حقـوقـاً، وإن عـلـى الـمسلم واجـبـات نـحـو

إخوانـه: فـالـديـن الـنصيحة، ولا يـؤمـن أحـدكـم حـتـى

Figure 3.5: Example of Cover Text (Alginahi et al., 2012)

c. Stego Key Used

In this step the study presents the proposed stego key called Triad-bit method. This method will integrate the three Arabic text steganography embedded techniques which are shifting point, kashida and sharp edges. Triad-bit method enciphers the encrypted embedding algorithm and deciphers the extract message embedded technique. In this study, the proposed method will be the important algorithm in embedding and extracting process.

The Triad-bit method is divided into two embedded techniques based on the number of secret bit that can be hidden in one character. The first embedded technique is called Stream-bit technique which utilizes the number of sharp edges of Arabic characters to embed the stream bit of a hidden message. Besides that, the second embedded technique is called single-bit embedded technique in which it can hide only one bit (0 or 1) in one Arabic character. Below is an explanation of the embedded technique.

i. Bit-stream Embedded technique

This embedded technique uses the number of sharp edges for Arabic letters to hide the stream of secrecy. The Arabic characters are grouped in five based on the number of the sharp edges as shown in table 3.1.

Table 3.1: Letter Group of Sharb_Edges Embedded Technique (Roslan et al., 2011)

ID of Group	Number of sharp-edges	Binary bit to be hide	Letter used
A	1	1,0	ق ف م ه ة و
B	2	11,10,01,00	ذ ث ت ب ا ظ ي ز ض د ر ط ل ن ى ص
C	3	000,100,010,110, 001,101,011,111	ح ء ع خ ج غ
D	4	0000,0001,0010, 0011,0100, 0101, 0110,0111,1000, 1001,1010,1011, 1100,1101,1110,1111	س ؤ ش
E	5	00000,10000,01000,110000, 00100,10100,01100,11100, 111000,…11111	ئ إ أ ك

Based on Table 3.1, the letter with contains one sharp edge is identified as group ID A such as “ف,ق,ة,ه, م,و” will hide 0 or 1bit of a hidden data. Meanwhile, a letter with two sharp edges is identified as group ID B such as “ا, ب, ت,ث,ذ,ض,ز,ي,ظ,ن,ل,ط ,ر ,د ,ص ,ى” that will hide four possible conditions: 11,10,00 or 01 bit of a hidden data.

ii. Single-bit Embedded technique

This embedded technique utilizes the points of Arabic letter and the extension inserted after the Arabic letters per word. Firstly, the shifting point embedded technique used in the pointed Arabic characters in the Arabic language called single-bit 1. Each character can be used to hide one secret bit ‘0’ or ‘1’ as shown in Table 3.2.

Table 3.2: Letter Group of Shifting Point Embedded technique (Odeh at el., 2012)

ID of Group	Letter effects	Binary bit to be hide	Letter used
A	Shifting up the point. No shifting the point.	1 0	ب, ث, ت , ج, خ, ذ, ز, ش ,ض,ظ ,غ, ف,ق, ن, ي

Based on Table 3.2, the letter that is used to hide ‘1’ will shift its point while no shifted of point will be used when 0 is hidden. Secondly, the kashida means a embedded technique that is considered as the pointed letters and the redundant Arabic extension characters. The pointed letter with kashida is used to host ‘1’ and the un-pointed letters with a kashida is used to hold the secret bit ‘0’.

Table 3.3: Letter Group of Kashida Embedded Technique (Al-Haidari et al., 2009)

ID of Group	Letter effects	Binary bit to be hide	Letter used
A	Un-pointed letter with kashida	0	ر د ح ا ص س ك ع ط و ه م ل
B	Pointed letter with kashida	1	ث ت ب ز ذ خ ج ض ش ف غ ظ ي ن ق

Nevertheless, these three selected embedded techniques can manipulate the Arabic characters and other languages that have a similar script. All these embedded techniques are prominent in Arabic text steganography, which has different uniqueness and flow process in embedding the hidden message. Hence, these selected embedded techniques are the main concern to develop the proposed method in Arabic text steganography.

3.4 Phase III: Evaluation Performance

The performance evaluation of the Triad-bit method in experimental design is the main objective of this phase. There are four stages such as the capability of stego text size, the capability of cover text fitness performance of Triad-bit method, the running time of Triad-bit method and the performance evaluation of Triad-bit method, as well as the comparison of performance evaluation involved in this phase.

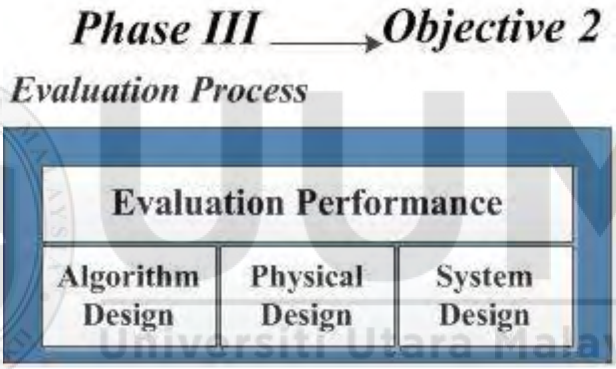


Figure 3.6: Phase III(Evaluation Process)

The first stage of this phase is the capability measurement of a stego text size. The second is the capability measurement of a cover text by several parameter metrics such as the usage and the usability. This is followed by the third stage which is the fitness performance of the Triad-bit method measured by several metrics performance such as capacity and saving space, and finally, the running time. Hence, the performance of Triad-bit method obtained will be justified through these stages.

3.4.1 Capability of Stego Text Size

3.4.2 Usage of Cover Text

Usage ratio is used to determine the total characters of a cover text that can be used to embed the hidden message(Roslan et al., 2014). This analysis is very important for steganographer to understand the capability of a cover text to hide a hidden message

$$\textit{Usage Ratio} = \left(\frac{\textit{Total of Used Character}}{\textit{Total Character of Cover Text}} \right) \text{x } \textit{100\%}$$

$$= \frac{\sum_{i=1}^{1<m<n} a_i}{\sum_{i=1}^{1<m<1n} b_i} \times 100\% \qquad \textit{(3.1)}$$

Where

a = *Total of used character*

b = *Total of cover text character*

3.4.3 Usability of Cover Text

The usability ratio is used to determine the total characters of a cover text that are able to be used in embedding the process of the hidden message (Roslan et al., 2014). This analysis is very important for a steganographer to understand the capability of the cover text to hide a hidden message.

$$\textit{Usability Ratio} = \left[\frac{\textit{Total of Usable Character}}{\textit{Total Character of Cover Text}}\right] \times \textit{100\%}$$

$$= \frac{\sum_{i=1}^{1<m<n} a_i}{\sum_{i=1}^{1<m<1n} b_i} \times 100\% \qquad (3.2)$$

where

a = *Total of usable character*

b = *Total of cover text character*

3.4.4 Fitness Performance of Stego Key

A basic step in this stage is to understand the fitness performance of the stego keys used. Therefore, the fitness performance of stego keys used such as single-bit 1, single-bit 2 and stream-bit are examined. These stego keys used are considered as types of Arabic text steganography method. The metrics performance for stego keys used in this research is adopted from several researchers in steganography works (Odeh *et al.*, 2013; Agarwal, 2013; Por *et al.*, 2013; Lee and Chen, 2013; Akhilandeswari and Jabin, 2014) such as capacity ratio, embedded fitness ratio and saving space ratio.

a. Capacity Ratio (CR)

Capacity ratio is used to determine the total of a hidden text that can be embedded in the cover text. This analysis is very important for a steganographer to understand the capability of a cover text to hide a hidden text.

$$CR = \left(\frac{\textit{Total Bits of Hidden Text}}{\textit{Total Bits of Cover Text}} \right) \text{ x } \textit{100\%}$$

$$= \frac{\sum_{i=1}^{1<m<n} a_i}{\sum_{i=1}^{1<m<1n} b_i} \times 100\% \qquad \textit{(3.3)}$$

where

a = *Total bits of hidden text*

b = *Total bits of cover text*

b. Embedded Fitness Ratio (ER)

Embedded fitness ratio is used to determine the total fitness of a hidden text that can be embedded in a cover text. This analysis is very important in order to understand the fitness capability of a cover text.

$$ER= \left(\frac{\textit{Total Bits of Stego Text – Total Bits of Cover Text}}{\textit{Total Bits of Cover Text + Total Bits of Hidden Text}} \right) \text{ x } \textit{100}$$

$$= \frac{\sum_{i=1}^{1<m<n} a_i}{\sum_{i=1}^{1<m<1n} b_i} \times 100\% \qquad \textit{(3.4)}$$

where

a = *Total number of embedded bits*

b = *Total bits of expected stego text*

c. Saving Space Ratio (SSR)

Saving space ratio is used to determine the total space of a text that can be saved during the embedding process in a cover text. This analysis is very important in order to understand the capability of the maximum space of key used that can be utilized in a cover text.

$$SSR = \left(\frac{\textit{Total Bits of Expected Stego Text} - \textit{Total Bits of Stego Text}}{\textit{Total Bits of Expected Stego Text}} \right) \text{x } 100$$

$$= \frac{\sum_{i=1}^{1<m<n} a_i}{\sum_{i=1}^{1<m<n} b_i} \times 100\% \qquad (3.6)$$

where

a = *Total number of saving space bits*

b = *Total bits of expected stego text*

3.4.5 Running Time

Running time is the period of time when the code of the system is executed (Din et al., 2013). In this part the TBSS system calculates the code execution for the single-bit 1, single-bit 2 and the stream-bit. The comparison of running time of the used embedded technique will provide the information to steganographer about the embedding performance.

3.5 Phase IV: Result and Discussion

This phase revisits the objectives of this study with the related contribution to the steganography on the Arabic text domain.

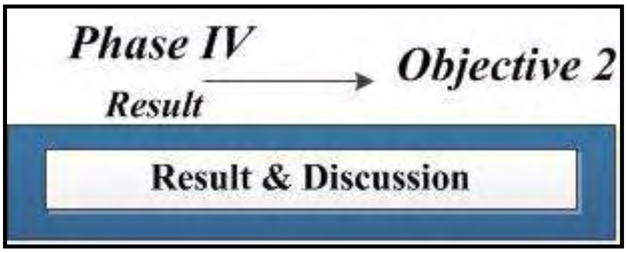


Figure 3.7: Phase IV(Investgation Problem)

3.6 Summary

This chapter organizes a circumstantial description of the research methodology. The beginning of this chapter has illustrated the general research design using phase I, phase II, and the result. Phase I is an investigation problem research started from literature review and problem formulization. Phase II is process design for establishing the input environment and analyzing process environment have been the focus in this chapter while the established input discussed is the necessary input components for establishing the system. Concurrently, the process analyzes a discussion on the component data from the established input in order to embed the process based on algorithm that will be used. Subsequently, this process presents an evaluation of the research that uses five metrics as explain in phase III. Significantly, the output environment displays the result of an evolution measurement after developing and measuring the experimental design in phase IV. Thus, the next topic will discuss the process design of proposed method.

CHAPTER FOUR
THE DESIGN OF TRIADBIT OF METHOD

4.1 Introduction

In this chapter the study described the proposed method called triad-bit method. Triad-bit method integrate three steganography embedded techniques the use the Arabic text as a cover text to embed the hidden message. In this chapter, the process design for embedding and extracting process will be introduced.

4.2 Process Design

The main consist of two main processes; the embedding process and extracting process. Figure 4.1 schematically illustrates the main process of this method.

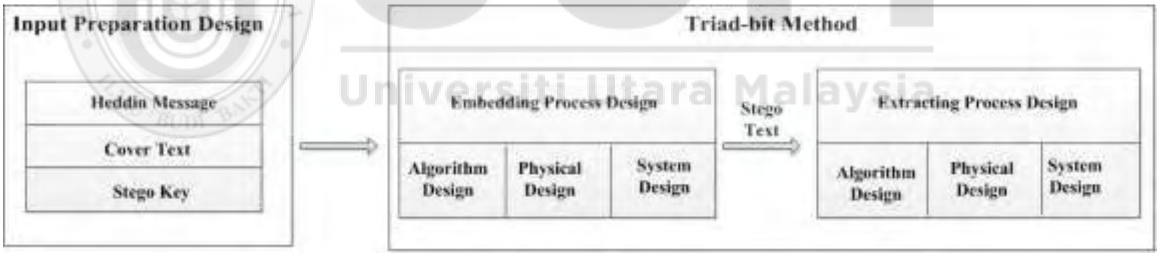


Figure 4.1: Modelling Process of Triad-bit Method

The embedding process is performed by the sender to hide the hidden message into the cover text based on which stego key that will be selected. Simultaneously, the extracting process is performed by the receiver to extract the hidden message from the stego text based on the same stego key used in the same embedding process.

4.2.1 Embedding Process Design

The embedding process is used by the sender to embed the hidden message into the cover text. This process is divided into three stages; preparation required inputs, selection stego key and Triad-bit embedding process. The output of the embedding process is the stego text which will be used by the receiver to retrieve the hidden message. Figure 4.2 shows the embedding flow process of Triad-bit embedded method.

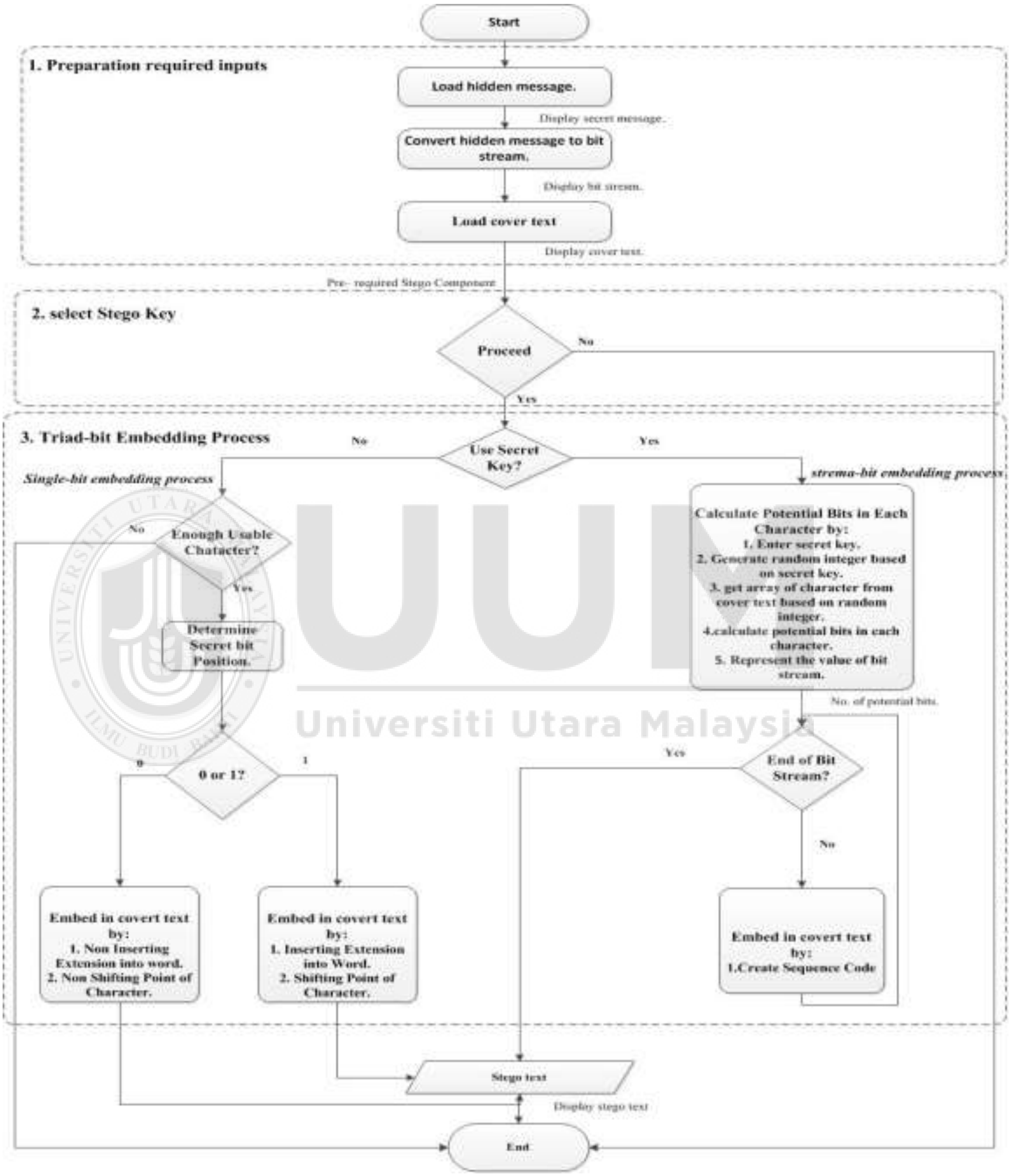

Start
1. Preparation required inputs
Load hidden message.
Display secret message.
Convert hidden message to bit stream.
Display bit stream.
Load cover text
Display cover text.
Pre- required Stego Component
2. select Stego Key
Proceed
No
Yes
3. Triad-bit Embedding Process
No
Use Secret Key?
Yes
Single-bit embedding process
strema-bit embedding process
No
Enough Usable Chatacter?
Yes
Determine Secret bit Position.
0 or 1?
0
1
Calculate Potential Bits in Each Character by:
1. Enter secret key.
2. Generate random integer based on secret key.
3. get array of character from cover text based on random integer.
4.calculate potential bits in each character.
5. Represent the value of bit stream.
No. of potential bits.
Yes
End of Bit Stream?
No
Embed in covert text by:
1. Non Inserting Extension into word.
2. Non Shifting Point of Character.
Embed in covert text by:
1. Inserting Extension into Word.
2. Shifting Point of Character.
Embed in covert text by:
1.Create Sequence Code
Stego text
Display stego text
End

Figure 4.2: Embedding Flow Process of Triad-bit method.

a. Preparation Required Inputs

At this stage, the sender needs to determine and prepare the required inputs by loading the hidden message from a file or typing it directly. Then, it converts the hidden message from a string data type to binary code as a bit stream to be able to encode into the cover text. After that, the sender will load the cover text file or type it directly. The cover text will be an Arabic text and the method will utilize the characteristic of Arabic text to hide the bit stream.

b. Selection Stego Key

The selected embedding algorithm will be determined at this stage by choosing one stego key from three stego keys. The Triad-bit method involves three stego keys (single-bit1, single-bit2 and stream-bit) based on the three Arabic text steganography embedded techniques (kashida, shifting point, and sharp edges).The selection of stego key is the important part to determine the flow of embedding process. The method will be terminated in case there is no selection of the stego key.

c. Triad-bit Embedding Process

This process is embedding the secret bit stream by integrating the three Arabic steganography embedded techniques (kashida, shifting point, and sharp edges). There are two embedding flow processes based on the existence of the secret key. The first flow is stream-bit embedding process in which the secret key is compulsory and the binary code of converted hidden message can hide as bit stream such as ‘0’ or ‘01’ or ‘1011’ for a character. Meanwhile, the second flow is a single-bit embedding process in which the secret key is not required and the binary code of converted hidden message can hide as single bit such as ‘0’ or ‘1’.

i. ***Stream-bit Embedding Process:*** The sender will key in the secret key which has become the important step to generate the random integer array. The value of the random integer array represents the position of used characters from the cover text in the embedding process. After creating the array of used characters, the potential number of hidden bits can be calculated for every character in used character array. Therefore, every bit stream that is hidden in one character will be converted into integer to create the sequence code. The embedding process will be ended when it reaches the end of the bit stream.

ii. ***Single-bit Embedding Process:*** In a case where the secret key is not required for this step, the embedding process will start by counting the usable characters in the cover text. The usable characters mean the characters are able to extend or the characters might have a point or points. When the usable characters are sufficient, the secret bit position of the used characters will be determined based on the number of the secret bits. There are two values of the secret bit. The first value is '1' which means that the extension will be inserted after the used characters per word or the points of the used characters will be shifted up 3 inches. The second value is '0'. It means that the used characters will remain as themselves without any extension to it or shifting the points of the used characters.

4.3 Algorithm Design

There are two main algorithms; the first is the embedding algorithm that involves steps to hide the hidden message into the cover text, and the second is extracting algorithm that applies different steps to extract the hidden message from the stego text. The explanation is stated as below:

4.3.1 Embedding Algorithm Design

In this algorithm there is a set of instructions designed to perform a specific task to hide the hidden message into the cover text. The cover text is an Arabic text where the embedding algorithm utilizes the characteristics of the Arabic characters by integrating the three Arabic text steganography embedded techniques. The embedding algorithm consists of three algorithms; preparation required input algorithm, stream-bit embedding algorithm and single-bit embedding algorithm. The explanations are as follows:

a. Preparation Required Input Algorithm

The required input is prepared in ***Algorithm 4.1*** by loading the hidden message and converting it into bit stream. Then the cover text that is used as a medium to hide the hidden message is loaded.

***Algorithm 4.1*:** Preparation Required Input

```
Load the hidden message.
Convert the hidden message into a bit stream (binary code).
Load the cover text.
```

In this part the sender will prepare the required input; the hidden message and the cover text. Hence, they are supposed to:

a. Load the file that contains the hidden message. The algorithm will calculate the size and the length of the hidden message.
b. Convert the hidden message into bit stream. The algorithm will calculate the length of the bit stream.
c. Load the file that will be used as a cover text to hide the bit stream of hidden message. The algorithm will calculate the length and size of the cover text.

b. Triad-bit Embedding Algorithm

This algorithm integrates the three Arabic text steganography embedded techniques. It is divided into the main algorithm based on which stego key selected by the sender to hide the hidden message. The first algorithm is stream-bit embedding algorithm that uses the secret key to embed stream bit such as ‘0’ or ‘10’ or ‘101000’ in one character. Meanwhile, the second algorithm is single-bit embedding algorithm which the secret key is not required and hide only one bit (0 or 1) in one character.

i. Stream-bit embedding algorithm

This part hides the stream bit in one used character such as ‘1’ or ‘01’ or ‘10100’. The secret key is an important input to start the embedding ***Algorithm 4.2*** as mentioned below:

Algorithm* 4.2: *Stream-bit embedding function

```
Get the inputs from pervious algorithm.
Enter the Secret Key.
Generate the random integer array based on Secret Key value.
Get the array of used character by representing the values of the random integer array as the positions of used character in a cover text.
Calculate the potential embedding numbers for each character in the used character array.
Start from the first character in the used character array and the first bit in the bit stream.
While Not end of bit stream do
    Get the potential amount of character in a used character array.
    Embed number of bits from bit stream equal to the potential amount of character in a used character array.
    Represent the embedding bits into an integer.
End while
Create the sequence code from converted embedding bits into an integer.
Show the usable character and total number of potential embedding numbers of used character.
```

From Algorithm 4.2, it is clear that the embedding process deals with the character as an isolated one without changing its feature. Concurrently, it utilizes the sharp edges characteristics of the Arabic letters to hide the bit stream into the cover text.

The function has two inputs; the first is the bit stream and the second is the cover text that derives from preparation required input algorithm. The important input is the secret key that should be inserted in the first function to achieve the embedding process.

The target of the secret key is to generate a random integer array that will be used to get the used character array. This array is created by representing the values of the random integer array to determine the position of used character in the cover text. After that, the function will calculate the potential numbers that can hide a bit stream in each used character. The embedding process starts by embedding the bit stream into potential numbers of used character as follows:

a. Calculate the potential numbers of used characters.
b. Embed number of bits from bit stream equally to the potential number of used characters.
c. Convert the embedded bit stream into integer array to create the sequence code.
d. Repeat the previous steps until the entire bit stream is embedded. The return of the function of the sequence code will use an important value with the secret key to extract the hidden message from the stego text. The size of the stego key is similar with the size of the cover text.

ii. Single-bit Embedding Algorithm

The ***Algorithm 4.3*** shows the case when the secret key is not required input to hide the hidden message ,this algorithm will hide only one secret bit 0 or 1in each used character.

Algorithm 4.3: **Single-bit embedding function**

Get the inputs from pervious algorithm.

Scan the cover text to get the length of usable characters by counting the pointed

```
characters and extendable character.
Convert the binary bit stream into a string.
If length of bit stream > length of usable characters then
Show exception message “Not enough usable characters”
Terminate the method.
While not end of bit stream do
Start from the first character of a covert text and from the first bit in the bit stream
If character is usable charter then
Determine the position of used character.
Count the usable character as used character
If bit in bit stream = 0 then
1.Set the used character with current font(Unchanged the feature of character)
2. Set the usable character without insert the extension into word
Else
If bit in bit stream = 1 then
1.Set the usable character with new font(Change the feature of character)
2. Set the usable character with insert the extension into a word

End if
End if
Else
Move to next character in a cover text
End if
Show the useable character and used character.
```

From Algorithm 4.3, it can be said that the main function is to embed a single bit (0 or 1) by utilizing the feature or characteristic of Arabic characters. The function calls its input from a preparation that requires input algorithm.The cover text is scanned to calculate the number of usable characters that will be used in the embedding process.

The usable character here means the character with points or the character that is able to extend. The binary bit stream into integer is used to calculate its length, and it is compared with a length of usable character. The result of this comparison is the exception message that shows insufficient usable characters and the method will be terminated in this situation.

The embedding process starts by scanning the cover text and the bit stream one by one in order to do the following:
When the character is a usable character, the function will determine its position and count this as used character. When the bit in the bit stream is ‘0’ the function will change the feature of the used character by changing its current font to a new font or extend the character by inserting an extension into a word. Repeat the previous steps until the entire bits in bit stream are embedded into the cover text. The function determines two values; number of usable characters and used characters. In addition, the algorithm will calculate the size of the stego key.

4.4 Extracting Process Design

This process is used by the receiver to retrieve the hidden message from the stego text. It is the reverse of the embedding process, and it needs the stego text. The final output of this process is the original hidden message. This process is divided into three stages such as preparation required inputs, selection stego key and Triad-bit extracting process. Figure 4.3 shows the extracting flow process of Triad-bit embedded technique.

a. Preparation Required Inputs

In this stage, the receiver will load the stego text. This stego text is the output of the embedding process based on the stego key used to hide the hidden message.

b. Selection Stego Key

The extracting algorithm will use the same stego key that is already used in the previous embedding process. There are three stego keys based on the three Arabic text steganography embedded techniques known as kashida, shifting point, and sharp edges.

c. Triad-bit Extracting Process

This process means extracting the hidden message by integrating the three Arabic steganography embedded techniques (kashida, shifting point, and sharp edges). There are two extracting flow processes based on the existence of the secret key in the previous embedding process. The first flow is stream-bit extracting process where the secret key is compulsory and the binary code of converted hidden message can be retrieved as bit stream such as '0' or '01' or '1011' for a character. Meanwhile, the second flow is a single-bit extracting process where the secret key is not required and the binary code of converted hidden message can be extracted as a single bit such as '0' or '1'. Below are the details of the processes:

i. ***Stream-bit Extracting Process:*** In this stage the sender will key in the secret key and the sequence code that is generated from the embedding process. The secret key will generate the random integer array. The value of the random integer array represents the position of used characters from the stego text in the extracting process. After creating the array of used characters, the potential number of hidden bits can be calculated for every character in the used character array. This process also represents the value of a sequence code based on the potential bits value into a bit stream. Finally, the bit stream is converted into a string to decipher the original hidden message.

ii. ***Single-bit extracting process:*** The secret key is not required for this step. However, the extracting process begins by counting the usable characters from the stego text. The usable characters mean that the characters have the ability to extend or have the point/s. Then, the algorithm will verify the changes of the usable characters as there are two changes to be considered in extracting the process.

On the other hand, if there is any additional extension inserted after the usable character or the points of usable character are shifted up, the algorithm will extract ‘1’.

However, if there is no additional extension inserted after the usable character or non-shifting up for the points of usable character, the algorithm will extract ‘0’.

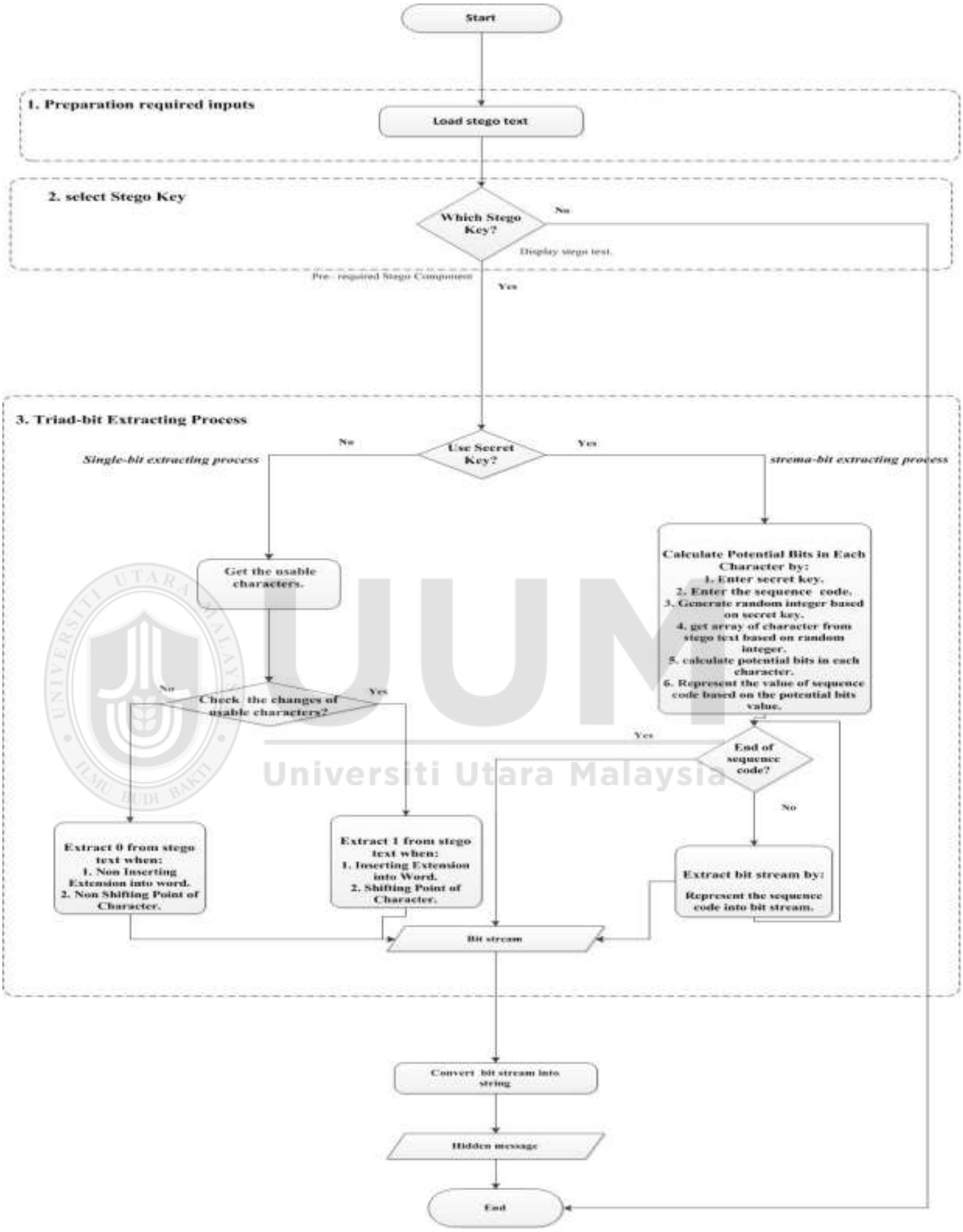

Start
1. Preparation required inputs
Load stego text
2. select Stego Key
Which Stego Key?
No
Display stego text.
Pre- required Stego Component
Yes
3. Triad-bit Extracting Process
No
Use Secret Key?
Yes
Single-bit extracting process
strema-bit extracting process
Get the usable characters.
No
Check the changes of usable characters?
Yes
Extract 0 from stego text when:
1. Non Inserting Extension into word.
2. Non Shifting Point of Character.
Extract 1 from stego text when:
1. Inserting Extension into Word.
2. Shifting Point of Character.
Calculate Potential Bits in Each Character by:
1. Enter secret key.
2. Enter the sequence code.
3. Generate random integer based on secret key.
4. get array of character from stego text based on random integer.
5. calculate potential bits in each character.
6. Represent the value of sequence code based on the potential bits value.
Yes
End of sequence code?
No
Extract bit stream by:
Represent the sequence code into bit stream.
Bit stream
Convert bit stream into string
Hidden message
End

Figure 4.3: Extracting Flow Process of Triad-bit Method.

4.4.1 Extracting Algorithm Design

The algorithm retrieves the hidden bit from a stego text by integrating three Arabic text steganography embedded techniques. Based on which stego key is selected in the previous embedding algorithm, the extracting algorithm will be using the same stego key as well. Indeed, there are two embedding algorithms namely the stream-bit embedding algorithm and the single-bit embedding algorithm. The explanations are as follows:

i. Stream-bit Extracting Algorithm

The ***Algorithm 4.4*** consists of steps that extract the bit stream to form the stego text based on the usage of the same secret key used in the embedding algorithm. It has no concern over the changed feature on the Arabic characters, but it utilizes the sharp edges number in the Arabic characters to extract the hidden message. The retrieved stream bit stream will be as ‘1’ or ‘01’ or ‘0011’. The details are placed below:

***Algorithm 4.4*:** Stream-bit Extracting Function

```
Enter the Secret Key.
Enter the sequence code.
Generate the random integer array based on Secret Key value.
Get the array of used character by representing the values of the random integer array as the positions of used chacter in a stego text.
Calculate the potential embedding numbers for each character in the used character array.
Start from the first character in the used character array.
While Not end of bit stream do
    Get the potential amount of character in a used character array.
    Represent the value of sequence code based on the potential bits value to get the bit stream of hidden message.
End while
Convert the bit stream into string to get the original hidden message.
```

From ***Algorithm 4.4***, it is clearly seen that the stream-bit extracting algorithm needs two inputs; the secret key and the sequence code that is generated from the embedding process. The secret key will generate the random integer number that represents the position of used characters from the stego text.

After that, the function will calculate the potential numbers that can extract a bit stream in each used character. This extracting algorithm is done as follows:

a. Calculate the potential extracting numbers of used characters.
b. Represent the value of sequence code stream into bit based on the potential bits value.
c. Convert the extracted bit stream into integer into string to get the original hidden message.

ii. Single-bit Extracting Algorithm

In this algorithm, the secret key is not required as the input. What is important is to know which stego key is used in the embedding algorithm. The single-bit extracting algorithm notices the changed feature of the Arabic characters to extract only ‘0’ or ‘1’.

Algorithm 4.5: single-bit embedding function

```
Scan the stego text to get the usable character array by counting the pointed characters and extendable character.
While not end of usable character array do
Start from the first character of the usable character.
If (character change = font change) or (character change = insert extension after character) then
1. Extract 1 bit.
Else
```

```
Extract 0 bit.

End if
Get the bit stream.
Convert the bit stream into string to get original hidden message.
```

From ***Algorithm 4.5***, it can be observed that the single-bit extracting algorithm scans for the characters that have changed its feature to extract‘0’ or ‘1’.

The extracting process begins by scanning the stego text and getting the usable characters (pointed characters or extendable characters) as described below:

a. When the usable characters have changes in its feature, the algorithm will extract ‘1’ bit.
b. When the usable character has no change in its feature, the algorithm will extract ‘0’ bit.
c. Repeat the previous steps until the bit stream of hidden message is retrieved.
d. Convert the bit stream into a string to retrieve the original hidden message.

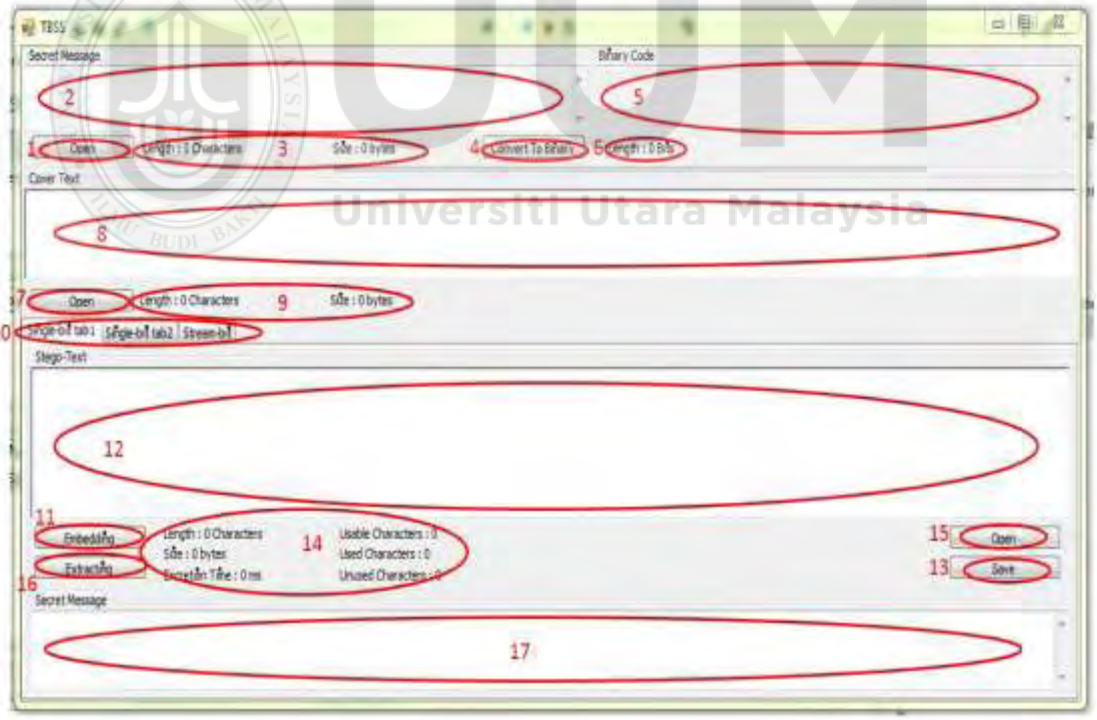
4.5 System Design

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. The system is divided into physical and logical.

4.5.1 Physical Design

The physical design is a graphical representation of a system showing the system’s internal and external entities as well as the flows of data into and out of these entities.

In this section, the prototype of Traid-Bit Steganography System (TBSS) will be discussed. TBSS has been developed with c# programming language by using Visual Studio 2013 IDE as a tool to develop the system. Visual Studio 2013 is a complete set of development tools for building ASP.NET web applications, XML Web Services, desktop applications, mobile applications and more. Visual C#, Visual Basic, and Visual C++ all use the same integrated development environment (IDE), which enables tool sharing and eases the creation of mixed-language solutions. In addition, these languages use the functionality of the .NET Framework, which provides access to key technologies that simplify the development of all kinds of projects of ASP Web applications and XML Web Services. Figure 4.4 shows the user interface that is concerned with how users add information to the system and with how the system presents information back to them. The expansion of TBSS components in Table 4.1.



Length : 0 Characters
Size : 0 bytes
Length : 0 Bits

Length : 0 Characters
Size : 0 bytes

Length : 0 Characters
Size : 0 bytes
Execution Time : 0 ms
Usable Characters : 0
Used Characters : 0
Unused Characters : 0

Figure 4.4: User Interface of TBSS

Table 4.1: Components of TBSS

Number	Component	Description
1	Open Button	**The button to load The hidden message file that be embedded in the cover text.**
2	Hidden message Area	**The area to show the hidden message before convert it into bit stream.**
3	Length and Size Result	**The area to show the calculation result of length and size of hidden message.**
4	Converted Button	**The button to convert the hidden message into bit stream.**
5	Converted Result	**The area to show the result of converting the hidden message from string to bit stream.**
6	Length Result	**The area to show the length calculation result of bit stream.**
7	Open Button	**The button to load the cover text file.**
8	Cover Text Area	**The area to show the cover text that be used to embed the bit stream of hidden message.**
9	Length and Size Result	**The area to show the length and size calculation result of cover text.**
10	Stego Key Tabs	**The tabs to choose one embedded technique from three Arabic steganography embedded techniques. The three embedded techniques are Single-bit tab1, single bit tab2 and stream bit.**
11	Embedding Button	**The button to start the**

		embedding process base on chosen embedded technique.
12	Stego Text Area	**The area to show the stego text that be used to show the result of embedding and to extract the bit stream of hidden message.**
13	Save Button	**The button to save the stego text**
14	Capacity Results	
15	Open Button	**The button to open the stego text for extracting the hidden message.**
16	Extracting Button	**The button to start the extracting process.**
17	Extracting Hidden message	**The area to show the extracting hidden message.**

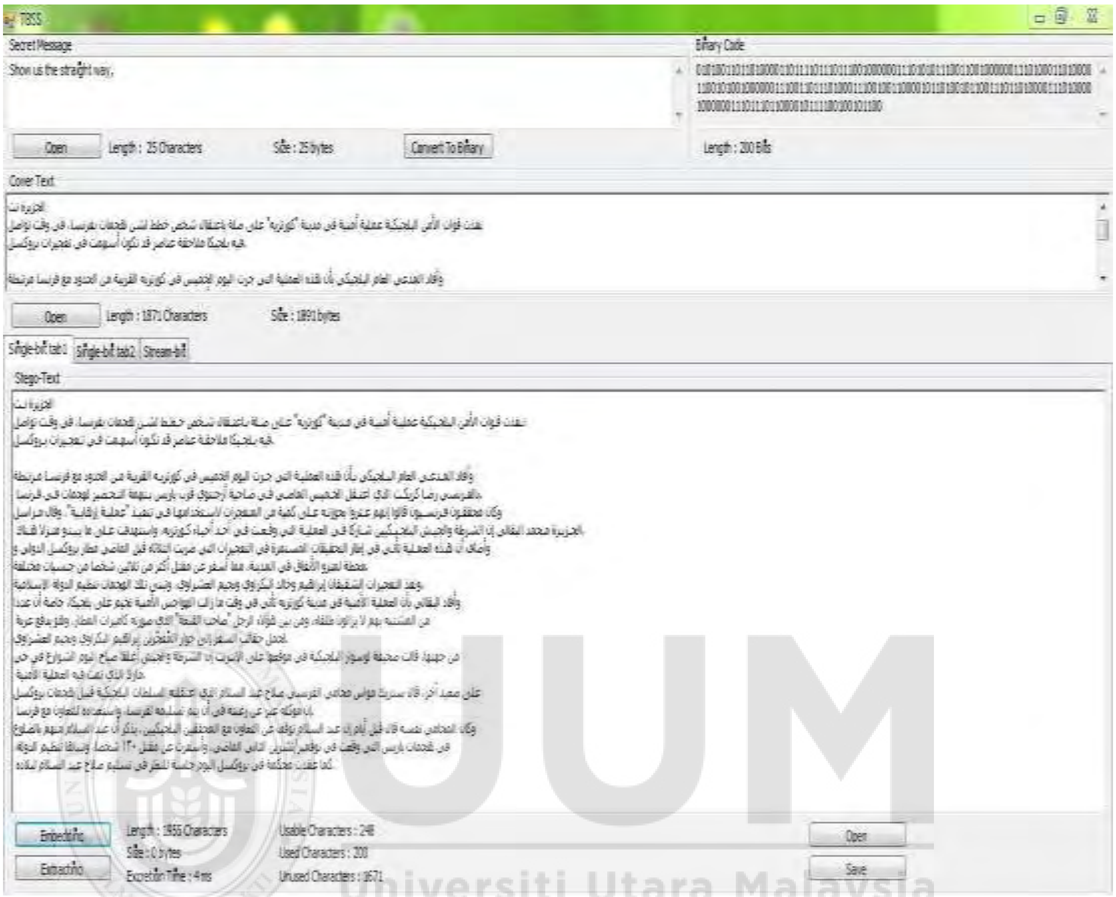
The Figure 4.4 and Table 4.1 shows the user interface of TBSS for embedding and extracting. For embedding process, firstly the sender will load or key in the hidden message. Then, convert it into a bit stream (binary code). The next step is the sender will load the covert text that will be used to embed the hidden message. The TBSS will calculate the length, and the size of the cover text. Thirdly, the sender will choose the stego key that will be used in the embedding process. In fact, there are three stego keys such as single-bit 1, single-bit 2and stream-bit, in which the sender can save the stego text as .txt file.

For extracting process, the receiver will load the stego text file. Then by pressing the extracting button the system will extract the hidden message from the given stego text based on the used stego key.

a. Embedding Physical Design Implementation

The embedding implementation of TBSS is introduced in this stage. It is divided into three embedding implementation for Single-bit1 technique, Single-bit 2 technique and Stream-bit technique.

i. Single-bit 1 embedding implementation



Show us the straight way,

Length : 25 Characters Size : 25 bytes

[illegible]

Length : 200 Bits

الجزيرة نت
نفذت قوات الأمن البلجيكية عملية أمنية في مدينة "كورتريه" على صلة باعتقال شخص خطط لشن هجمات بفرنسا، في وقت تواصل
فيه بلجيكا ملاحقة عناصر قد تكون أسهمت في تفجيرات بروكسل.

وأفاد المدعي العام البلجيكي بأن هذه العملية التي جرت اليوم الخميس في كورتريه القريبة من الحدود مع فرنسا مرتبطة

Length : 1871 Characters Size : 1891 bytes

الجزيرة نت
نفذت قوات الأمن البلجيكية عملية أمنية في مدينة "كورتريه" على صلة باعتقال شخص خطط لشن هجمات بفرنسا، في وقت تواصل
فيه بلجيكا ملاحقة عناصر قد تكون أسهمت في تفجيرات بروكسل.

[illegible]

Length : 1855 Characters
Size : 0 bytes
Execution Time : 4 ms

Usable Characters : 248
Used Characters : 200
Unused Characters : 1671

Figure 4.5: Embedding Implementation of Single-bit1 technique

The provided Figure 4.5 presents the implementation of embedding interface by using single-bit 1. It is clear that the hidden message is loaded and it is converted into bit stream. The covert text is loaded and the embedding result is shown in stego text area. The system calculated the length, size, execution (run) time, used character and usable characters.

iii. Single-bit 2 Embedding Implementation

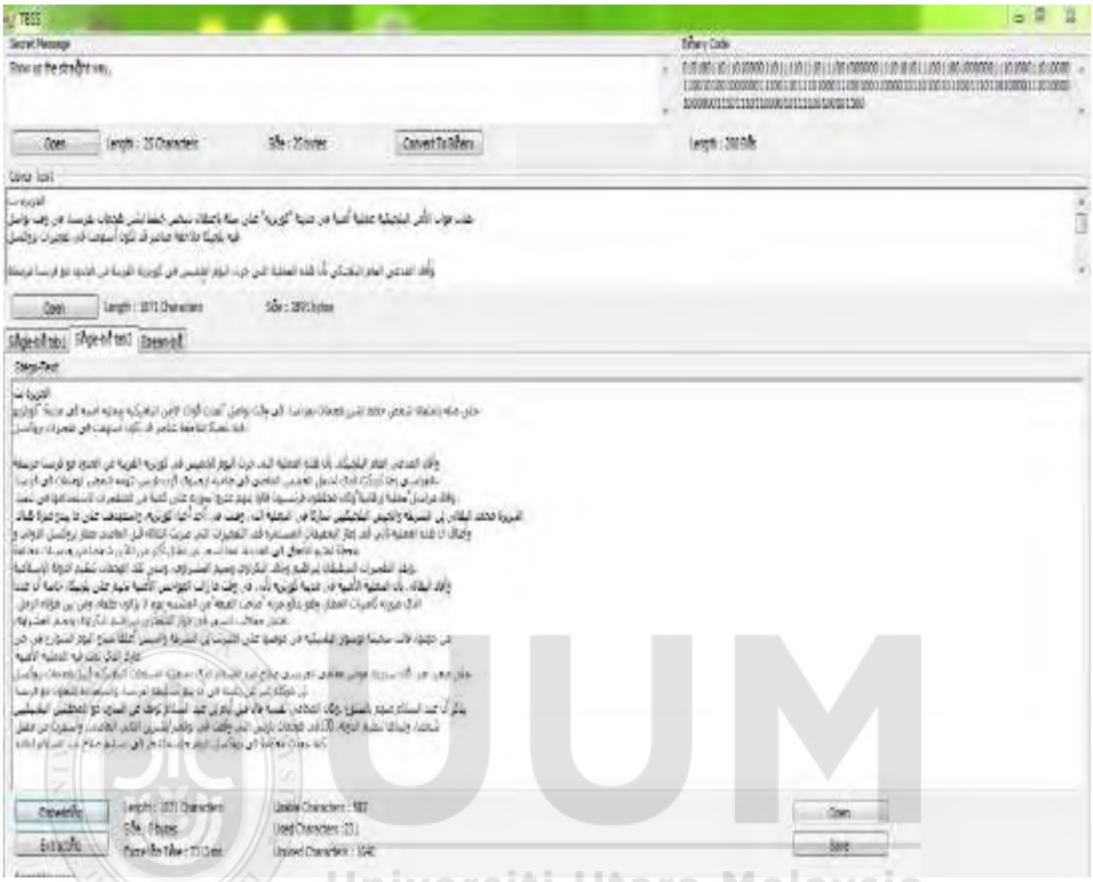
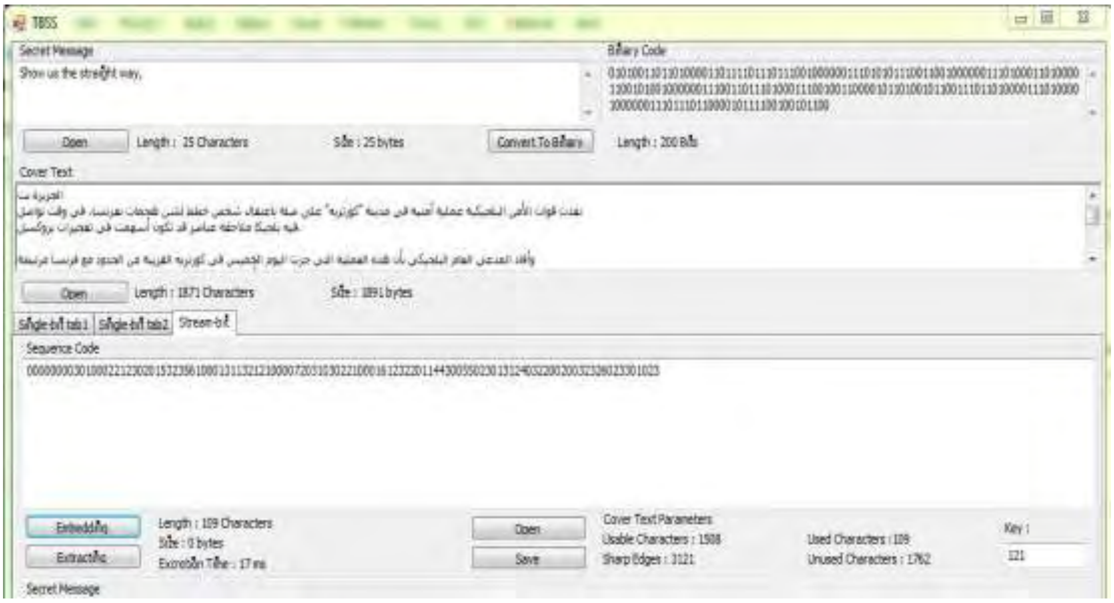


Figure 4.6: Embedding Implementation of Single-bit 2 technique

Figure 4.6 presents the implementation of embedding interface by using single-bit 1. It is clear that the hidden message is loaded and then converted into bit stream. The cover text is loaded and the embedding result is shown in stego text area. The system calculated the length, size, execution (run) time, used character and usable characters.

iii. Stream-bit Embedding Implementation



Secret Message

Show us the straight way,

Binary Code

```
010100110110100001101111011101110010000001110101011100110010000001110100011010000
110010100100000011100110111010001110010011000010110100101100111011010000111010000
10000001110111011000010111100100101100
```

Open | Length : 25 Characters | Size : 25 bytes | Convert To Binary | Length : 200 Bits

Cover Text

الجزيرة نت

نفذت قوات الأمن الباكستانية عملية أمنية في مدينة "كويتة" على صلة باعتقال شخص خطط لشن هجمات تفجيرية في وقت تواصل فيه باكستان ملاحقة عناصر قد تكون أسهمت في تفجيرات بروكسل.

وأفاد المتحدث العام الباكستاني بأن هذه العملية التي جرت اليوم الخميس في كويتة القريبة من الحدود مع [illegible]

Open | Length : 1871 Characters | Size : 1891 bytes

Sequence Code

```
0000000030100022123020153235610001311321210000720310302210001612322011443005502301312403220020032328023301023
```

Length : 109 Characters

Size : 0 bytes

Excution Time : 17 ms

Cover Text Parameters

Usable Characters : 1508

Sharp Edges : 3121

Used Characters : 109

Unused Characters : 1762

Key :

121

Secret Message

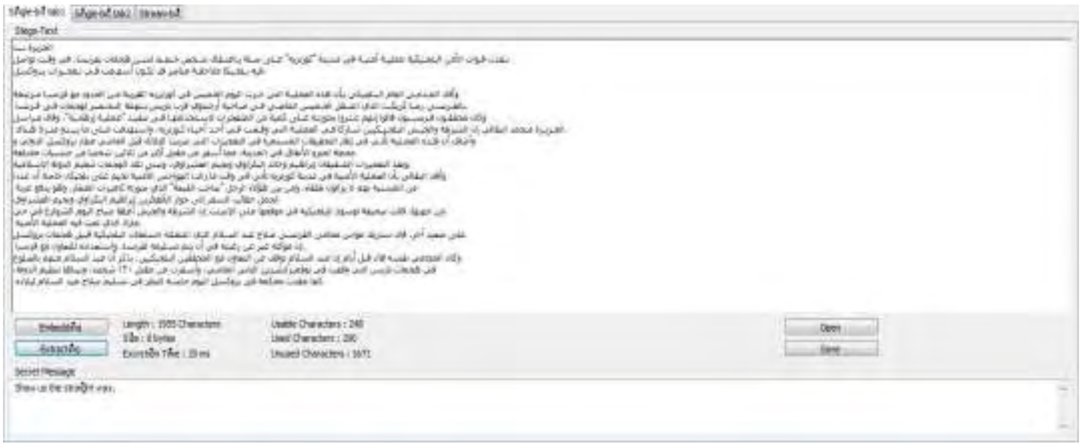
Figure 4.7: Embedding Implementation of Stream-bit Technique

In the above Figure 4.7, the implementation of embedding interface by using Stream-bit technique. It can be clearly seen that the secret key is the important input to generate the sequence code. The sequence code is generated by pressing the embedding button. The stego text size is similar with the cover text size because there is no change on cover text by using this embedded technique. The system calculated the length, size, execution (run) time, used characters, usable characters and sharp edges. Also, the sender saved the sequence code and the secret key in different files before sending it to the receiver.

j. Extracting Physical Design

The extracting implementation of TBSS is introduced in this stage. It is divided into three extracting implementation for Single-bit1 technique, single-bit2 embedded technique and Stream-bit technique.

i. *Single-bit 1 Technique*



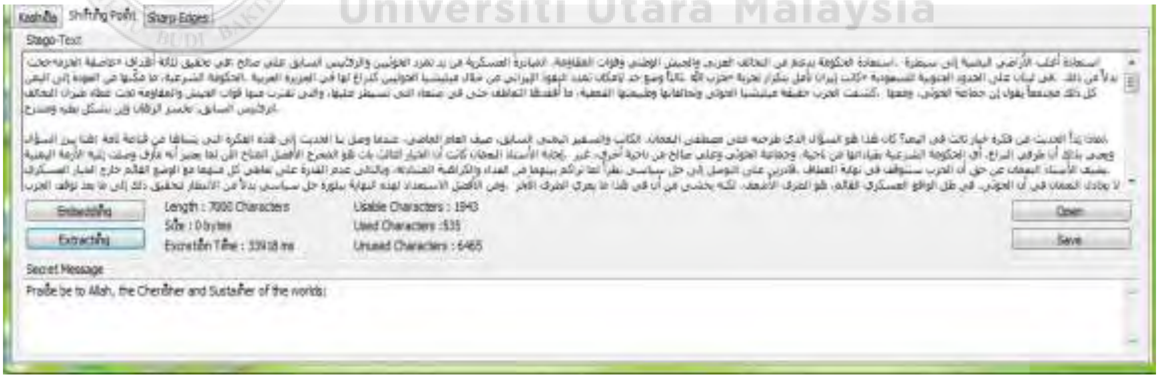
Length : 1905 Characters
Size : 0 bytes
Execution Time : 20 ms
Usable Characters : 240
Used Characters : 234
Unused Characters : 1671

Show us the straight way.

Figure 4.8: Extracting Implementation of Single-bit 1 Embedded Technique

The Figure 4.8 presents the implementation of extracting interface by using single-bit 1.The stego text is loaded and shown in stego text area. The system extracts the hidden message from a given stego text based on the steganography embedded technique that used in embedding process.

ii. *Single-bit Technique*



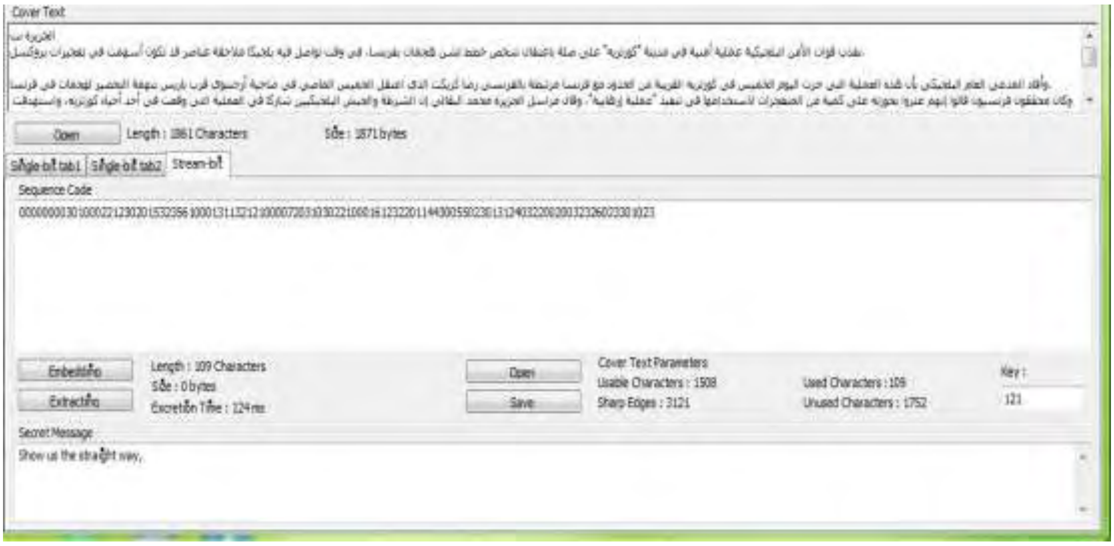
Length : 7000 Characters
Size : 0 bytes
Execution Time : 33418 ms
Usable Characters : 1943
Used Characters :535
Unused Characters : 6465

Praise be to Allah, the Cherisher and Sustainer of the worlds;

Figure 4.9: Extracting Implementation of Single-bit 2 Technique

The Figure 4.9 shows the implementation of extracting interface by using single-bit 2.The stego text is loaded and shown in stego text area. The system extracts the hidden message from a given stego text based on the steganography embedded technique that used in embedding process.

iii. Stream-bit 1 Technique



Cover Text: [illegible]

Length : 1861 Characters

Size : 1871 bytes

Sequence Code: 000000030100022123020153235610001311321210000720310302210601612322011443005502301312403220020032326023301023

Length : 109 Characters

Size : 0 bytes

Excretion Time : 124 ms

Cover Text Parameters

Usable Characters : 1508

Sharp Edges : 3121

Used Characters : 109

Unused Characters : 1752

Key : 121

Secret Message: Show us the straight way,

Figure 4.10: Extracting Implementation of Stream-bit Technique

The above Figure presents the implementation of extracting interface by using stream-bit. The sequence code is a secret key that are loaded and shown in a sequence code area and a secret key text. The system extracts the hidden message from a given stego text based on the steganography embedded technique that is used in the embedding process.

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling by using an over-abstract (and sometimes graphical) model of the actual system.

4.6 Logical Design

The logical design is a graphical representation of a system showing the system's process and the flows of data into and out of the process.

4.6.1 Embedding Logical Diagram

The logical diagram of the embedding process is presented. It is divided into the embedding use case model and embedding class diagram. The details are as follows:

a. Embedding Use Case Model

This use case model describes the proposed functionality of the embedding part of TBSS system. This use case represents a discrete unit of interaction between a sender and the TBSS system. A use case is a single unit of meaningful work to embed the hidden message into a cover text. There are five main use cases and one of them extends to the other two use cases as shown in Figure 4.11.

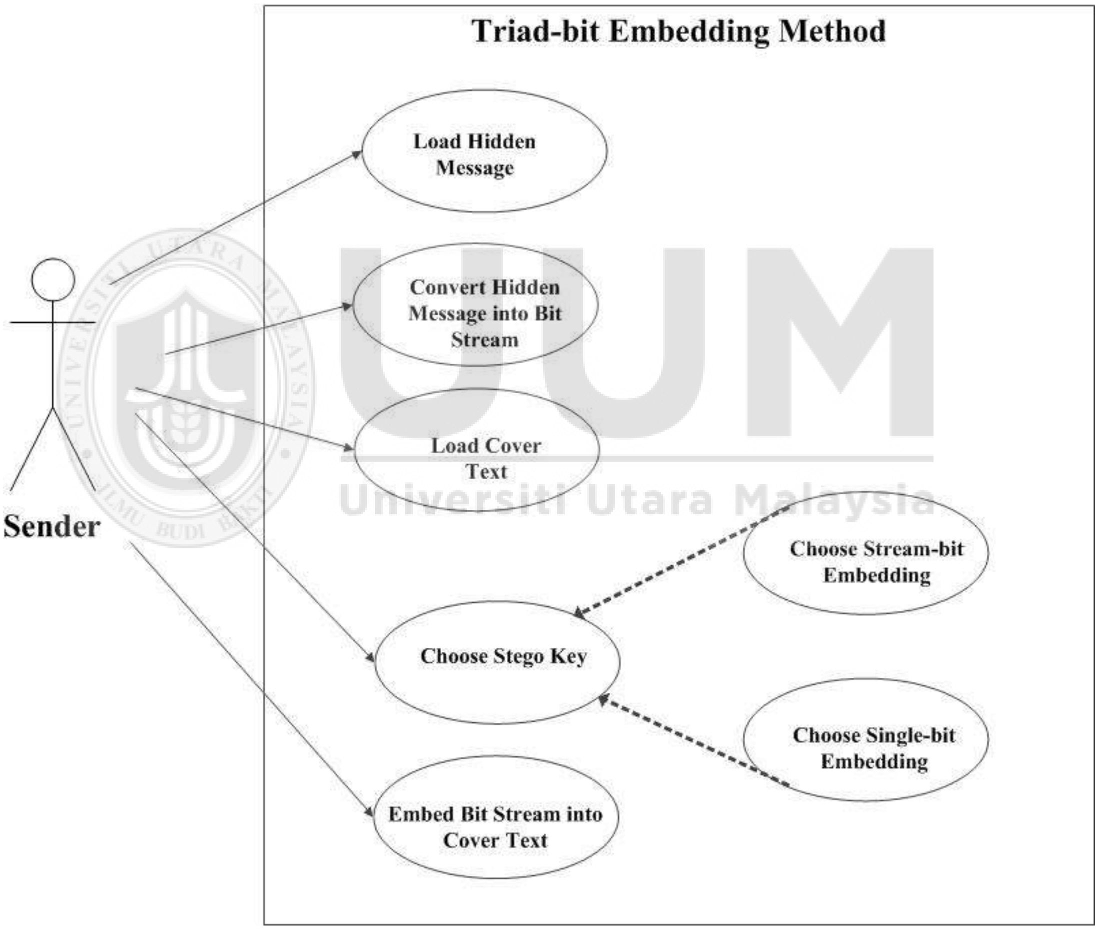

Triad-bit Embedding Method
Load Hidden Message
Convert Hidden Message into Bit Stream
Load Cover Text
Choose Stream-bit Embedding
Choose Stego Key
Choose Single-bit Embedding
Embed Bit Stream into Cover Text
Sender

Figure 4.11: Use Case Diagram for Embedding Triad-bit Method

Table 4.1: Components of Embedding Use Case Diagram

Entity	Description
Sender	An actor entity that interact with the system to perform the embedding process.
Load Hidden Message	The function to load the hidden message file into the system
Convert Hidden Message	The function to convert the hidden message in string type into stream bit type.
Load Cover Text	The function to load the cover text file into the system
Choose Stego Key	The function to choose the stego key that be used for embedding process.
Choose Stream-bit Embedding	The extend function of Choose Stego Key function to select the Stream-bit technique for embedding process.
Choose Single-bit Embedding	The extend function of Choose Stego Key function to select the single-bit embedded technique for embedding process.
Embed Bit Stream Into Cover Text	The function to embed the hidden message into cover text to get the stego text

b. Class Diagram

The class diagram is a static diagram. It represents the static view of the embedding part of TBSS. It describes the attributes and operations of a class and also the constraints imposed on embedding of the TBSS. From Figure 4.11 it can be seen that there are four classes that are vividly shown such as the cover text class, the hidden message class, the stego key class and finally, the stego text class.

The cover text class has two attributes to represent the length and size of the cover text, and it has one operation to represent the loading operation of the cover text. Each cover text embeds only on a hidden message.

The hidden message class has one attribute representing the length of the hidden message, and it has three operations representing the loading, the converting and embedding the hidden message.

The stego key class includes three attributes representing the stego key type, the used letter and the secret key numbers. In addition this class has one operation that represents the chosen of stego key. Every stego key generates on a stego text for every embedding process involved.

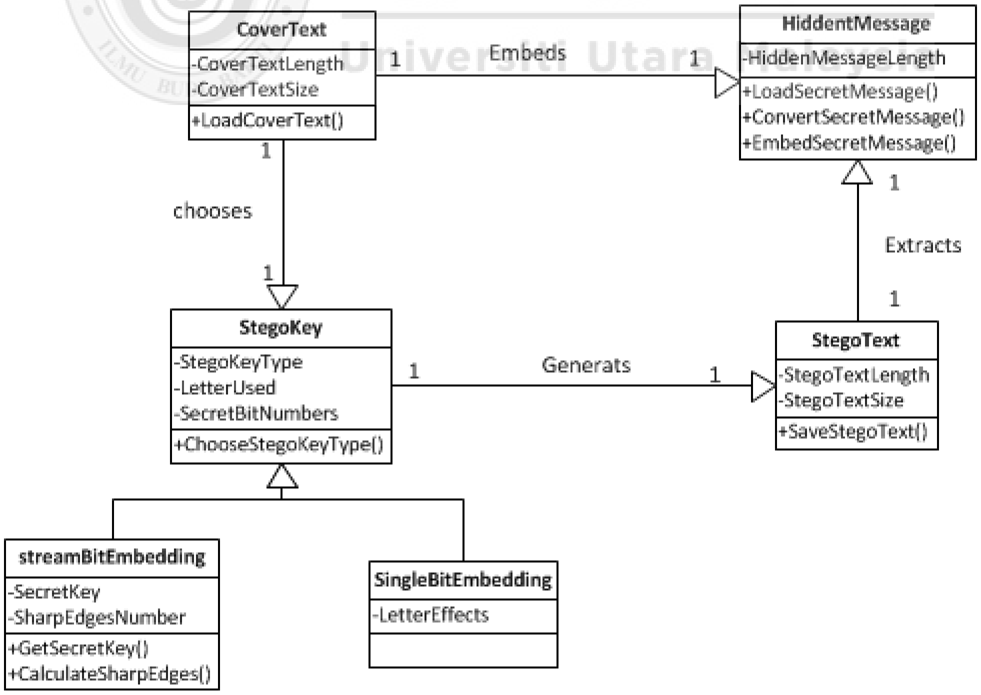


Figure 4.12: Class Diagram of Embedding & Extracting Triad-bit Method

4.6.2 Extracting Logical Design

The logical diagram of extracting process is illustrated above. It is divided into the extracting use case model and extracting class diagram. Below are the explanations:

a. Extracting Use Case Model

The use case model describes the proposed functionality of the extracting part of TBSS system. This use case represents a discrete unit of interaction between a reception (human or machine) and TBSS system. A use case is a single unit of meaningful work to extract the hidden message from a stego text. There are three main use cases and one of them extends to two other use cases as show in Figure 4.13.

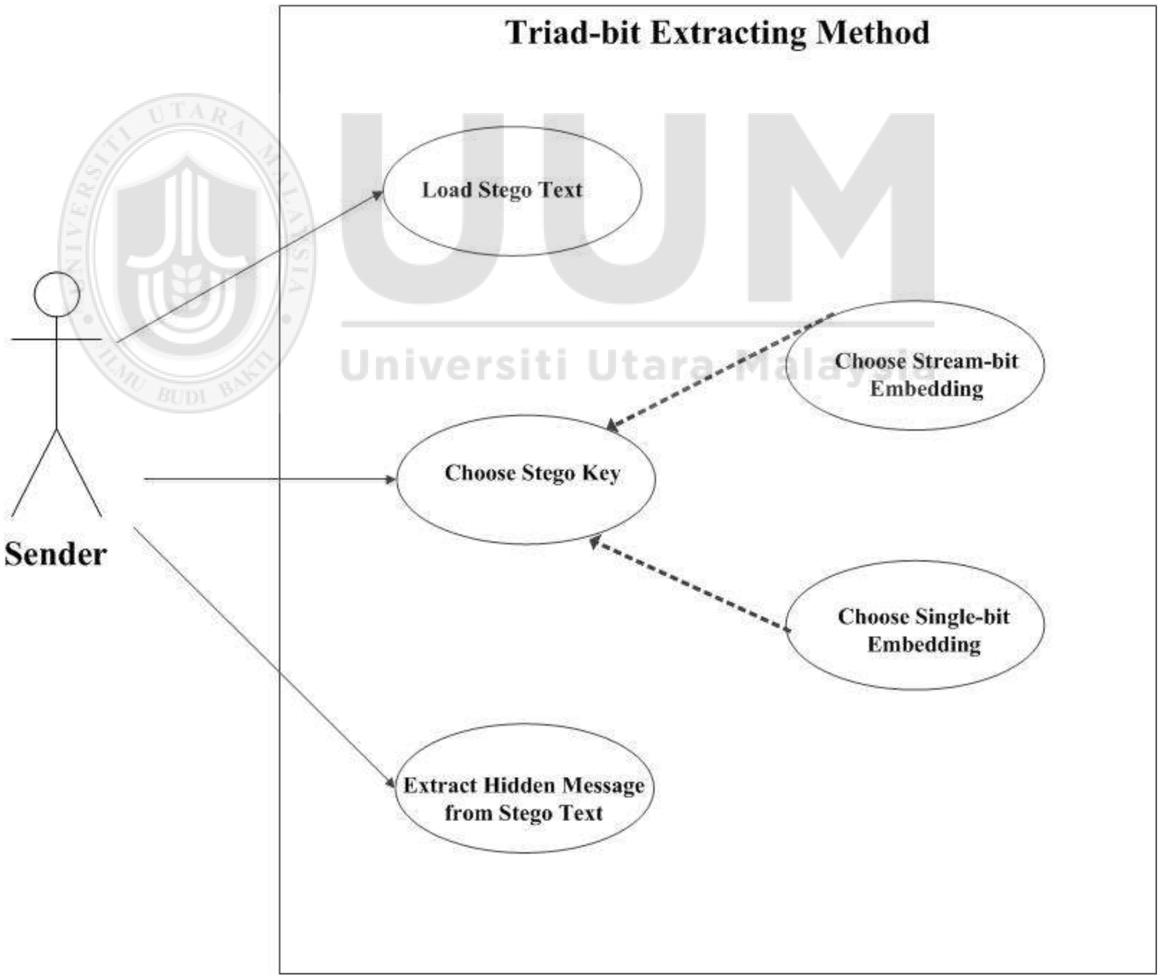

Triad-bit Extracting Method
Load Stego Text
Choose Stream-bit Embedding
Choose Stego Key
Choose Single-bit Embedding
Extract Hidden Message from Stego Text
Sender

Figure 4.13: Use Case Diagram for Extracting Triad-bit Method

Table 4.2: Components of Extracting Use Case Diagram

Entity	**Description**
Sender	An actor entity that interact with the system to perform the embedding process.
Load Stego Text	The function to load the stego text file into the system
Choose Stego Key	The function to choose the stego key that be used for embedding process.
Choose Stream-bit Embedding	The extend function of Choose Stego Key function to select the Stream-bit technique for embedding process.
Choose Single-bit Embedding	The extend function of Choose Stego Key function to select the single-bit embedded technique for embedding process.
Extract Bit Stream from Stego Text	The function to extract the bit stream of hidden message from stego text to get the stego text

b. Extracting Class Diagram

The class diagram is a static diagram. It represents the static view of the extracting part of TBSS. It describes the attributes and operations of a class and also the constraints imposed on extracting of TBSS. From figure 4.13 it can be clearly seen that there are three classes involved such as the stego text class, the hidden message class and the stego key class.

The stego key class includes three attributes representing the stego key type, the used letter and the secret key numbers. In addition, this class has one operation representing the chosen stego key. Every stego key generates on a stego text for every embedding process, and every stego text can choose only one stego key for every extracting process. The hidden message class has one attribute that represents the length of the hidden message, and it has three operations that represent the loading, the converting and embedding the hidden message.

4.7 Summary

This chapter described the triad-bit method. In the beginning of this chapter, the study introduced the process design of embedding and extracting process. The process design illustrated the flow process of embedding and extracting process of triad-bit method. Then, the study explain the algorithm design that be built to embed and extract the hidden message based on the three Arabic steganography embedded techniques; kashida, shifting point and sharp_edges. Then, this chapter presented logical and physical design of TBSS. This system will be used to evaluate the triad-bit method based on five metric as will be explain in Chapter 4.

CHAPTER FIVE
RESULT AND DISCUSION

5.1 Introduction

As elaborated upon in the research methodology that was presented in the previous chapter, triad-bit method integrate tree different embedded techniques to embed the hidden messages in cover text, which. This chapter discusses the results evaluation process of triad-bit method based on five metric. The first metric is evaluate the capability of stego text by comparing the size of stego text and size of expected text. The second metric is usage ratio that evaluate the used character during the embedding process. The third metric is usability, it is refer to how many character can be used to embed the hidden message. The fourth metric is evaluate the fitness performance of triad-bit method. Finaly, the running time of embedding process.

5.2 Size Capability of Stego Text

i. Single-bit1 technique

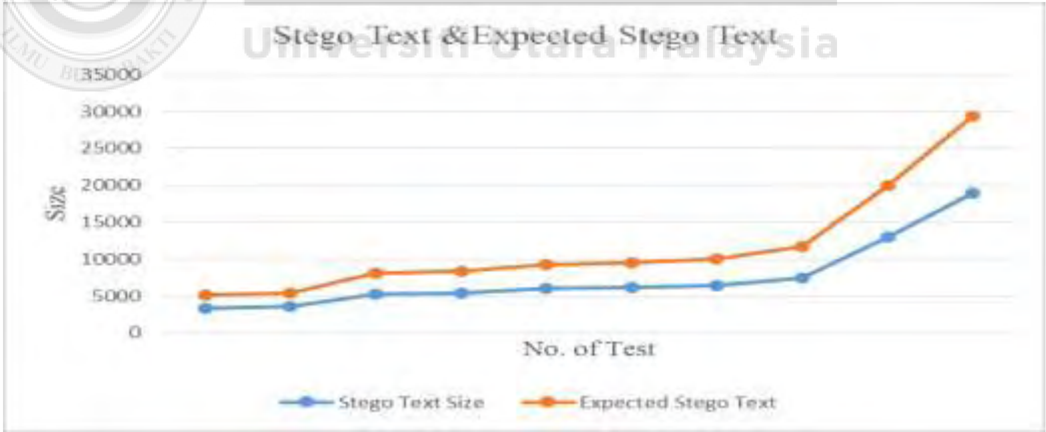


Figure 5.1: Size Comparison of Stego Text and Expected Stego Text by Using Single-bit1 Technique.

As illustrated by the Figure 5.1 the size comparison between the stego text and the expected stego text in Single-bit1 technique. Overall, it can be seen that the size of the stego text system and expected stego text followed a fairly similar pattern when the size of both texts increased steadily. The curve pattern of the stego text and the expected stego text also increased consistently. Hence, it shows that the continuous patterns of the curve are stable.

ii. Single-bit2 embedded technique

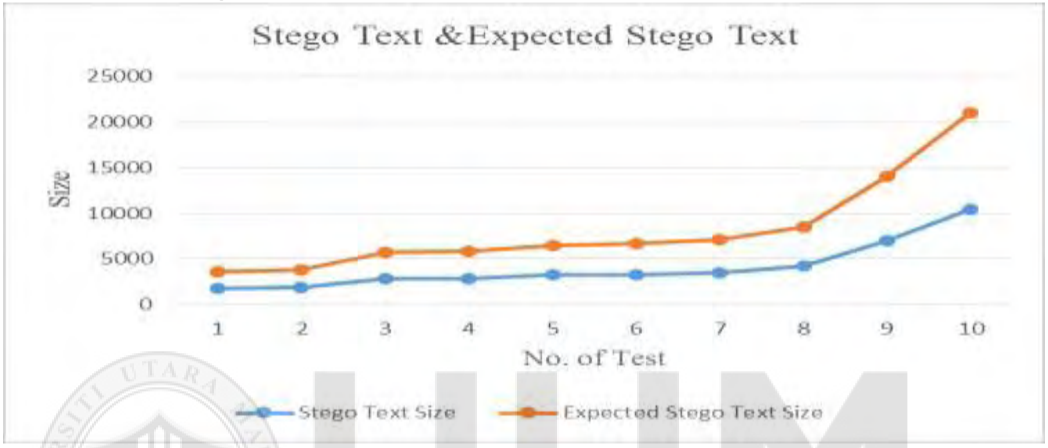


Figure 5.2: Size Comparison between Stego Text and Expected Stego Text by Using Single-bit 2 Technique.

From the Figure 5.2 clearly shows the size comparison between the stego text system and the expected stego text in Single-bit 2 technique. Even though both texts had a steady increase, the expected stego text overruled the stego text system making it reach a higher level than the latter.

iii. Stream-bit technique

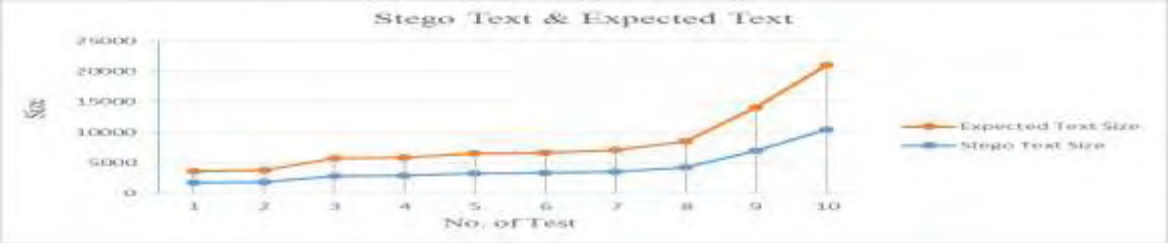
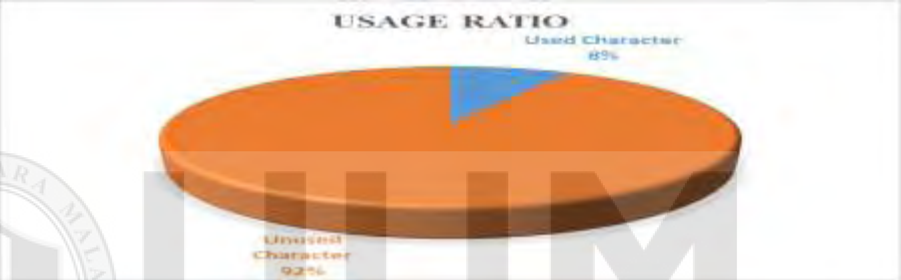


Figure 5.3: Size Comparison between Stego Text and Expected Stego Text by Using Stream-Bit Technique.

From the Figure 5.3 it is clear that the different sizes between the system of a stego text and the expected stego text in the Stream-bit technique are compared. The size of the stego text system and the expected stego text followed a fairly similar pattern by steadily increasing to a different size. The curve patterns of a stego text and an expected stego text are repeated consistently. Hence, it shows that the continuous patterns on the obtained curve are stable.

5.3 Usage Ratio of Cover Text

i. **Single-bit1 technique**



ii.

Figure 5.4: Cover Text Character Usage For Data Hiding Using Single-bit 1 Technique

The Figure 5.4 reflects the cover text usage of data hiding in the single-bit embedded technique. It can be clearly seen that the single-bit 1 only used 8% of the average space in the cover text to embed the hidden message. The largest percentage indicates the space of unused characters to conceal. In overall, it is clear that the single-bit 1 embedded technique has a higher capacity in hiding secret messages, despite requiring only less characters (in the cover text).

iii. **Single-bit2 embedded technique**

iv.

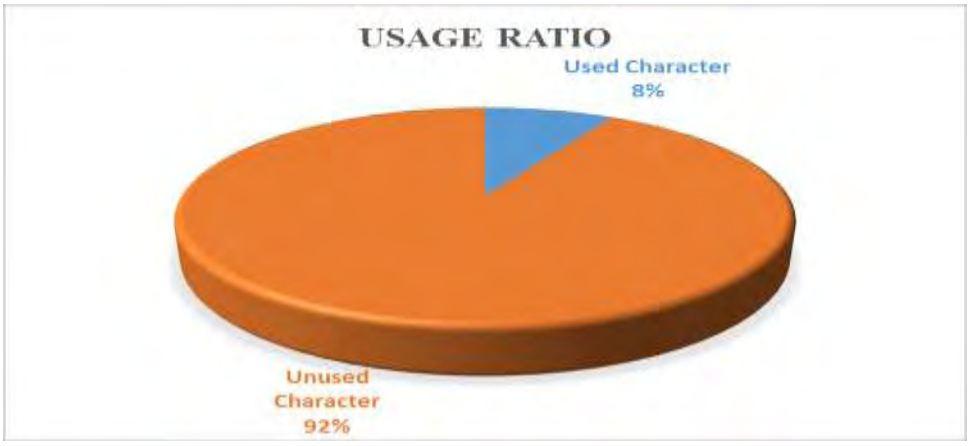


Figure 5.5: Cover Text Character Usage for Data Hiding Using Single-bit Technique

The Figure 5.5 compare the cover text usage of data hiding in the single-bit embedded technique. The used character calculated for 9% average of space in the cover text to embed the hidden message. On the other hand, the larger average is for unused character indicate to unused space to conceal the hidden message into cover text.

v. **Stream-bit technique**

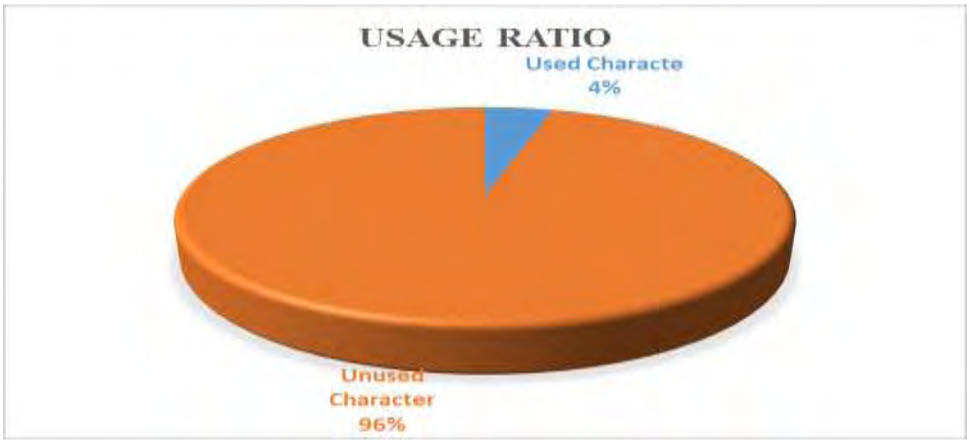


Figure 5.6: Cover Text Character Usage for Data Hiding Using Stream-bit 1 Technique

The Figure 5.6 elucidates the cover text usage of data hiding in the Stream-bit 1 embedded technique. The most significant point in the figure is that the used character had the highest percentage with 96% in the cover text to embed the hidden message. In comparison, the lowest used character is only 4%. All in all, the Stream-bit technique utilizes less Arabic character of the cover text to embed the hidden message.

5.4 Usability Ratio of Cover Text

i. Single-bit1 technique

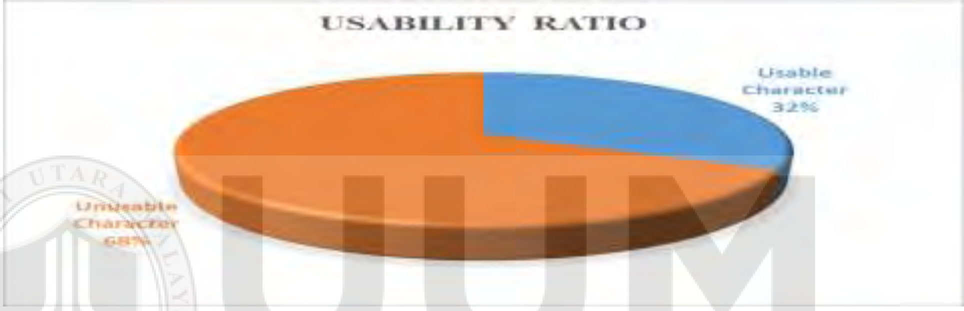


Figure 5.7: Cover Text Character Usability For Data Hiding Using Single-bit 1 Technique

The Figure 5.7 illustrates the cover text usability of data hiding in the single-bit ‘1’ embedded technique. It can be clearly seen that the highest component is represented by the unusable character with 68% to conceal the hidden message compared to 32% for the useable character in the cover text to embed the hidden message. Thus, it can be said that the single-bit ‘1’ embedded technique has a lower usability in embedding the hidden messages, despite the fact that the existing higher characters cannot be used in the embedding process.

ii. Single-bit2 embedded technique

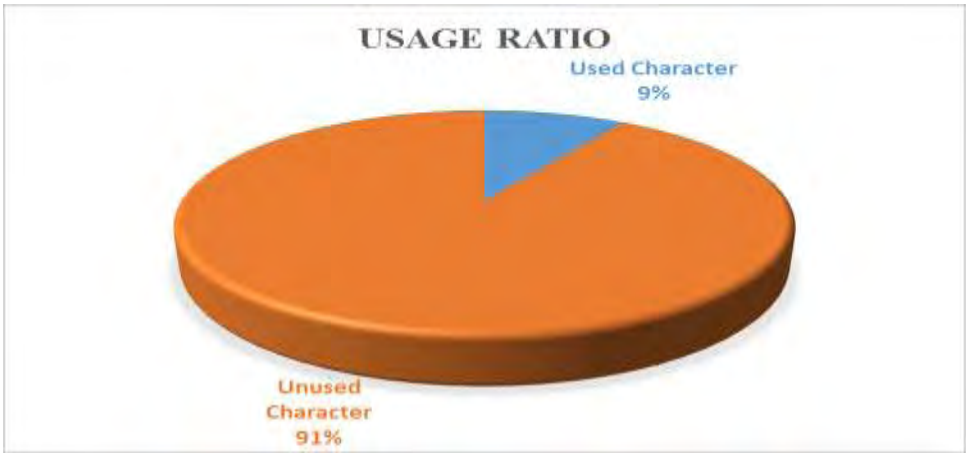


Figure 5.8: Cover Text Character Usability for Data Hiding Using Single-bit 1 Technique

The illustrations depict the cover text usability of data hiding in the single-bit embedded technique. Overall, more than half of the cover text characters is reserved as unusable characters. The usable character is the minor part (39%) of space in the cover text to embed the hidden message. However, the larger part (61%) is for unusable character to hide l the hidden message into cover text. Therefore, the most significant fact that emerged from the figures is that unusable character allocate more than half of the cover text space.

iii. Stream-bit technique

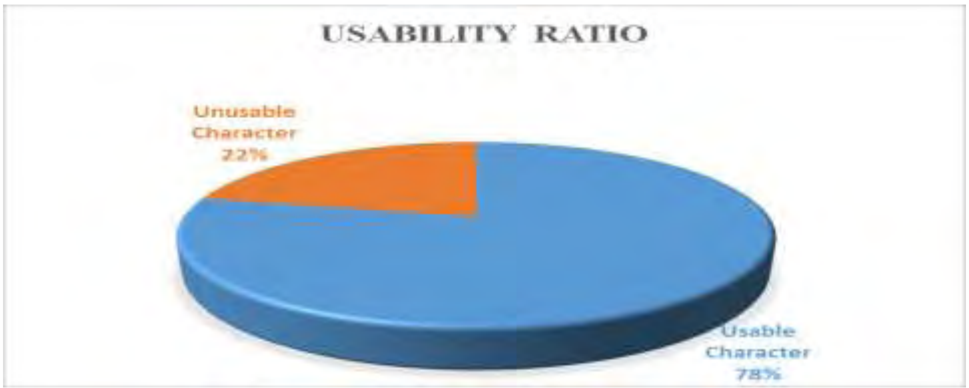


Figure 5.9: Cover Text Character Usability for Data Hiding Using Stream-bit Technique

The above Figure 5.9 provides information on the cover text usability of data hiding in the single-bit embedded technique. It is clearly seen that the single-bit 1 used 78% of the total space in the cover text to embed the hidden message. In contrast, the unusable character used only 22% of the overall space. To sum up, the stream bit embedded technique has a higher usability compared to the previous embedded technique. It can utilize most of Arabic characters to embed the hidden message.

5.5 Fitness Performance of Triad-bit Method

a. Capacity Ratio (CR)

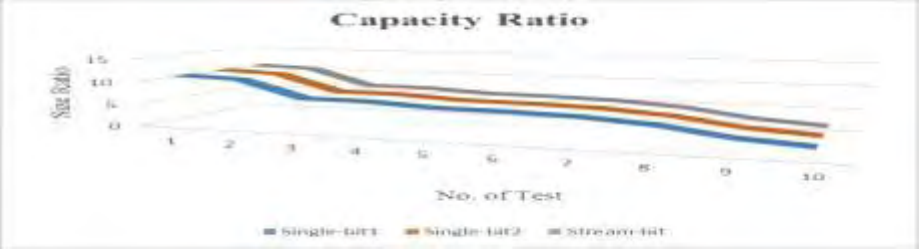


Figure 5.10: Size Comparison between Stego Text and Expected Stego Text by using Single-bit 1 Technique.

From the given Figure 5.10 chart shows the comparison of capacity ratio by using the single-bit 1, single-bit 2 and bit-stream. It is clearly seen that the capacity ratio declined gradually throughout the whole data set, while the curve patterns of capacity ratio are repeated consistently. In short, it shows that the continuous pattern on obtained curve is stable.

b. Embedded Fitness Ratio (ER)

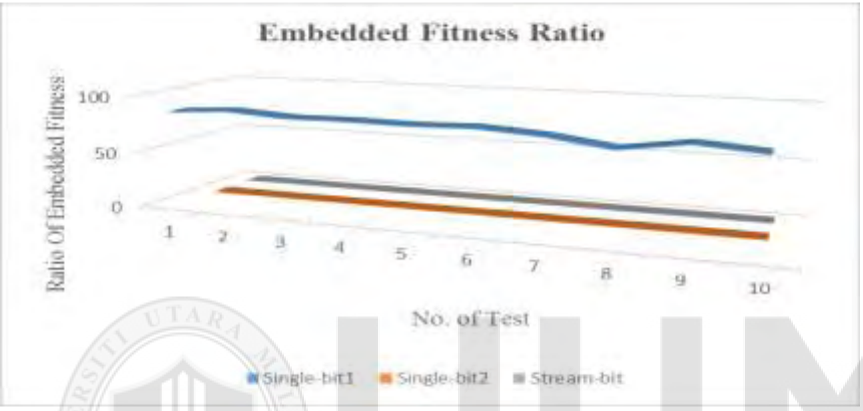


Figure 5.11: Size Comparison between Stego Text and Expected Stego Text by using Single-bit 2 technique

The Figure 5.11 illustrates the embedded fitness ratio for the single-bit 1, single-bit 2 and the stream-bit. It is clearly seen that the embedded fitness ratio of single-bit 2 decreased slightly compared to the other embedded techniques.

c. Saving Space Ratio (SSR)

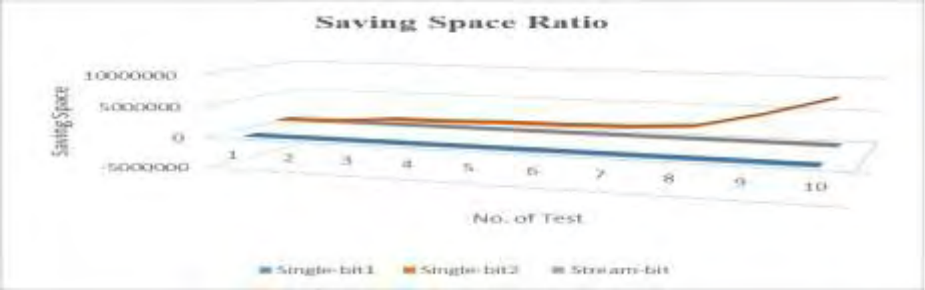


Figure 5.12: Size Comparison between Stego Text and Expected Stego Text by Usingsing Stream-bit Technique.

The above Figure 5.12 shows the saving space ratio for single-bit 1, single-bit 2 and stream-bit. It can be seen that the saving space ratio of single-bit 2 increased gradually than the other embedded techniques.

5.6 Running Time of Triad-bit Method

i. **Single-bit1 technique**

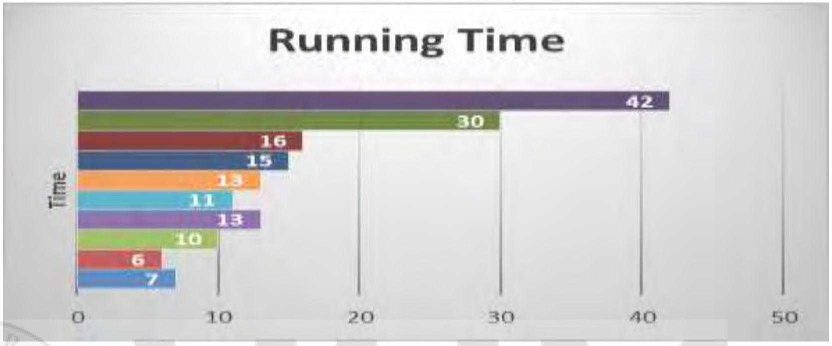


Figure 5.13: The Running Time by Using Single-bit1 Technique

According to the given Figure 5.13 show the running time for embedding the hidden message into the cover text by using the single-bit 1 embedded technique. Units are measured in millisecond. It is obvious that the first embedding spends only 7 milliseconds and it increased based on the increment of the cover text size gradually.

The maximum value is 42 milliseconds for the largest cover text size. The par pattern of the cover text and stego text are repeated consistently. It also shows that the continuous patterns on the obtained curve are stable. Thus, the running time to embed the hidden message increase based on the size of the cover text regardless the hidden message.

ii. **Single-bit2 Embedded technique**

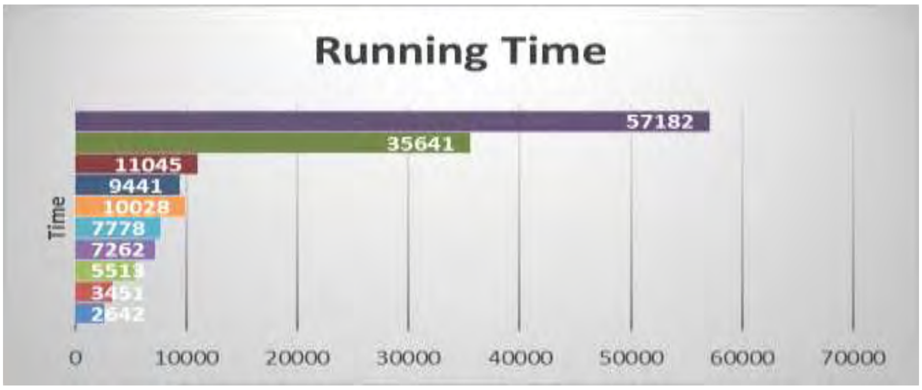


Figure 5.14: The Running Time by Using Single-bit 2 Technique

The given Figure 5.14 shows the running time for embedding the hidden message into the cover text by using the Single-bit 2 technique. Units are measured in millisecond. It can be seen that the first embedding spend 2642 milliseconds and it increased gradually based on the increment of the cover text size. The maximum value is 57182 milliseconds for the largest cover text size. Meanwhile, the par pattern of the cover text and the stego text are repeated consistently.

This shows that the continuous patterns on the obtained curve are stable. Hence, the running time to embed the hidden message by using single-bit 2 is very high compared to using the Single-bit 2 technique for a similar data set.

iii. **Stream-bit technique**

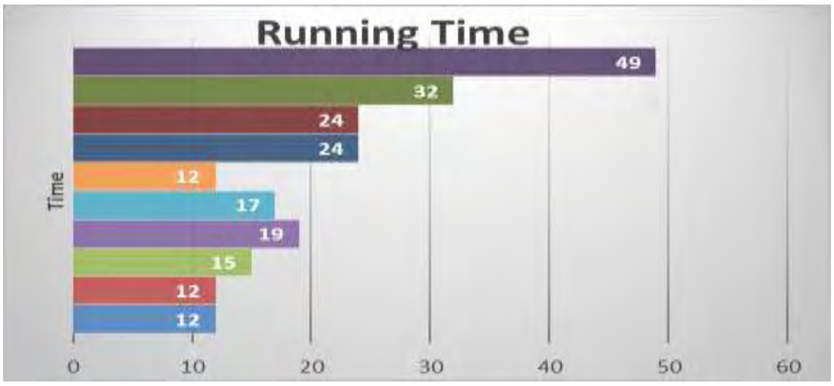


Figure 5.15: The Running Time by Using Stream-bit Technique

The above Figure 5.16 show the running time for embedding the hidden message into the cover text by using the Stream-bit technique. Units are measured in millisecond. It is obvious that the first embedding spent only 12 milliseconds and it increased gradually based on the increment of the cover text size. The maximum value is 49 milliseconds for the largest cover text size. Meanwhile, the curve pattern of the cover text and the stego text are repeated consistently. This shows that the continuous patterns on the obtained curves are stable. In overall, the running time to embed the hidden message increased based on the size of the cover text regardless the hidden message. Also, the running time for the embedding process is fairly similar for the single-bit embedded technique and Stream-bit technique.

5.7 Summary

This chapter is merely about the analysis and discussion on the Triad-bit steganography method. The result obtained from it shows that the size of the stego text is smaller than the size of the expected stego text when this method is used. This is because this method provides an enhancement in the stego text capacity which makes the size of both stego texts increased throughout the whole data set. On top of that, the study found that the Stream-bit technique has a higher capacity compared to the single-bit 1 embedded technique and the Single-bit 2 technique when a similar dataset is being used. The stream-bit used less character, with only 4%, from the cover text to embed the hidden message into the cover text.

The capacity ratio of single-bit 1, single-bit 2 and stream-bit showed an increment over the whole data set. On the other hand, the embed fitness ratio of single-bit ‘2’ declined slightly compared to the single-bit ‘1’ and Stream-bit techniques. This happens because there is no increase in the cover text size after the hidden message is embedded. In comparison, the saving ratio of single-bit 2 has a higher ratio than the single-bit 2 and stream-bit.

The running time to embed the hidden message by using the single-bit 2 is very high and not acceptable compared to single-bit 1. The reason is the single-bit 2 applies an exchange between two fonts when the hidden bit is 1. The single-bit 1 spent a lower running time and becomes an independent font type compared to the single-bit 2 running time. Therefore, the single-bit 1 is inserted to be an extension to the Arabic character when the hidden bit is 1. Similarly, the stream-bit running time to embed the hidden message is low regardless the font type and the extension inserted.

In conclusion, the single-bit 1, single-bit 2 and stream-bit have many similarities in different results although the stream-bit has a better improvement in term of performance. Therefore, this chapter is evaluating the Triad-bit method that meet the objective no. 2.

CHAPTER SIX
CONCLUSION AND FUTURE WORK

6.1 Introduction

This chapter revisits the objectives of this study with the related contribution to the steganography on the Arabic text domain. The chapter continues discussion with summarizes the main contribution of this thesis by referring to the research objectives. This chapter finishes up the thesis by highlighting the limitation and future works of this research that gives the significance to the steganography community.

6.2 Discussion

This section summarizes the finding corresponding to the research objectives listed in Section 1.5 of Chapter 1. There are the discussion of this study:

6.2.1 Revisiting the Research Objectives 1

One of the important contribution in this study is a formalization of integrated approach into steganography method. This enables the utilization of Arabic steganography method. One reason for this success in fact that there is a boundary of solution which the Arabic text steganography method does not evaluate the embedding performance during the capacity improvement. This integrated embedded technique enables the steganographer to utilize the Arabic steganography method that are capable to analyze and evaluate the embedding performance.

Based on the expansion in chapter 2, it is identified that the integrated embedded technique can be utilized in Arabic text steganography method. Based on section 2.2.2.8, this study found that, the most prominent kashida, shifting and sharp_edges are proven to be utilize an integrated embedded technique of embedding Arabic steganography method. Thus, the study has developed an integrated embedded technique called triad-bit method which integrated the kashida, shifting point and sharp_edges. Hopefully, the idea of integrated embedded technique will be utilize by researcher in the future work special in Arabic text steganography.

6.2.2 Revisiting the Research Objectives 2

In this study the analysis and evaluation for embedding performance of proposed method is implemented. The experiment for embedding performance is done through several parameters such as size capability, usage ratio, usability ratio, fitness performance and running time. For the experiment in Chapter 4, it is found that, the result of proposed method stated that the size capability of proposed method is improved because the system stego text size is lower than the expected stego text size. In addition, the proposed method used less character from the cover text such as 8% by using Single-bit1 technique, 9% by using Single-bit 2 technique and 4% by using Stream-bit technique. The study found that the usability ratio of proposed method is high special by using Stream-bit technique (78%). The running time for the proposed method is the embedding time based on the used embedded technique. The study found the single-bit 2 spent longer time to embed the hidden message into cover text. Therefore, the proposed method provide accurate analysis and evaluation for embedding performance in Arabic text steganography embedded technique. Thus, it has shown that integrated embedded technique called triad-bit method.

6.3 Limitation of the Research Work

It is important to state that there are several limitations of this research work which are identified such as:

- **Dataset used**. This research work does not utilize several of datasets. It is only depend on non-fully Arabic diacritic text.
- **Method of keys used**. This research work is depending to embedding method which is provided by authors.

6.4 Contributions

The study produces new method in Arabic steganography domain called triad-bit method. It is the effective way to evaluate the performance of kashid, shifting point and sharp_edges embedded methods. It introduces one solutions for various cases still exist in that steganography embedded technique in Arabic domain. The study provide the steganographer the evaluation knowledge on kashida, shifting and sharp_edges embedded techniques. The study has evaluated the proposed method based on five evaluation metric.

6.5 Future Work

In future, the study will try to add new evaluation formula in order to achieve a good embedding performance and improve the capability of the proposed method.

As a conclusion for this study, this chapter has discussed and concluded the research summary based on the objectives of the study. At the end of the discussion, recommendations for further research were outlined. Overall, this thesis provided a novel triad-bit method by integrate several embedded methods in Arabic text steganography domain.

REFERENCES

Aabed, M. a., Awaideh, S. M., Elshafei, A. R. M., & Gutub, A. a. (2007). Arabic diacritics based steganography. *ICSPC 2007 Proceedings - 2007 IEEE International Conference on Signal Processing and Communications*, (September 2015), 756–759. Ahmadoh, E. M. (2015). Utilization of Two Diacritics for Arabic Text Steganography to Enhance Performance, *3*(1).

Al-Alwani, W., Mahfooz, A. Bin, & Gutub, A. A. A. (2007a). A Novel Arabic Text Steganography Method Using Extensions. *Proceeding of World Academy of Science, Engineering and Technology*, *1*(3), 483–486.

Al-Alwani, W., Mahfooz, A. Bin, & Gutub, A. A. A. (2007b). A Novel Arabic Text Steganography Method Using Extensions. *Proceeding of World Academy of Science, Engineering and Technology*, *1*(3), 502–505.

Alginahi, Y. M., Kabir, M. N., & Tayan, O. (2012). An Enhanced Kashida-Based Watermarking Approach for Increased Protection in Arabic Text-Documents, 1–10.

Al-Haidari, F., Gutub, A., Al-Kahsah, K., & Hamodi, J. (2009). Improving security and capacity for arabic text steganography using "Kashida" extensions. *2009 IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2009*, 396–399.

Alla, K., & Prasad, R. S. R. (2009). An evolution of hindi text steganography. *ITNG 2009 - 6th International Conference on Information Technology: New Generations*, Alnazer, A., & Gutub, A. (2009). Exploit Kashida Adding to Arabic e - Text for High Capacity Steganography Objective y Proposed Approach, (October), 19–21.

Al-Nazer, A., & Gutub, A. (2009). Exploit Kashida adding to Arabic E-text for High Capacity Steganography. *NSS 2009 - Network and System Security*, 447–451.

Ammar, O., Khaled, E., & Miad, F. (2014). Steganography in Text by Using MS Word Symbols.

Ardakani, S. B., Latif, A. M., & Mirzaie, K. (2015). Presentation a new method for Steganography in Persian text of an electronic document, *36*.

Atallah, M. J., McDonough, C. J., Raskin, V., & Nirenburg, S. (2000). Natural Language Processing for Information Assurance and Security: An Overview and Implementations. In *NSPW '00: Proceedings of the 2000 workshop on New securityparadigms*(pp.51–65).

Azawi, A. F. Al, & Fadhil, M. A. (2011).An Arabic Text Steganography Technique Using ZWJ Regular Expression *3*(3), 419–424.

Bensaad, M. L., & Yagoubi, M. B. (2011). High capacity diacritics-based method for information hiding in Arabic text. *2011 International Conference on Innovations in InformationTechnology*,433–436.

Bhattacharyya, S., Banerjee, I., & Sanyal, G. (2011). A Survey of Steganography and Steganalysis Technique in Image, Text, Audio and Video as Cover Carrier. *Journal of Global Research in Computer Science*, *2*(4), 1–16.

Chang, C.-Y., & Clark, S. (2010a). Linguistic Steganography Using Automatically Generated Paraphrases. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, (June), 591–599.

Chang, C.-Y., & Clark, S. (2010b). Turn-taking and affirmative cue words in task-oriented dialogue. *Dissertation Abstracts International, B: Sciences and Engineering*, *70*(8), 4943. Changder, S., Debnath, N. C., & Ghosh, D. (2010). A greedy approach to text steganography using properties of sentences. *Proceedings - 2011 8th International Conference on Information Technology: New Generations, ITNG 2011*, 30–35.

Computing, M., Kumar, R., & Singh, a J. (2015). Understanding Steganography over Cryptography and Various Steganography Techniques, *4*(3), 253–258.

Cvejic, N., & Jic, C. V. E. (2004). *Algorithms for Audio Watermarking and*. *Processing*.

Din, R., & Samsudin, A. (2011). A Conceptual Framework for Natural Language Steganalysis, *1*.

Gaur, M., & Sharma, M. (2015). A New PDAC (Parallel Encryption with Digit Arithmetic of Cover Text) Based Text Steganography Approach for Cloud Data Security, 1344–1352.

Gutub, A. a, Ghouti, L. M., Elarian, Y. S., Awaideh, S. M., & Alvi, A. K. (2010). Utilizing Diacritics Marks for Arabic Text Steganography. *Kuwait Journal of Science & Engineering (KJSE)*, *37*(1), 89–109.

Gutub, A. A. A., Al-Haidari, F., Al-Kahsah, K. M., & Hamodi, J. (2010). E-text watermarking: Utilizing “Kashida” extensions in Arabic language electronic writing. *Journal of Emerging Technologies in Web Intelligence*, *2*(1), 48–55. Gutub, A., Ghouti, L., & Amin, A. (2007). Utilizing Extension Character “Kashida”With Pointed Letters For Arabic Text Digital Watermarking.

Kamel, I., & Banawan, S. (2012). Hiding information in the placement of maneuverable words. *2012 International Conference on Innovations in Information Technology, IIT 2012*, 255–260.

Khairullah, M. (2009). A novel text steganography system using font color of the invisible characters in microsoft word documents. *2009 International Conference on Computer and Electrical Engineering, ICCEE 2009*, *1*, 482–484. Khan, E. A. (2014). Using arabic poetry system for steganography, *6*, 55–61.

Khan, S., & Abhijitha, B. (2015). Polish Text Steganography Method Using Letter Points and Extension.

Khan, S., Sankineni, R., Balagurunathan, P., Suresh, N., & Shree, D. (2015). Czech Text Steganography Method by Selective Hiding Technique, *I*, 1–4.

Liu, G., Ding, X., Su, B., & Meng, K. (2013). A Text Information Hiding Algorithm Based on Alternatives. *Journal of Software*, *8*(8), 2072–2079. http://doi.org/10.4304/jsw.8.8.2072-2079

Malik, S., & Mitra, W. (2015). Hiding Information- A Survey, *3*(3), 232–240.

Memon, J. A., Khowaja, K., & Kazi, H. (2008). Evaluation of Steganography for Urdu / Arabic Text . *Pace Pacing And Clinical Electrophysiology*, 232–237.

Mersal, S. (2014). Arabic Text Steganography in Smartphone, *03*(02), 441–445.

Mohamed, a. a. (2014). An improved algorithm for information hiding based on features of Arabic text: A Unicode approach. *Egyptian Informatics Journal*, *15*(2), 79–87. Odeh, A. (2012). Steganography in Arabic Text Using Zero Width and Kashidha Letters. *International Journal of Computer Science and Information Technology*, *4*(3), 1–11.

Odeh, A. (2015). ROBUST TEXT STEGANOGRAPHY ALGORITHMS FOR SECURE DATA COMMUNICATIONS.

Odeh, A., Alzubi, A., Hani, Q. B., & Elleithy, K. (2012). Steganography by multipoint Arabic letters. *2012 IEEE Long Island Systems, Applications and Technology Conference, LISAT 2012*. Odeh, A., Elleithy, K., & Faezipour, M. (2013). Steganography in Arabic text using Kashida variation algorithm (KVA). *9th Annual*

Conference on Long Island Systems, Applications and Technology, LISAT 2013. http://doi.org/10.1109/LISAT.2013.6578239

Por, L. Y., Ang, T. F., & Delina, B. (2008). WhiteSteg: A new scheme in information hiding using text steganography. *WSEAS Transactions on Computers*, *7*(6), 735–745.

Por, L. Y., & Delina, B. (2008). Information Hiding : A New Approach in Text Steganography, (September).

Prasad, R. S. R., & Alla, K. (2011). A new approach to Telugu text steganography. *ISWTA 2011 - 2011 IEEE Symposium on Wireless Technology and Applications*, 60–65. Rafat, K. F. (2009). Enhanced text steganography in SMS. *2009 2nd International Conference on Computer, Control and Communication, IC4 2009*.

Roslan, N. A., Mahmod, R., & Udzir, N. U. R. I. (2011). Sharp_Edges Method in Arabic Text Steganography, *33*(1).

Roslan, N. A., Mahmod, R., Udzir, N. U. R. I., & Zurkarnain, Z. A. (2014).Primitive Structural Method for High Capacity , *67*(2), 373–383.

Roy, S., & Manasmita, M. (2011). A novel approach to format based text steganography. *Proceedings of the 2011 International Conference on Communication, Computing & Security - ICCCS '11*, 511. Sabir, A. S. (2013). A New Arabic Text (Diacritics , non Diacritics) Steganography, *31*(3), 85–96.

Samphaiboon, N., & Dailey, M. N. (2008). Steganography in thai text. *5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON 2008*, *1*, 133–136.

Shirali-Shahreza, M. (2008). Text steganography by changing words spelling. *International Conference on Advanced Communication Technology, ICACT*, *3*(MARCH 2008), 1912–1913. Shirali-shahreza, M. H. (2007). Text Steganography in chat. *Internet, 2007. ICI 2007.* …. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4401716

Shirali-shahreza, M. H., & Shirali-Shahreza, M. (2006). A new approach to Persian/Arabic text steganography. *Proceedings - 5th IEEE/ACIS Int. Conf. on Comput. and Info. Sci., ICIS 2006. In Conjunction with 1st IEEE/ACIS, Int. Workshop Component-Based Software Eng., Softw. Archi. and Reuse, COMSAR 2006*, *2006*, 310–315. Shirali-Shahreza, M. H., & Shirali-Shahreza, M. (2010).

Arabic/Persian text steganography utilizing similar letters with different codes. *The Arabian Journal for Science and Engineering*, *35*(1), 213–222.

Shirali-Shahreza, M., & Shirali-Shahreza, M. H. (2008a). An improved version of persian/arabic text steganography using “La” word. *Proceedings of IEEE 2008 6th National Conference on Telecommunication Technologies and IEEE 2008 2nd Malaysia Conference on Photonics, NCTT-MCP 2008*, (August), 372–376. Shirali-Shahreza, M., & Shirali-Shahreza, S. (2008b). Persian/arabic unicode text steganography. *Proceedings - The 4th International Symposium on Information Assurance and Security, IAS 2008*, 62–66. http://doi.org/10.1109/IAS.2008.12

Singh, P., Chaudhary, R., & Agarwal, A. (2012). A Novel Approach of Text Steganography based on null spaces. *Iosrjournals.Org*, *3*(4), 11–17. Wilson, a, Blunsom, P., & Ker, a D. (2014). Linguistic steganography on Twitter: Hierarchical language modeling with manual interaction. *Media Watermarking, Security, and Forensics 2014*, *9028*.

Yuling, L., Xingming, S., Can, G., & Hong, W. (2007). RFalse, 2094–2097.

APENDIX A

The Size of Data Set by Using Single-bit1 Technique

Hidden Message Size	Cover Text Size	Stego Text Size	Expected Stego Text
25	1785	3333	1810
25	1871	3552	1896
25	2844	5291	2869
25	2890	5391	2915
25	3249	6033	3274
25	3318	6204	3343
25	3531	6485	3556
25	4237	7481	4262
25	7023	12940	7048
25	10487	18894	10512
29	1785	3167	1814
29	1871	3594	1900
29	2844	5339	2873
29	2890	5433	2919
29	3249	6081	3278
29	3318	6248	3347
29	3531	6537	3560
29	4237	7527	4266
29	7023	12988	7052
29	10487	18942	10516
30	1785	3167	1815
30	1871	3386	1901
30	2844	5317	2874
30	2890	5421	2920
30	3249	6067	3279
30	3318	6234	3348
30	3531	6516	3561
30	4237	7519	4267
30	7023	12968	7053
30	10487	18924	10517
42	1785	3447	1827
42	1871	3652	1913
42	2844	5387	2886
42	2890	5487	2932
42	3249	6141	3291
42	3318	6304	3360
42	3531	6587	3573
42	4237	7595	4279

42	7023	13046	7065
42	10487	18996	10529
52	1785	6325	1837
52	1871	6325	1923
52	2844	5557	2896
52	2890	3720	2942
52	3249	7657	3301
52	3318	19064	3370
52	3531	3513	3583
52	4237	5469	4289
52	7023	6217	7075
52	10487	6374	10539
53	1785	3521	1838
53	1871	3726	1924
53	2844	5479	2897
53	2890	5567	2943
53	3249	6229	3302
53	3318	6388	3371
53	3531	6669	3584
53	4237	7671	4290
53	7023	13128	7076
53	10487	19090	10540
63	1785	3561	1848
63	1871	3780	1934
63	2844	5527	2907
63	2890	5613	2953
63	3249	6281	3312
63	3318	6440	3381
63	3531	6723	3594
63	4237	7723	4300
63	7023	13176	7086
63	10487	19126	10550

<u>The Size of Data Set by Using Single-bit2 Technique</u>

Hidden Message Size	Coever Text Siz	Stego Text Size	Expected Stego Text Size
25	1785	1785	1810
25	1871	1871	1896
25	2844	2844	2869
25	2890	2890	2915
25	3249	3249	3274

25	3318	3318	3343
25	3531	3531	3556
25	4237	4237	4262
25	7023	7023	7048
25	10487	10487	10512
29	1785	1785	1814
29	1871	1871	1900
29	2844	2844	2873
29	2890	2890	2919
29	3249	3249	3278
29	3318	3318	3347
29	3531	3531	3560
29	4237	4237	4266
29	7023	7023	7052
29	10487	10487	10516
30	1785	1785	1815
30	1871	1871	1901
30	2844	2844	2874
30	2890	2890	2920
30	3249	3249	3279
30	3318	3318	3348
30	3531	3531	3561
30	4237	4237	4267
30	7023	7023	7053
30	10487	10487	10517
42	1785	1785	1827
42	1871	1871	1913
42	2844	2844	2886
42	2890	2890	2932
42	3249	3249	3291
42	3318	3318	3360
42	3531	3531	3573
42	4237	4237	4279
42	7023	7023	7065
42	10487	10487	10529
52	1785	1785	1837
52	1871	1871	1923
52	2844	2844	2896
52	2890	2890	2942
52	3249	3249	3301
52	3318	3318	3370
52	3531	3531	3583
52	4237	4237	4289
52	7023	7023	7075

52	10487	10487	10539
53	1785	1785	1838
53	1871	1871	1924
53	2844	2844	2897
53	2890	2890	2943
53	3249	3249	3302
53	3318	3318	3371
53	3531	3531	3584
53	4237	4237	4290
53	7023	7023	7076
53	10487	10487	10540
63	1785	1785	1848
63	1871	1871	1934
63	2844	2844	2907
63	2890	2890	2953
63	3249	3249	3312
63	3318	3318	3381
63	3531	3531	3594
63	4237	4237	4300
63	7023	7023	7086
63	10487	10487	10550

<u>The Size of Data Set by Using Stream-bit Technique</u>

Hidden Message Size	Cover Text Size	Stego Text Size	Expected Text Size
25	1785	1785	1810
25	1871	1871	1896
25	2844	2844	2869
25	2890	2890	2915
25	3249	3249	3274
25	3318	3318	3343
25	3531	3531	3556
25	4237	4237	4262
25	7023	7023	7048
25	10487	10487	10512
29	1785	1785	1814
29	1871	1871	1900
29	2844	2844	2873
29	2890	2890	2919
29	3249	3249	3278
29	3318	3318	3347

29	3531	3531	3560
29	4237	4237	4266
29	7023	7023	7052
29	10487	10487	10516
30	1785	1785	1815
30	1871	1871	1901
30	2844	2844	2874
30	2890	2890	2920
30	3249	3249	3279
30	3318	3318	3348
30	3531	3531	3561
30	4237	4237	4267
30	7023	7023	7053
30	10487	10487	10517
42	1785	1785	1827
42	1871	1871	1913
42	2844	2844	2886
42	2890	2890	2932
42	3249	3249	3291
42	3318	3318	3360
42	3531	3531	3573
42	4237	4237	4279
42	7023	7023	7065
42	10487	10487	10529
52	1785	1785	1837
52	1871	1871	1923
52	2844	2844	2896
52	2890	2890	2942
52	3249	3249	3301
52	3318	3318	3370
52	3531	3531	3583
52	4237	4237	4289
52	7023	7023	7075
52	10487	10487	10539
53	1785	1785	1838
53	1871	1871	1924
53	2844	2844	2897
53	2890	2890	2943
53	3249	3249	3302
53	3318	3318	3371
53	3531	3531	3584
53	4237	4237	4290
53	7023	7023	7076
53	10487	10487	10540

63	1785	1785	1848
63	1871	1871	1934
63	2844	2844	2907
63	2890	2890	2953
63	3249	3249	3312
63	3318	3318	3381
63	3531	3531	3594
63	4237	4237	4300
63	7023	7023	7086
63	10487	10487	10550

Usage and Usability of Cover Text by Using Single-bit1 technique

Usable character	Used Character	Unused Character
540	231	1527
587	231	1630
775	231	2597
828	231	2642
888	231	3001
984	231	3072
873	231	3276
1148	231	3975
1948	231	6783
2928	231	10209
540	263	1495
587	263	1598
775	263	2565
828	263	2610
888	263	2969
984	263	3040
873	263	3244
1148	263	3943
1948	263	6751
2928	263	10177
540	271	1487
587	271	1590
775	271	2557
828	271	2602
888	271	2961
984	271	3032

873	271	3236
1148	271	3935
1948	271	6743
2928	271	10169
540	367	1391
587	367	1494
775	367	2461
828	367	2506
888	367	2865
984	367	2936
873	367	3140
1148	367	3839
1948	367	6647
2928	367	10073
873	447	3060
1948	447	6567
828	447	2426
587	447	1414
1148	447	3759
2928	447	9993
540	447	1311
775	447	2381
888	447	2785
984	447	2856
540	455	1303
587	455	1406
775	455	2373
828	455	2418
888	455	2777
984	455	2848
873	455	3052
1148	455	3751
1948	455	6559
2928	455	9985
540	535	1223
587	535	1326
775	535	2293
828	535	2338
888	535	2697
984	535	2768
873	535	2972
1148	535	3671
1948	535	6479
2928	535	9905

Usage and Usability of Cover Text by Using Single-bit2 technique

Usable character	Used Character	Unused Character
578	200	1558
660	200	1661
894	200	2628
888	200	2673
1019	200	3032
1018	200	3103
1011	200	3307
1233	200	4006
3387	200	6814
3421	200	10240
578	232	1758
660	232	1629
894	232	2596
888	232	2641
1019	232	3000
1018	232	3071
1011	232	3275
1233	232	3974
2287	232	6782
3421	232	10208
578	240	1758
660	240	1621
894	240	2588
888	240	2633
1019	240	2992
1818	240	3063
1011	240	3267
1223	240	3966
2287	240	6774
3421	240	10200
578	336	1422
660	336	1525
894	336	2492
888	336	2537
1019	336	2896
1018	336	2967
1011	336	3171
1233	336	3870
2287	336	6678
3421	336	10104

1011	416	3507
2287	416	6598
888	416	2457
660	416	1445
1233	416	3790
3421	416	10024
578	416	1342
894	416	2412
1019	416	2816
1018	416	2887
578	424	1334
660	424	1437
894	424	2404
888	424	2449
1019	424	2808
1018	424	2879
1011	424	3083
1233	424	3782
2287	424	6590
3421	424	10016
578	504	1254
660	504	1324
894	504	2324
888	504	2369
1019	504	2728
1018	504	2799
1011	504	3003
1233	504	3703
2287	504	6510
3421	504	9936

Usage and Usability of Cover Text by Using Stream-bit technique

Usable character	un-usable character	Used Character
1409	349	118
1508	353	116
2264	564	97
2244	629	118
2617	615	97

2691	612	96
2755	752	120
3082	1124	115
5628	1386	106
8274	2166	114
1409	349	135
1508	353	132
2264	564	112
2244	629	135
2617	615	110
2691	612	110
2755	752	137
3082	1124	132
5628	1386	121
8274	2166	131
1409	349	139
1508	353	137
2264	564	115
2244	629	139
2617	615	115
2691	612	113
2755	752	142
3082	1124	136
5628	1386	126
8274	2166	135
1409	349	190
1508	353	185
2264	564	155
2244	629	190
2691	541	154
691	2612	154
2755	752	193
3082	1124	186
5628	1386	170
8274	2166	184
2755	-997	235
5628	-3767	208
2244	584	233
1508	1365	227
3082	150	228
8274	-4971	224
1409	2098	233
2264	1942	191
2617	4397	191

2691	7749	187
1409	349	237
1508	353	231
2264	564	193
2244	629	237
2617	615	193
2691	612	192
2755	752	240
3082	1124	232
5628	1386	212
8274	2166	228
1409	349	279
1508	353	272
2264	564	227
2244	629	280
2617	615	228
2691	612	225
2755	752	283
3082	1124	274
5628	1386	250
8274	2166	269

<u>Running Time Calculation by Using Single-bit1, Single-bit2 and Stream-bit</u>

Single-bit 1 Running Time	Single-bit 2 Running Time	Stream-bit Running Time
7	2642	12
6	3451	12
10	5513	15
13	7262	19
11	7778	17
13	10028	12
15	9441	24
16	11045	24
30	35641	32
42	57182	49
3	2559	20
6	3483	13
10	5696	18
12	7565	20
11	7532	19
14	9583	14
13	9257	27

15	11565	28
30	35588	37
44	59488	87
4	2625	14
7	3532	15
12	5766	19
14	7501	22
12	8138	22
14	10288	16
14	9883	31
15	11422	30
31	35866	41
42	58672	57
6	2471	19
7	3381	21
10	5525	25
13	7129	30
11	7720	27
14	9741	28
14	9548	40
16	11018	40
30	36197	49
43	56912	80
65	9244	106
83	33411	58
12	6837	34
7	10551	23
17	52631	42
48	2416	88
7	5591	22
35	7578	37
11	10036	40
15	2559	24
22	2738	29
7	3624	27
11	7543	42
12	7668	35
12	9928	34
14	10974	27
15	34524	49
17	56604	48
32	2502	63
43	3440	105
6	2502	28

8	3440	30
11	5268	36
13	7657	42
13	7650	40
14	9616	31
15	9256	59
16	10683	56
31	34367	75
44	56147	116