Journal of Theoretical and Applied Information Technology
20th September 2014. Vol. 67 No.2
© 2005 - 2014 JATIT & LLS. All rights reserved·

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PRIMITIVE STRUCTURAL METHOD FOR HIGH CAPACITY TEXT STEGANOGRAPHY

[1]NUUR ALIFAH ROSLAN, [2]RAMLAN MAHMOD, [3]NUR IZURA UDZIR, [4]ZURIATI AHMAD ZURKARNAIN

[1]Department of Multimedia, FCSIT, UPM, Malaysia-43400
[2] Prof., Department of Computer Science, FCSIT, UPM, Malaysia-43400
[3]Assoc. Prof., Department of Computer Science, FCSIT, UPM, Malaysia-43400
[4]Assoc. Prof., Department of Computer Science, FCSIT, UPM, Malaysia-43400

E-mail: [1]nuuralifahroslan@gmail.com, [2]ramlan@upm.edu.my, [3]izura@upm.edu.my, [4]zuriati@upm.edu.my,

ABSTRACT

High capacity for hiding secret information is typically the main concern in text steganography, as well as robustness and perceptual invisibility in a perfect steganography algorithm. The major issue in text steganography is the difficulty of using a large amount of redundant information to hide secret bits (1 and 0) in perceptible appearance. We propose a Primitive Structural algorithm for Arabic text steganography to encounter this issue. This algorithm hides the secret bits in the primitive structure (i.e. sharp edges, dots, typographical proportion) for the Arabic character. Therefore, this new algorithm presents a high data-embedding capacity since each character has more than one potential place to hide the secret information. The main processes involved are the preparation of the secret message as a binary process, identifying the Primitive Structural for each character in the cover-text process, and finally the bit hiding process. The experiments have shown that the data-embedding capacity percentage is increased up to 4% for our first experiment and for the second experiment the results show the increase in capacity up to 21% compared to the our previous method, thus resolving the capacity issue.

Keywords: *Information Hiding, Data Embedding, Text Steganography, Arabic Text Steganography, High Capacity*

1. INTRODUCTION

Throughout the decades, steganography has become an art of science in hiding secret information, and the digital information today has motivated steganography techniques in a digital way of hiding information. The term 'steganography' is derived from the ancient Greek root words meaning "covered writing". This technique consists of three main elements, namely the secret information, a cover medium, and the stego-media. A basic model for classic steganography is illustrated in Figure 1.

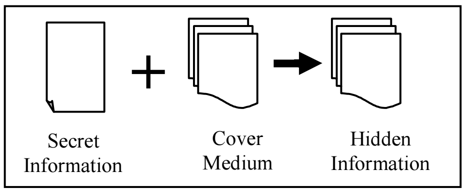

Figure 1: Classic Steganography Model

Based on Figure 1, secret information is confidential information between the sender and the receiver. Meanwhile, a covered medium is any media (whether text, image, audio or video) which will conceal the secret information. Steganography will produce a stego-media which, the stego-media can be publicly exposed without rising any suspicious. The secret information hidden in the media can only be disclosed by the receiver.

This hiding technique usually use an image as a medium of cover, since digitally, image have more redundant information as places for hiding compared to text. A redundant information in text may produce a meaningless sentence or may also affect the structure of the sentence and this will obviously raise suspicious compared to a redundant information in an image which involves pixels— a slight change of pixels in an image (i.e:1204 pixels) would not result in huge or apparent changes for the entire image.

Existing text steganography techniques do not provide high capacity and higher security, even though they have high invisibility (secrecy). We therefore propose the Primitive Structural method, which makes up for the weakness of the previously described method.

This method uses the main primitive structure of the Arabic character such as the sharp edges, dots and the typographical proportion (the calligraphy proportion used for writing Arabic calligraphy) for hiding the secret bits. It uses randomization in placing the secret bits to create a secure algorithm.

The paper is structured as follows: Section 2 presents a background on text steganography, while Section 3 introduces Arabic text steganography and the previous related works. Section 4 describes our proposed method and its implementation. The experiments carried out, along with results and analysis, respectively, are gathered in Section 5. Finally, discussion, conclusion and the future works are laid out in Section 6.

2. TEXT STEGANOGRAPHY

Text steganography technique uses text as a cover medium and this ancient media has now evolved with technology through digitalization. Bennett [1] has classified text steganography into three basic categories, namely format-based, random and statistical generation and linguistic method, as shown in Figure 2.

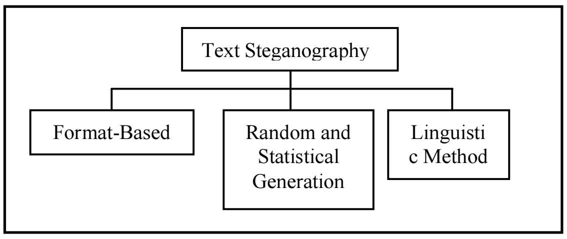

Figure 2: Classification Of Text Steganography (Source: Bennett, 2004)

The format-based method uses the physical text formatting of text as a place to hide information. The terms random and statistical generation refers to the statistical properties, which is based on the character sequences and word sequences. The final method specifically refers to the linguistic properties, which frequently uses linguistic structure [2] as the place for hiding information.

However, Bergmair [3] interpreted linguistic steganography by excluding techniques that directly operate on typesets or written text such as a graphic, speech as a waveform, and also excluded techniques that rely on specific file formats such as ASCII (American Code for Information and Interchanges) or HTML (Hypertext Markup Language).

Therefore, in our review we narrowed the classification of text steganography into three main categories, namely linguistic method, non-linguistic method, and programming language-based method. Figure 3 illustrates our proposed categories.

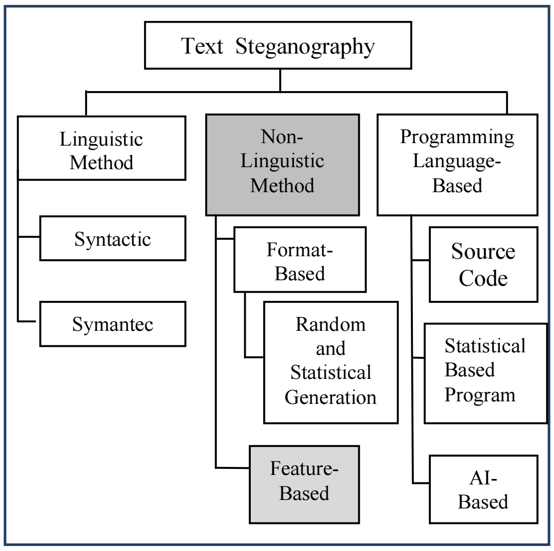

Figure 3: New Categories of Text Steganography

To provide redundant information with high capacity in hiding secret information without raising suspicions, we narrowed down our research focus to the feature-based method. The feature-based method has recently gain attention among the researchers and has been implemented in many writing scripts such in Arabic [4], [5],[6], [7], [8] and [9] Persian [10], [11] and [12], Urdu [13], Japanese [14], Korean [15], Chinese [16], Hindi [17] and Thai [18]. Our research focus is on Arabic text steganography.

3. ARABIC TEXT STEGANOGRAPHY AND PREVIOUS WORKS

Since 2006, Arabic text steganography research has been actively carried out. Many approaches attempt to use the Arabic characters such as the enhancement of dots using the multipoint approach

[4], kashidah [19], diacritics [20] , La [21], pseudo-space approach [22] and finally sharp-edges [9]. Recently, researchers have been attempting to enhance their methods and there are many more properties in Arabic script to be discovered and exploited. The following sections discuss previous approaches to Arabic text steganography.

3.1 Dot Approach

These methods [11] hide secret information in the points (dots) location within the pointed (dots) letters. The length of the secret information is identified (example 20 bits) and compressed. The cover medium text is scanned line by line, character by character.

Whenever a pointed letter is detected, its points (dots) location may be affected by the hidden information bit. If the hidden bit is one, the point (dot) is slightly shifted up; otherwise the concerned cover-text character point location remains unchanged.

Figure 4 shows point (dot) shifting for the Arabic letter "Fa". This method has the advantage in capacity and secrecy. The other remaining characters are also changed randomly to allow more secrecy for this method. Nevertheless, the main weakness for this method is the lack of robustness. The hidden information is lost in any retyping or scanning. Moreover, output text is fixed for the use of only one font type.

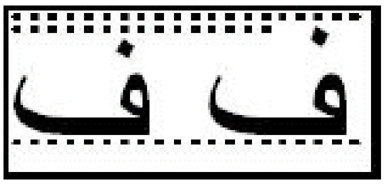

Figure 4: Points (Dots) Shift-Up For Arabic Letter "Fa"

Odeh [4] presented a multipoint Arabic characters for hiding information. In the Arabic language, there are five multipoint letters; each character can be used to hide two bits to determine the shifting and distance between letter points. Even though this approach has a high capacity for hiding bits and is less suspicious compared to the Dot Approach, it also has disadvantages—the retyping process will remove all the hidden data.

3.2 Arabic extension character (kashidah) Method

The Arabic extension character proposed by Gutub [23] exploits the existence of the redundant Arabic extension character (kashidah) and the pointed (dots) letters. This method is more practical: the pointed letters with a kashidah hold the secret bit "1" and unpointed (without dots) letters with a kashidah hold "0". The character extension has a standard character hexadecimal code of 0640 in Unicode system; this method does not have any effect on the writing content. Through our review, this method currently improvised in security part which having a collision in the secret bits position.

3.3 La Word Method

The La Steganography method [21] uses a special form of La word (a combination of the Lam and Alef characters) for hiding the data by inserting an Arabic extension character between the Lam and Alef. For hiding bit 0, they use the normal form of La, whereas bit 1 is hidden using the special word La with a unique code in the Standard Unicode (i.e. FEFB in the Unicode hex notation).

This method is not limited to electronic documents (e-documents) and may be used on printed documents. However, the use of La word is generally limited in Arabic sentences and this affects the capacity to hide information. Also, too-frequent use of La word will raise suspicions about the stego-text.

3.4 Pseudo-Space Approach

The pseudo-space and pseudo-connection character method [22] hides one bit in each letter: the method will examine whether the letter of a word is connected to the next letter. If it is, they insert a ZWJ (Zero Width Joint) letter between the two letters to hide bit 1, adding nothing for hiding bit 0.

Because the letters are connected, adding ZWJ for connecting the letters together does not have any effect on the appearance of the text. The drawback of this method is that removing the Unicode of ZWJ also affects the secret information.

3.4 Diacritics

The method proposed by Gutub [20] exploits the Arabic diacritics. There are eight diacritics (additional marks used in Arabic text to assist in pronouncing the words) in Arabic text as shown in Figure 6. In this method, the cover-text is assumed to be a fully diacritics text. The bit '1' is kept in selected diacritics and the bit '0' is kept in the nondiacritic character, while the other diacritics remain unused. This method provides high capacity but low invisibility as it attracts the attention of the reader. Furthermore, this method also needs fully diacritical text, although most Arabic text contains no diacritics.

Figure 6: Arabic Diacritics

3.5 Similar Letter with Different Codes
Shirali-Shahreza's (2010) presented a new method related to Unicode. They use the Arabic characters of Ya (ی) and Kaf (ک) to hide the secret bit (1 and 0). Both of them have a similar shape but different Unicode representation for the related characters in Arabic and Persian. From this research, we found that Arabic characters are mostly inherited by the other languages such as Persian, Urdu and Jawi (traditional Malay scripts).

These scripts, which are similar to Arabic, have huge potential to hide more secret bits since each of them have additional characters to suit their own language pronunciation, e.g. Jawi has additional six characters (چ, ڠ, ڬ, ڤ, ڽ, ۏ). However, this method has low capacity for hiding the secret bits since they are restricted to hiding in two characters only.

3.6 Sharp-Edges Method
Our previous work, namely the Sharp Edges method [9], is a recent approach to Arabic text steganography. This method uses sharp-edges in Arabic characters to hide the secret information (bits 1 and 0). Figure 7 illustrates examples of sharp-edges in Arabic characters. This method has a high capacity, with high invisibility for bit hiding. In this method, we introduced keys for positioning the secret bit.

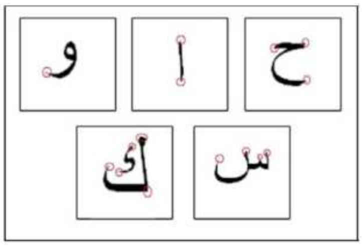
Figure 7: Arabic Character's Sharp-Edges

The different number of sharp-edges shows the possibility to hide the secret bits 1 and 0. The character with the number of sharp-edges 1 is possible to hide the secret bit in two possibilities at the sharp-edges position, which is either hiding secret bit 0 or 1. Meanwhile, if the number of sharp-edges is two, secret bit position is in four possible conditions; 11, 10, 00 or 01.The advantages of this method are perceptual transparency and they solve the retyping problem inherent in the previous works.

Through experiments using the same amount of data, we have shown that the sharp-edges method has the highest capacity in hiding the secret bits compared to Pseudo-Space approach, dot approach, and La word method. However, we identified drawbacks in our previous methods related to the limited character usage in the cover-text, and a drawback in security as the random position for the sharp edges method are limited to only odd and even inputs of keys.

The randomness in positioning the secret bits which provides high security for text steganography is important in having a robust steganography algorithm. Therefore, we enhance our previous work by increasing the capacity for hiding secret message with security improvement by presenting the primitive structural method.

4. PRIMITIVE STRUCTURAL METHOD

Primitive structural method has three main entities, specifically the sharp-edges, the dots, and the typo proportion. The sharp-edges are defined as the sharp starting point and the sharp end point of an Arabic character [9]. The dots entity refers to the dots in the Arabic characters. There are 15 Arabic characters with dots—one, two, or three dots. The typo proportion was adopted from the calligraphy writing method. A rhombic dot is the shape formed when a calligrapher presses their pen to a paper in one downward motion, producing a diamond shape.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 8 illustrates examples of proportion for the Arabic character. Each Arabic character has fixed calculation proportion [24]. As the digital typography for writing Arabic document uses the Nasakh style of writing, we use the Nasakh proportion based on the vertical writing guide line.

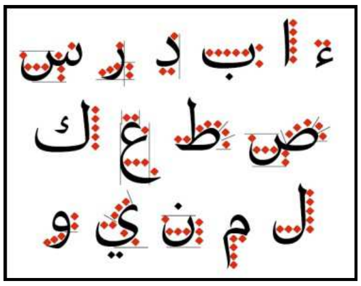

Figure 8: Arabic Character Typo Proportion

Each Arabic character has its primitive structure as mentioned earlier. Table 1 shows several Arabic characters and their characteristic properties.

Table 1: Arabic Characteristic Properties

Char	Unicode	Primitive Structural Method Entities			Num. Of Potential Hiding Place
		Sharp-Edges	Dot (s)	Typo-graphical Proportion	
ا	\U0627	2	0	1	3
ب	\U0628	2	1	3	6
ت	\U062A	2	2	3	7
ث	\U062B	2	3	3	8
ج	\U062C	3	1	5	9

Based on Table 1, each entity has their potential number of places for hiding the secret information, and each hiding place corresponds to a bit of the secret message. Therefore, for each character there is more than one place to hide the secret bits. For example, the character ت (ta) has seven potential hiding places which means we can hide seven secret bits in character ت (ta).

Primitive structural method has three main processes; there are two inputs which will be processed concurrently, and the bit hiding process as the main process of the algorithm. Figure 9 shows a block diagram of the proposed method for data hiding.

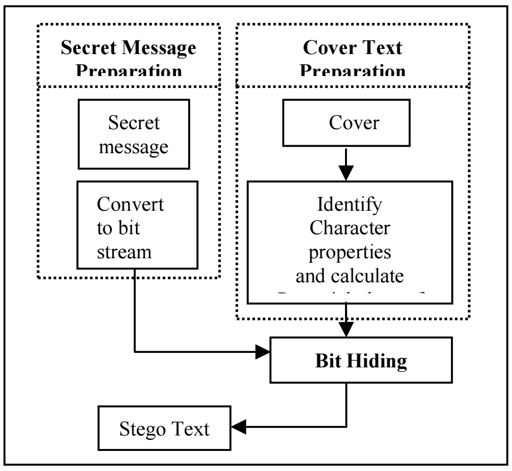

Figure 9: The Block Diagram Of The Proposed Method For Hiding Data

4.1 Secret Message Preparation Process

The secret message preparation involves Latin character as the input. The secret message is converted to binary string and the secret bits will go through length calculation. Then, the results are validated in the bit hiding process. Figure 10 below shows a pseudo-code of the algorithm.

```
Input:  M :message, Latin alphabet with ASCII encoding
Output: Mb: message, binary string

while inputs having characters do
  if message contains white space (" ") and  Arabic
     Character (["^\\u600-\\u06FF]"]) remove text contains
     (" ") and  replace any character with  coding["^\\u600-
     \\u06FF]"]  to ("")
       else end if    set message to strBytes = Mb
                      Calculatet the bit length, Mb.getLength
end while
```

Figure 10: Secret Message Preparation Algorithm

4.2 Cover Text Preparation Process

The cover-text will be restricted to Arabic characters using the Arabic Unicode (range: 0600-06FF). Through this process, the algorithm should recognize the Arabic character and display it in an isolated form for the entire cover-text. Figure 11 shows pseudo-code for the algorithm.

```
Input:  C : cover text, Arabic form character with
            encoding (U600-U06FF )
Output:  CP: the number of cover text potential
               hiding place
while inputs having characters do
     if message contains white space (" ") and
        Alphanumeric Character [^a-zA-Z0-9] remove
        text contains (" ") and replace any Alphanumeric
        Character [^a-zA-Z0-9] to ("")
        else end if
           Calculate the bit length, C.getLength
             for within C.
                     get Length
                     Identify the Arabic Characters
                     Calculate Potential Hiding Place,
                     CP.getLength
             end for loop
end while
```

4.3 Bit Hiding Process

Next, based on the results from both input preprocessing, there is a validation process to ensure the cover text length is more than the secret bits length before proceeding to the hiding process. By referring to the secret bits and the cover text, the secret bit will be placed based on the primitive structure of the character. Figure 14 shows the bit hiding algorithm, and Figure 12 illustrates pseudo-code for the algorithm.

```
Input:  C: Cover text ; Mb: Message in binary & CP:
Cover Potential hiding place
Output: St: Stegotext
while inputs having characters do
            if C contains white space and Alphanumeric
            Character [^a-zA-Z0-9] remove text contains
            white space and replace any Alphanumeric
            Character [^a-zA-Z0-9] to ("")
            else end if
          Calculate the bit length, C.getLength
                  for within C.getLength
                  Identify the Arabic Characters
                  Calculate Potential Hiding Place,
                  CP.getLength
                  end for loop
if Mb > CP
        Identify primitive structure  of character in C
        Read Mb concatenate string based on CP
end while
```

Figure 12: Hiding Bit Pseudo Code

Using our method, all characters were fully used as potential places for hiding the secret bits. Each character has a total potential place of hiding referring to their characteristics. This means each character has more than one bit to hide the secret bits. This results in a huge capacity for hiding the secret bits randomly. To retrieve the secret message, we will reverse the process. Figure 13 shows the retrieving process block diagram.

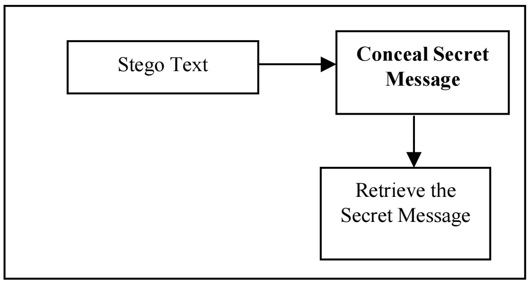

Figure 13: The Block Diagram Of The Proposed Method For Retrieving Data

4.4 Implementation of Primitive Structural Method

The algorithm implementation was done by creating a stego-char program using Java. There are two main modules: the hiding and retrieving part. The hiding module used by the sender needs to prepare a shared secret key, secret message (in Latin), and cover text (Arabic Text). Meanwhile, the retrieve module will be used by receiver to extract the secret message.

First, the stego-char converts the secret message input from ASCII to a binary string and the program will calculate the length of the secret message in bits. Next is the preparation of the cover text which is in Arabic text document or Arabic text. Based on the cover text, the capacity of the hiding place will be determined based on the Primitive Structural for each Arabic character. As previously mentioned, each Arabic character has its own number of potential hiding places.

Therefore, based on the number of potential hiding places the secret bits string is placed according to the appropriate length. For example, for a secret message of 16 bits long (i.e. 10100101 10100101) the total potential hiding place for character ba "ب" is 11. Then, we take 11 bits from the secret bits from right to left, as highlighted in bold bits below:
1010**0101 10100101**

Here, the character ba "ب" will hold secret bits 0101 10100101 and the algorithm will repeat the process for the next cover text and the next secret bits until the entire secret message is hidden into the cover text. Stego-char will produce a stego-text with the shared secret key will be given to the receiver to retrieve back the secret message.

The Primitive Structural algorithm has security element by using the random position. The random positions are based on the categories of shapes in

Arabic character. Arabic characters can be classified into four categories based on their shape or figure. The categories are *Basitah*, which are characters shaped like the letter C (د,ذ,ر,ز,و), *Maujauwaqaf*, characters with an open gap shape (ب, ت,ث,س,ش,ص,ض,ف,ق,ن,ي,ه), *Mustadirah,* characters which are cursive and circular (ج,ح,خ,ع,غ), and finally is *'Umudiyah,* which are characters having a vertical stroke (ك, ا ,ل, م,ط, ظ) [24] .

Based on these categories, we randomly place the secret bits in our algorithm to achieve a robust steganography algorithm. In order to retrieve the secret message, the receiver needs to input a stego-text; through the reverse algorithm the secret message will be retrieved. However, for this paper we focus on the capacity measurement only, while the evaluation for the security element will be part of our future work.

4.7 Experiments and Capacity Results Analysis

As this paper focuses on the capacity measurement for text steganography method, we have conducted two experiments for achieving two different objectives in capacity measurement. The first experiment compares the Primitive Structural method with the Sharp-Edges method using different approaches in preparing the secret message and the cover text. The second experiment is a comparison of our method with other previous works such as Pseudo Steganography, Dot Steganography, and La Steganography methods.

4.7.1 Experiment 1: Hiding Capacity Percentage Comparison With Sharp Edges Method

This experiment consists of two approaches in comparing with our previous work, Sharp-Edges [20]. The first approach uses a fixed length (bits) of secret messages and different lengths (bits) of cover text. Using this approach, we can compare both methods by looking for which one having the highest percentage in hiding the secret message. Meanwhile, the second approach in the experiment involves a fixed length (bits) of cover text and different lengths (bits) of secret messages. We compared both methods by looking for the least percentage in used character for hiding the secret message in the fixed cover text. The details of both approaches are described below.

(a) First Approach: Fixed Secret Message with Different Cover Text

In this experiment, we used a fixed secret message with different lengths of cover-text; for the second approach we used a fixed cover text with different lengths of secret messages. The data set for this approach, which consists of the secret message and the cover text, are taken from Adnan's [23], and the secret messages are from the English translation of Yusuf Ali for first chapter in Al-Quran, which contains seven versus.

Meanwhile, for the cover text, we collected 15 Friday's speech of Ibn Othaimen, the Islamic Scholar which can be retrieved from his official website (ibnoothaimeen.com). Table 2 shows five out of 15 implemented data and the results.

Table 2: Different Cover Text Lengths

Cover Text	**Length of Cover Text (Bits)**
C1	12166
C2	10554
C3	9107
C4	8899
C5	14880
C6	11448
C7	10112
C8	9886
C9	9321
C10	9897
C11	10200
C12	10654
C13	9124
C14	10455
C15	7323

The capacity percentage for potential hiding place for the secret message is calculated using the following equation:

$$\text{Capacity (\%)} = \frac{\underline{\textit{Potential Hiding Place}}}{\textit{Cover Text length (bit)}} \times 100$$

The result from the first approach is illustrated in Figure 14. This experiment involved a fixed length of secret message and various lengths of cover text, which provides various capacities and potential places to hide the fixed secret message in bits.

Through the experiment results in Figure 2, the primitive structural method has larger capacity compared to sharp edges method. This experiment shows the Primitive Structural method has an average of 4% more potential hiding place compared to the Sharp Edges method.

(b) Second Approach: Fixed Cover Text with Different Secret Message

The second approach in our first experiment focuses on the focuses on the percentage measurement of the used character in the cover text for hiding the secret bits. The data consists of verses of English translation of Surah Al-Fatihah as separate (and therefore different secret messages) and the last speech of the prophet Muhammad SAW (in Arabic text) as the fixed cover text. The following Table 3 shows the various secret message lengths and Figure 3 illustrates the results from the experiments.

Table 3: Seven Secret Text Messages With Different Lengths

Secret Text	Length of Secret Text (Bits)
S1	38
S2	47
S3	24
S4	24
S5	32
S6	20
S7	87

The capacity percentage for the used character in the cover text is calculated using the following equation:

$$\text{Capacity (\%)} = \frac{\textit{Used Character for hiding}}{\textit{Cover Text length (bit)}} \times 100$$

The results from the first approach are illustrated in Figure 15 and Figure 16 below:

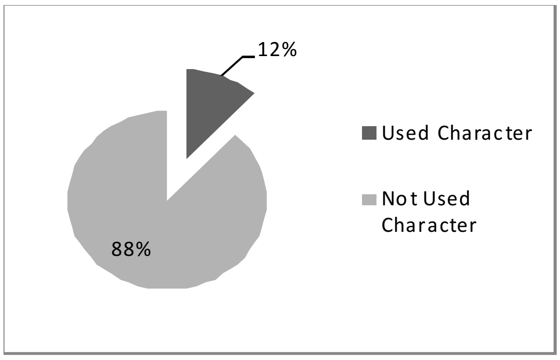

Figure 15: Cover Text Character Usage For Data Hiding Using Primitive Structural Method.

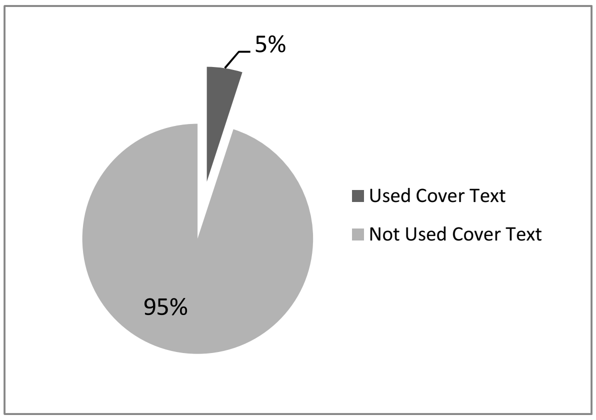

Figure 16: Cover Text Character Usage For Data Hiding Using Sharp Edges Method.

Figures 15 and 16 represent the percentage used by the fixed cover text in hiding various lengths of secret messages. The results for both experiments show that the Primitive Structural method only used 5% average of space in cover text to hide the secret message, while the Sharp Edges method needed an average of 12% of cover text to conceal the secret message. The smaller percentage indicates less need for cover text for the same length of secret message.

Through both approaches for this first experiment it is shown that the Primitive Structural method has higher capacity in hiding secret messages, despite requiring only less characters (in the cover text).

The next section will describe the second experiment for capacity measurement, which is a comparison of the Primitive Structural method against other previous methods.

4.7.1 Experiment 2: Comparison with the previous works

The next conducted experiment is the comparison of Primitive Structural method with the previous methods, i.e. the pseudo steganography method, dot steganography method, La steganography method and sharp edges method.

To compute the capacity with different text sizes the data for the experiments were selected from online-newspapers published in March 2013 as shown in Table 4 below.

Table 4: The Online Newspaper For The Experiments Data

Website Address	Text Size (Kilobyte)
Www.Hamshahri.Net	6.82
Www.Jamejamdaily.Com	3.84
Www.Javandaily.Com	8.03
Www.Kayhannews.Ir	2.92

The capacity measurement is calculated based on author's previous works [9] . As our method has *t*, i.e. the Primitive Structural for hiding each secret bit in *c* cover-text file (kilobyte), therefore the hiding capacity of our method as bit/kilobyte is:

$$\text{Hiding Capacity} = \frac{t\ \text{(indicate each bit)}}{c\ \text{(kilobyte)}}$$

Based on the results, our method has an average 80% of hiding places for secret bits per 1 kilobyte cover text. Table 5 shows the capacity measurement compared to the previous methods. There should be 8 pieces of data from the online newspaper. However, we could only retrieve the data from these four online newspapers since most of the other data have not been available recently.

The results have been simplified in Figure 17, showing the comparison of capacity for each method. Based on Figure 13, our method increases 21% in capacity compared to the sharp-edges method, 52% compared to Pseudo-Steganography method, 86% compared to the Dot steganography and up to 99% compared to the La steganography. Therefore, our proposed algorithm presents higher capacity for hiding the secret bits for Arabic text steganography.

Compared with Sharp-Edges, the newly proposed algorithm is independent from any reference table in order to place the secret information. We proposed a random position by randomly choose the character based on the main structure of the character itself. Using this approach, there may be many random positions within the cover text compared to Sharp-Edges, which restricts the positions to only the odd and even numbers calculated by the shared-key input.

Since our algorithm uses the Unicode, it does not require any specific font as compared to the Dot's Steganography and this gives an advantage to our algorithm. Through our algorithm, the stego-text document will be robust against any modification such as changing the font style, and therefore it will not lose the hidden secret information if there is a change in the font style.

5. CONCLUSION AND FUTURE WORKS

The Primitive Structural algorithm covers almost all the main structures of the Arabic character itself, such as sharp-edges, dots and typo-proportion. Each bit of secret information will be placed at corresponding place in the cover text. Besides high capacity of hiding secret information, our stego-text has high perceptual transparency, which means that it can be publicly published without raising any suspicions. This will be the main contribution of our research in Arabic text steganography.

Any changes will not be detected by the naked eyes because we are not dealing with the images but the Unicode of the Arabic characters. The limitation of this method are in the security part. As steganography has three important parameters, namely perceptual transparency, robustness, and hiding capacity, our future work will be focusing on randomness experiments for security.

ACKNOWLEDGEMENT

This research was supported by the Research University Grant Scheme (RUGS), Universiti Putra Malaysia —RUGS 05-02-12-1830RU (9340000).

REFERENCES

[1] K. Bennett, “Linguistic Steganography: Survey, Analysis, and Robustness Concerns for Hiding Information in Text,” West Lafayette, 2004.
[2] M. T. Chapman, “Hiding The Hidden : A Software System for Concealing Ciphertext as Inoncuous Text,” University of Wisconsin-Milwaukee, 1997.
[3] R. Bergmair, “A Comprehensive Bibliography of Linguistic Steganography,” 2006.
[4] A. Odeh, A. Alzubi, Qassim Bani Hani, and K. Elleithy, “Steganography by Multipoint Arabic Letters,” in *Systems, Applications and Technology Conference (LISAT), 2012 IEEE Long Island*, 2012.
[5] A. Odeh, K. Elleithy, and M. Faezipour, “Text Steganography Using Language Remarks,” *ASEE Northeast Sect. Conf.*, 2013.
[6] A. F. Al-Azawi and M. A. Fadhil, “Arabic Text Steganography using Kashida

Extensions with Huffman Code.," *J. Appl. Sci.*, 2010.

[7] A. A. Gutub and M. M. Fattani, "A Novel Arabic Text Steganography Method Using Letter Points and Extensions," *World Acad. Sci. Eng. Technol.*, pp. 28–31, 2007.

[8] A. Odeh and K. Elleithy, "Steganography in Arabic Text Using Zero Width and Kashidha Letters," *Int. J. Comput. Sci.*

[9] N. A. Roslan, R. Mahmod, and N. U. R. I. Udzir, "Sharp-Edges Method in Arabic Text Steganography," *J. Theor. Appl. Inf. Technol.*, vol. 33, no. 1, 2011.

[10] M. H. Shirali-Shahreza and M. Shirali-Shahreza, "Steganography in Persian and Aabic Unicode using Pseudo-Space and Pseudo Connection Characters," *J. Theor. Appl. Inf. Technol.*, pp. 682–687, 2008.

[11] M. H. Shirali-Shahreza and M. Shirali-Shahreza, "A New Approach to Persian/Arabic Text Steganography," in *5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering,Software Architecture and Reuse (ICIS-COMSAR'06)*, 2006, pp. 310–315.

[12] M. H. Shirali-Shahreza and M. Shirali-Shahreza, "Arabic/Persian text steganography utilizing similar letters with different codes," *Arab. J. Sci. Eng.*, vol. 35, no. 1, pp. 213–222, 2010.

[13] J. A. Memon, K. Khowaja, and H. Kazi, "Evaluation of steganography for Urdu/Arabic text," *J. Theor. Appl. Inf. Technol.*, 2008.

[14] T. Amano, "A Feature Calibration Method for Watermarking of Document Images," in *Document Analysis and Recognition, Proceedings of the Fifth International Conference*, 1999.

[15] Y.-W. Kim and I.-S. Oh, "Watermarking Text Document images Using Edge Direction Histograms," *Pattern Recognit. Lett.*, vol. 25, no. 11, pp. 1243–1251, Aug. 2004.

[16] Xinmei Sun, Peng Meng, Yun Ye, and Liusheng Hang, "Steganography in Chinese Text," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, 2010, vol. 8, pp. V8–651–V8–654.

[17] S. Changder, N. Debnath, and D. Ghosh, "Hindi Text Steganography–An Approach to Information Hiding," *J. Comput. Methods*, 2010.

[18] N. Samphaiboon and M. N. Dailey, "Steganography in Thai Text," *Int. Conf.*, 2008.

[19] A. Al-Nazer and A. Gutub, "Exploit Kashida Adding to Arabic e-Text for High Capacity Steganography," *Third Int. Conf. Netw. Syst. Secur.*, pp. 447–451, 2009.

[20] A. A. Gutub, L. M. Ghouti, Y. S. Elarian, S. M. Awaideh, and A. K. Alvi, "Utilizing Diacritic Marks for Arabic Text Steganography," vol. 37, no. 1.

[21] M. Shirali-Shahreza, "A New Persian / Arabic Text Steganography Using ' La ' Word," *Adv. Comput. Inf. Sci. Eng.*, p. 2008, 2008.

[22] M. Shirali-shahreza, "Pseudo-space Persian/Arabic Text Steganography," in *2008 IEEE Symposium on Computers and Communications*, 2008, vol. 3, pp. 864–868.

[23] A. A. Gutub and A. A. Al-nazer, "High Capacity Steganography Tool for Arabic Text Using 'Kashida,'" *ISC Int'l J. Inf. Secur.*, 2010.

[24] M. H. Awang, *Panduan Menulis dan Kaedah Mengajar Kaligrafi Nasakh*. Dewan Bahasa dan Pustaka, 2009.

[25] F. Al-Haidari, A. Gutub, K. Al-Kahsah, and J. Hamodi, "Improving Security and Capacity for Arabic Text Steganography Using 'Kashida' Extensions," *2009 IEEE/ACS Int. Conf. Comput. Syst. Appl.*, pp. 396–399, 2009.

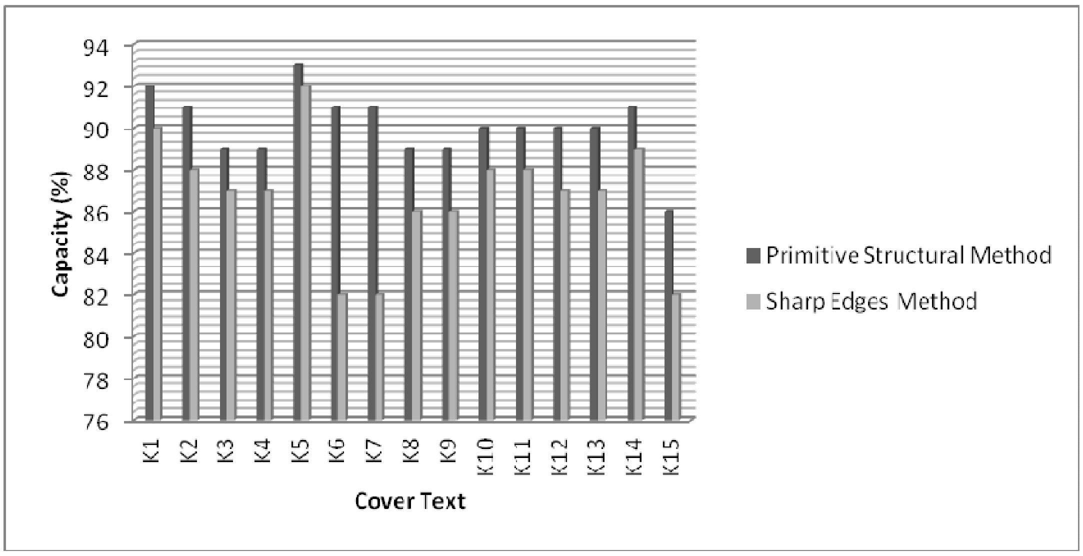

Figure 14: Capacity Measurement Percentage

Table 5: Comparison Results Between Primitive Structural Method With The Previous Methods

Online-Iranian Newspaper	Text Capacity (Bit)	Our Capacity Measurement (Bit/Kilobyte)	Sharp Edges Steganography Method (Bit/Kilobyte)	Pseudo Steganography Method (Bit/Kilobyte)	Dot Steganography Method (Bit/Kilobyte)	La Steganography Method (Bit/Kilobyte)
Hamshahri	6.82	1001	610	406	120	1.03
JameJam	3.84	729	608	405	113	1.48
Javan	8.03	758	707	402	115	1.00
Keyhan	2.92	861	725	404	106	2.05

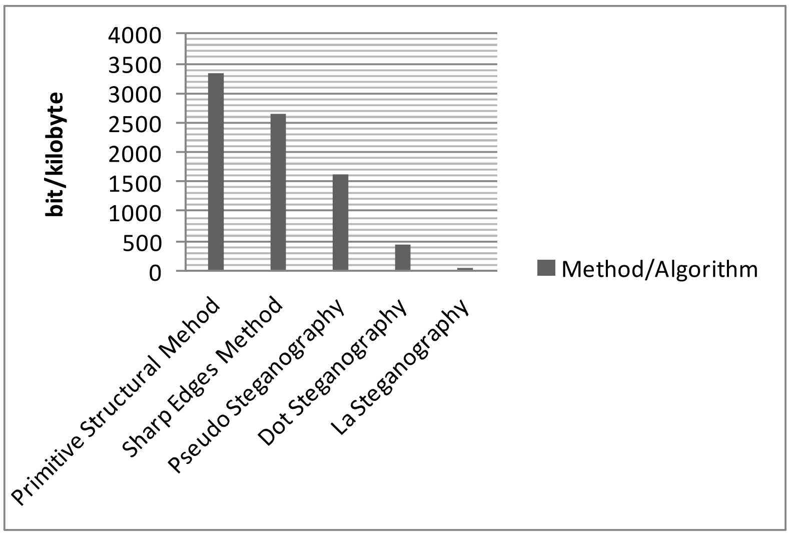

Figure 17: Comparison Graph Between Primitive Structural Method And The Previous Methods.