Arabic Text Steganography in Smartphone

Samira Mersal
Faculty of Science, Suez Canal University,
Ismailia, Egypt
Email:

Safiah Alhazmi, Razan Alamoudi, Noura Almuzaini
College of Computer Science and Engineering, Taibah University,
Almadinah Almunawarra, KSA

***Abstract*— Our work focuses on building an algorithm that helps in transferring information securely and keeping them save from any eavesdropping or malicious by using the concept of steganography. Steganography is one of the approaches used to protect information. It is the science of forming hidden messages such that the intended recipient is the only party aware of the existence of the message. Steganography algorithms use various cover media such as text, images, sounds and video.**

This paper presents a new Arabic text steganography algorithm based on sharp-edges concept. We select Arabic text as a cover text because it is our mother language and it is used in most of our communications. Java language is used to implement the algorithm for Smart phones. The result of the algorithm was compared to other methods in terms of capacity and showed it has highest capacity.

Keywords— ***Steganography, Arabic text steganography, Sharp edges, Data hiding, Smartphone application***.

I. INTRODUCTION

teganography is a Greek word coming from cover text S"Stegano" means hidden and "Graptos" means writing. In steganography, the secure data will be embedded into another object, so middle attacker cannot catch it. Most of steganography techniques use cover media such as pictures, video clips, sounds, and text, as shown in Figure 1. However, text steganography is not preferred because it is difficult to find redundant bits in text files. Choosing carrier file is very sensitive where it plays a key role to protect the embedded message [1].

There are three aspects in information hiding systems contend with each other: capacity, security, and robustness. Capacity refers to the amount of information that is able to be hidden in the medium, whereas security is important when a secret communication is kept to be secret and undetectable by eavesdroppers. Lastly, robustness can be explained as the amount of modification the stego-medium can withstand before an adversary can destroy hidden information[2].

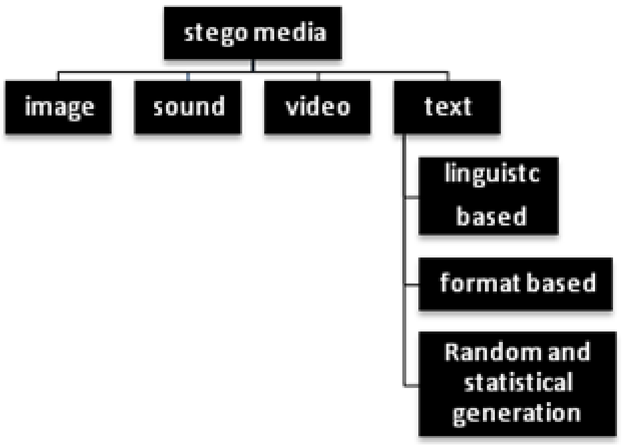

Figure 1: Stego Media

The following section II presents the characteristics of Arabic text. Section III presents related work. Our proposed work will be presented in section IV. Section V presents experimental results. Conclusion and future work presented in section VI followed by references in section VII.

II. CHARACTERISTICS OF ARABIC TEXT

- Arabic text alphabet does not differentiate between the upper and lower case or between written or printed letters.
- Arabic and English languages have dots in the character set. Arabic has dots in 15 letters out of 28 characters, while the Latin character set has dot in only two letters (i and j)[3]. Table 1, classifies the dots (pointed) and un-dots (un-pointed) character set of Arabic letters.

Table 1: Arabic letters

Un-pointed letters	**Pointed letters**
ا ح د ر س ص ط ع ك ل م هـ و	ب ت ث ج خ ذ ز ش ض ظ غ ف ق ن ي

- Most of Arabic letters can be connected together like (يعلمون) where in English all words consist of separated letters[1]. Connectivity is a result of the cursive nature of Arabic script. However, 8 out of the 28 Arabic letters do not connect to subsequent letters. Besides, even connectable letters do not connect to subsequent letters when the end of the word has been reached.

- Arabic language uses different marks. The main reason to use these symbols is to distinguish between words that have the same letters but with different meaning. Meanings of the words are different depending on Arabic Diacritics (Harakat)[1]. Diacritics are optional, not very common, practice in modern standard Arabic, except for holy scripts. Table 2 shows some letters with mark and their pronunciation.*Abbreviations and Acronyms.*

Table 2: Some Letters with Mark and their Pronunciation.

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

III. RELATED WORK

There are many algorithms used for Arabic text steganography. In this section we will present the most popular of them.

The dot method proposed by Shirali Shahreza[4] hides the secret information in the points (dots) location within the pointed (dots) letters. The cover text is scanned, character by character. Whenever a pointed (dots) 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.

Kashidah method proposed by Gutub and Fattani[5] 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 will hold the secret bit “1” and the un-pointed (without dots) letters with a Kashidah to hold “0”.

The pseudo-space and pseudo-connection character method proposed by M. H. Shirali-Shahreza and M. Shirali-Shahreza[6] hides one bit in each letter. First, we look whether the letter of a word is connected to the next letter or not. If it is, they insert a ZWJ (Zero Width Joiner) letter between the two letters to hide bit 1 and do not add anything for hiding bit 0.

Arabic Diacritics method proposed by Aabed[7] uses Fatha ◌َ"" to encode 1 otherwise encode 0. When OCR (Optical Character Recognition) detects the same message with different diacritics, it might conclude that there is a hidden data. In addition, retyping will remove the embedded message.

Other algorithms of Arabic text steganography exist in [8][9][10][11][12][13][14][15].

IV. PROPOSED ALGORITHM

A new algorithm based on the sharp-edges concept is proposed and implemented for smart phones. Sharp edges concept defined by Nuur Roslan et al in [3]. In the following sub-sections we will explain our algorithm, its architecture, components, and data used in it. Also, explaining the main processes of the algorithm, hiding process and retrieving process.

A.1 Algorithm Architecture

The algorithm will identify sharp edges, then through the hiding process, the secret message will be positioned randomly based on the capacity of the sharp edges in the cover text. Reverse will then be applied to retrieve back the secret message. Our steganography main architecture consists of two main processes, hiding process and retrieving process. Figure 2 and Figure 3 illustrates the main Processes of the algorithm.

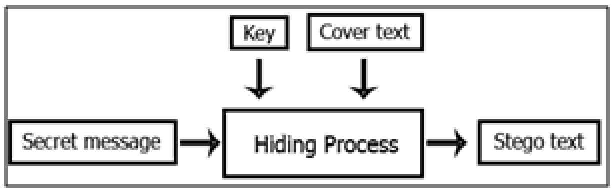

Figure 2: Hiding Process

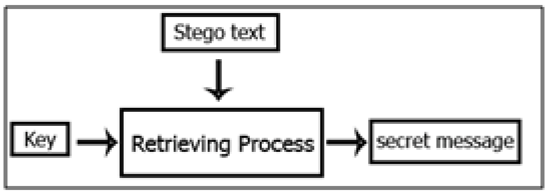

Figure 3: Retrieving Process

The hiding process is used by the sender of the hidden secret message. Meanwhile, the retrieving process is used by the recipient to retrieve the secret message from the stego text. The hiding module requires a key to randomly position the secret bit. This process will results a stego-text which can be publicly distributed without any suspicions.

The retrieving process uses the same key as the sender which is retrieved from the stego-text or entered by the user. Once the key is recognized by the algorithm, it will retrieve the secret message. The extracted secret message is the final output from this algorithm. Through this algorithm, we will analyze the result on the capacity of hiding places compared to Nuur's algorithm. In the next two sup-sections we will give an explanation for both processes, which represent the main functions of the algorithm.

A.2 Hiding Process

The hiding process is used by the sender to hide the secret message into the cover text. This process involves select the input file, which represents the secret message, and a group of sub processes which are Key Type Identification, Identification of cover text used in hiding message, Capacity Calculations, and Bit Hiding. The output of this process will be a stego text which will be used by the recipient to retrieve the secret message.

A.2.1 Key Type Identification

In this algorithm, the usage of a key is important to randomly position the secret bit in the cover text and to retrieve back the secret message. This process generates random key or receives it from the user. The output from this process will be used in the next process to generate a sequence of random numbers which represent the places in which the secret bits will be hidden.

A.2.2 Identification of Cover Text

It identifies the cover text that will be used in the hiding process. The user has the option to select file stored in his device or to enter the cover text. The number of the letters in the cover text is identified to generate sequence of random numbers from the stego key. This process is important to ensure that the positions of the secret bits are placed randomly within the range of the cover text. In our algorithm we assume that each letter in the cover text will be used only one time in the hiding process. So the method used for generating the numbers must ensure that the generated numbers are not repeated in the sequence.

A.2.2.1 Capacity Calculations

The capacity means that the maximum number of bits that can be hidden in the specified cover text. It will calculate the number of sharp-edges in the cover text letters. The total value of the sharp-edges will determine the capacity for hiding the secret bits. The calculation is based on the number of sharp-edges as listed in Table 3.

Table 3: Number of Sharp-Edges for each Letter

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 ؤ	ك أ إ ئ

In Addition, this process will convert the secret message to binary presentation, and then calculate the length of binary string. Each one of the bits is positioned in one sharp-edge in the cover text. So we can determine whether the cover text is suitable to hide the secret message or not.

A.2.2.2 Bit Hiding Process

This is the main process of the algorithm. It will process the binary string based on the number of sharp-edges in the cover text.

First, the process will hide the 24-bit of the key in the first 24 sharp-edges of the letters in the cover text. Then, it starts hiding the secret message by checking the number of sharp-edges of the letter which its position in the cover text is determined by the first number of the sequence of random numbers. Next, it will take the binary number of the secret message which is based on the sharp-edges of letter determined before. Then, take the decimal value of this binary number to generate code sequence of the secret message.

There is a relationship between the number of sharp-edges and the probability of the binary bit in each letter. Each code number used in the code sequence is a decimal number corresponding to binary secret bits which hidden in a specific character of the cover text. If the number of sharp edges in the letter is less than 4 sharp edges, the binary bit will be represented in the code sequence as one decimal digit. However, if the number of sharp-edges in the letter equal 4 or 5 sharp-edges it will be represented as 2 decimal digits in the code sequence. The resulting code sequence is used in the retrieving process with the stego-text. Figure 4, shows the pseudo code of the hiding process.

A.3 Retrieving Process

This process is used by the recipient to retrieve the secret message from the stego-text. It is the reverse of bit hiding process, and it needs the stego-text, the stego-key, and the code sequence.

First of all the receiver will extract the key from the first 24 bits of the sharp-edges in the stego-text. Next, the algorithm will generate a sequence of random numbers from this key to recognize the letters used to hide the secret message. Once the algorithm has recognized them, it rematches the code number with number of sharp-edges in each letter to return a string binary which represents the secret bits. Then it will convert them to string and the result will be the original secret message which is the final output from this process. Figure 5, shows the pseudo code of the retrieving process.

```
Hiding process:
    1. Convert the secret message into binary sequence.
    2. Calculate the number of sharp-edges (E) in the
       cover text.
    3. Suppose (B) is the number of bits in the binary
       sequence.
    4. If (E<B) then display "unsuitable cover text error
       ".
    5. Hide the key in the first 24 sharp-edges in the
       cover text.
    6. Calculate the number of letters in the cover text
       (N).
    7. Use the key to generate a sequence of random
       numbers in the range (1, N).
    8. While (B>0):
            a. Get the next number (S) in the
               sequence of random number.
            b. Get the letter number (S) of the cover
               text.
            c. Hide a number of (e) bits of the secret
               message equal to the number of sharp-
               edges in it.
            d. Subtract (e) from (B).
    9. End of the process.
```

Figure 4: The pseudo code of the hiding process.

```
    1. Extract the key from the first 24 sharp-edges in the cover
       text.
    2. Use the key to generate a sequence of random numbers
       in the range (1, N).
    3. Calculate the number of code sequence (C).
    4. While (C>0):
            e. Get the next number (S) in the sequence of
               random number.
            f. Get the letter number (S) of the cover text.
            g. Calculate the number of sharp-edges (e) in
               the letter.
            h. IF (e)>3 THEN take 2 digits (D) of the code
               sequence.
               Else take 1 digit (D) of the code sequence.
            i. Convert (D) to binary.
            j. Subtract (D) from (C).
    5. . End of the process.
```

V. RESULTS AND DISCUSSION

We experiment our algorithm for steganography on Arabic letters using the same example used by Nuur[3] on the following cover text:
عن أبي عبد الرحمن عبد الله بن عمر بن الخطاب رضي الله عنهما قال : سمعت رسول الله صلى الله عليه وسلم يقول : ((بني الإسلام على خمس : شهادة أن لا إله إلا الله و أن محمداً رسول الله، وإقام الصلاة، و إيتاء الزكاة، و حج البيت، وصوم رمضان)). رواه البخاري ومسلم.
In this example we find that the number of letters in the cover text is 187 letters. The Number of sharp-edges in the cover text is 391 edges. Also, the secret message we used in this example is: "Meet me tonight". The algorithm will convert the secret message to binary string, which will produce 104 bits.

First, the algorithm will generate 3-digit random key or receive it from the user. For example, the key generated by using the random function is 896, and we convert it to binary. Then, it is used to generate a sequence of random numbers, which are used to specify the letters on which we will hide on, to insure that the position of the secret bit is randomly placed.

A. Our method

We will work on dotted and un-dotted letters. We will use a generated random numbers to allocate letters which are enough to hide 104 bits of the secret message, this result the following characters.
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

Based on the number of sharp-edges of the first letter we will hide the number of bits from the secret message equal to this number, and a number is added to the code sequence and so on until we hide all the bits of the secret message, for example:

The letter (ر) contains 2 sharp-edges, so it will take the first 2 binary bits which is (01) and represent them in decimal which is equal to 1. The resulting code sequence will be used with the stego-text in the retrieving process.

B. Nuur's Method

In this method a key, which is received from the user, used to generate a random number. Then, calculations will total up the random numbers and determine it whether the result is an odd or even number.

1. Odd Method

It works on un-dotted letters, if the result number is odd. The un-dotted letters that are enough for hiding the secret message are
ع أ ع د ا ل ر ح م ع د ا ل ل ه ع م ط ا ر ا ل ل ه ع ه م ا ا ل س م ع ر ا ل ل ه
ص ل ى ا ل ل ه ع

The algorithm calculates the number of sharp-edges in these letters that we used to hide secret bit which is: 104 edges.

2. Even method

It works on dotted letters, if the result number is even. However this method in this example could not hide all the secret bits. It hides only 82 bits out of 104 bits without hiding the key bits. Secret message:" Meet me tonight" = 104 bits.

Table 4: Comparison between Methods

Method	**Cover text's sharp-edges**	**Sharp-edges used (for hiding secret bits)**	**Remaining sharp-edges**
Our method (dot and un-dotted letters)	391	104	287
Odd method (un-dotted letters)	305	104	201
Even method (dot letters)	86	86	0 and it did not fit all the secret bit

As seen in Table 4, the capacity of our algorithm compared to the methods of the Nuur's algorithm is greater. Based on the statistics in this table, we still can hide 35 more letters in the cover text using our method, 25 more letters using odd method and we could not hide the whole secret message in the cover text using even method. There is still 18 bits of the secret message which represent 3 letters of the secret message that could not be hidden.

VI. CONCLUSION AND FUTURE WORK

In this paper we have presented a new algorithm of Arabic text steganography using sharp-edges for hiding information. Result of proposed algorithm is compared with the result of Nuur Roslan’s algorithm; we found that our algorithm has more capacity. We will try to add new features to our algorithm that makes it more strong regarding the security and capacity.

REFERENCES

[1] Ammar Odeh and Khaled Elleithy: "Steganography in Arabic Text using Zero Width and Kashidha Letters", International Journal of Computer Science & Information Technology (IJCSIT), Vol. 4, No. 3, June 2012.

[2] M.Grace Vennice, Prof.Tv.Rao, M.Swapna, Prof.J.Sasi kiran, "Hiding the Text Information Using Steganography", International Journal of Engineering Research and Applications (IJERA), Vol. 2, Issue 1, Jan-Feb 2012

[3] Nuur Alifah Roslan, Ramlan Mahmod, and Nur Izura Udzir, "Sharp-Edges Steganography in Arabic Characters for Information Hiding", LAP LAMBERT Academic Publishing, pp. 116, 11th May 2012.

[4] M. Shirali Shahreza,"A New Approach to Persian/Arabic Text Steganography", Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2006),Honolulu, HI, USA, pp. 310-315, 12th July 2006.

[5] A. Gutub and M. Fattani, "A Novel Arabic Text Steganography Method using Letter Points and Extensions", Proceedings of the WASET International Conference on Computer,Information and Systems Science and Engineering (ICCISSE), Vienna, Austria, Vol.21, pp. 28-31, 2007.

[6] M.H. Shirali Shahreza and M. Shirali Shahreza, "Steganography in Persian and Arabic Unicode Texts using Pseudo-Space and Pseudo-Connection Characters", Journal of Theoretical and Applied Information Technology (JATIT), Vol. 4, No. 8, pp. 682-687, August 2008.

[7] M. Aabed, S. Awaideh, A. Elshafei, and A. Gutub, "Arabic diacritics Based Steganography", Proceedings of IEEE International Conference on Signal Processing and Communications (ICSPC 2007), 2007.

[8] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for Data Hiding", IBM Systems Journal, vol. 35, Issues 3&4, pp. 313-336, 1996.

[9] K. Rabah, "Steganography- The Art of Hiding Data", Information Technology Journal, vol. 3, Issue 3, pp. 245-269, 2004.

[10] F. Deguillaume, S. Voloshynovskiy, and T. Pun, "Character and Vector Graphics Watermark for Structured Electronic Documents Security", US Patent Application 10/949, 27th September 2004.

[11] M. Hassan Shirali Shahreza, Mohammad Shirali Shahreza, "A New Approach to Persian/Arabic Text Steganography", 5th IEEE/ACIS International Conference on Computer and Information Science (ICISCOMSAR 06), pp. 310- 315, July 2006.

[12] M. Hassan Shirali Shahreza, Mohammad Shirali Shahreza:"Arabic/Persian Text Steganography Utilizing Similar Letters with Different Codes",The Arabian Journal for Science and Engineering, Vol. 35, Number 1B, April 2010.

[13] Yousef Salem Elarian, Aleem Khalid Alvi, and Adnan A. Gutub, "Using Multiple Diacritics in Arabic Script for Steganography", King Fahd University of Petroleum & Minerals, 2008.

[14] M. Shirali Shahreza, "A New Persian/Arabic Text Steganography Using “La” Word", Proceedings of the International Joint Conference on Computer, Information, and Systems Sciences, and Engineering (CISSE2007), Bridgeport, CT, USA, pp. 339-342, 2007.

[15] Mujtaba S. Memon and Asadullah Shah, "A Novel Text Steganography Technique to Arabic Language using Reverse Fatha (الفتحة)", Pak. j. eng. technol. sci., Vol. 1, No. 2, pp. 106-113, 2011.