Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 10 (5): 436-439, 2010
ISSN 1812-5654

Arabic Text Steganography using Kashida Extensions with Huffman Code

A.F. Al-Azawi and Moayad A. Fadhil
Department of Software Engineering, Information Technology Faculty,
Philadelphia University, Amman, Jordan

Abstract: In this study, a text steganography technique suitable for Arabic texts is proposed, together with its implementation. This technique hides information by inserting extension characters (Kashida) at suitable word positions. We insert extension character in a word position to hold secret bit one and leaving position empty to hold secret bit zero. The Huffman compression algorithm is used to convert the embedding message into a compressed binary form and an Arabic text steganography technique based on character extensions is used to insert the compressed binary into the determined positions of the words in the cover text. This method was compared with some existing Arabic text stganography methods and it was shown more capacity and security.

Key words: Arabic text, cryptography, hiding information, text steganography, text watermarking, feature coding

INTRODUCTION

Images are the most popular carrier file for steganography because of the abundance of images available on the Internet. Another reason is the fact that the way images are stored creates a great amount of redundant space which is the ideal place to hide information. However, text steganography is not normally preferred due to the difficulty in finding redundant bits in text files (Morkel *et al*., 2005; Bennett, 2004). The structure of text documents is normally very similar to what is seen, while in all other cover media types, the structure is different than what we observe, making the hiding of information in other than texts easy without a notable alteration. The advantage to prefer text steganography over other media is its smaller memory occupation and simpler communication (Shirali-Shahreza and Shirali- Shahreza, 2006).

Text is one of the oldest media used in steganography; before the electronic age, letters, books and telegrams hide secret messages within their texts. Text steganography refers to the hiding of information within text (i.e., character-based) messages (Bennett, 2004) and may involve anything from changing the formatting of an existing text, to changing words within a text, to generating random character sequences or using context free grammars to generate readable texts (Bennett, 2004). Arabic text steganography methods will be explored in the next section. Compared to other media, text presents challenge for information hiding. This challenge requires the design of a robust algorithm that can work under constraint of low embedding bandwidth and our goal is to build Arabic text steganography system that is able to embed messages in Text format: *.doc, *.rtf, *.pdf, email, etc.

ARABIC TEXT STEAGNOGRAPHY METHODS

Some researches on hiding information in Arabic texts had been performed. Different methods, as examples, are presented in the following:

Dots steganography method: This method depends on the points in the Arabic letters. This large number of points in Arabic letters made the points in any given Arabic text can be utilized for steganography information security. The dots letters is used to hide bits. The dots slightly shifted up more than normal to represent the hidden bit 1 and kept the pointed character normal to hide 0. In this method, robustness is weak since it depends on using same fixed font, where using different font to produce unknown letters (Bennett, 2004).

Pointed letters and extensions method: This method considers the two features, the existence of the points in the letters and the redundant Arabic extension character (Kashida) to hide secret information bits. It uses the pointed letters with extension to hold the secret bit one and the un-pointed letters with extension to hold zero. This steganography method can have the option of adding extensions before or after the pointed letters. Figure 1 shows an example to hide the secret bits (110010) by adding extension before the pointed letter (Gutub and Fattani, 2007).

This steganography method does not suffer from increasing cover size due to hidden message embedding and also useful to other languages having similar texts to Arabic such as Persian and Urdu scripts (Gutub *et al*., 2007).

Corresponding Author: Ali Fouad Al Azawi, Department of Software Engineering, Information Technology Faculty, Philadelphia University, Amman, Jordan

Secret bits	110010
Cover-text	من حسن اسلام المرء تركه مالا يعنيه
Steganographic Text	من حسن اسلام المرء تركه مالا يعنيه ↑↑ ↑ ↑ ↑↑ 1 1 0 0 1 0

Fig. 1: Text Steganography example adding extensions before letters

Arabic diacritics method: This method utilizes the advantages of diacritics in Arabic to implement text steganography. Arabic text uses eight different diacritical symbols and this method uses the most frequent diacritical symbol Fatha. One of these methods, at start a fully diacritized Arabic text is used as cover media. To hide a bit 1, all diacritics are removed from the cover media until a Fatha is found and to hide a bit 0, the first non Fatha diacritic is kept. That means each Fatha represents 1 and other diacritic represents 0. The overall process is repeated for as long as there are bits remaining to be hidden (Aabed *et al*., 2007).

We need to note that diacritics approach, as well as the Kashidah approach, hiding a bit is equivalent to inserting a character (a diacritic mark or a Kashidah) (Gutub *et al*., 2008). The main advantages of this method are: provides the highest capacity, fast, does not require large computational power and can be implemented manually. While the main disadvantages are: suspicions raise since, it is uncommon nowadays to send diacritized text, the output text has a fixed frame due to the use of only one font and the information is lost in case of retyping (Aabed *et al*., 2007; Gutub *et al*., 2008).

Arabic unicode texts using pseudo-space and pseudo connection: In Arabic unicode texts, there are two characters, pseudo-space (ZWNJ-zero with non joiner) and pseudo-connection (ZWJ-zero with joiner) characters which are, respectively prevents Arabic letters from joining or forces them to join together. This method, first looks if the letter in a word connected to the next letter or not. To hide bit 1, ZWJ letter is inserted between letters if connected or ZWNJ letter is inserted between letters if not connected and do not add anything for hiding bit 0. This method is not dependent on any special format and can hide information in numerous formats such as HTML pages, Microsoft Word Documents and also capable to hide a bit of information in each letter (Shirali-Shahreza, and Shirali-Shahreza, 2008).

PROPOSED APPROACH

Benefiting from (Gutub and Fattani, 2007) Arabic Text Steganography method using letter points and extensions

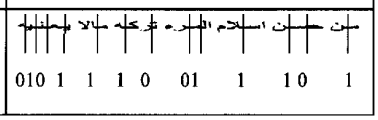
Secret bits	010111011101
Cover-text	من حسن اسلام المرء تركه مالا يعنيه
Steganographic Text	 010 1 1 1 0 01 1 10 1

Fig. 2: Text Steganography example

and trying to overcome the low capacity aspect, we propose a technique to hide information in a suitable position inside words instead of pointed letters only. These positions are determined to keep the Arabic text beauty if the text is justified and this allows messages to be hidden without affecting the cover text. We insert the extension letter in the determined position to hold secret bit one and leaving the position empty to hold secret bit zero. Figure 2 shows an example to hide the secret bits (010111011101) with 6 more bits than the example shown in Fig. 1 for pointed letters and extensions method.

The message were hidden or extracted by the use of the following described algorithms, implemented in J# language.

Compression: The compression Huffman encoding schemes is used and the frequencies of Arabic characters found in Al-Bukari HADITH book is used (Al-Muhtaseb *et al*., 2009) to generate encoding table and depending on the text message, we get approximately 26-44% compression rate. This was chosen because the extension character method provides very limited storage space and compression algorithm was needed.

Hiding secret message: At the first a Huffman tree is generated using the frequencies found (Al-Muhtaseb *et al*., 2009) of Arabic characters. Then the algorithm encodes the embedded message to binary form corresponds to Huffman tree. This reduces the total amount of bits required to be embedded. The approach finds suitable positions in the words, inserting extension letter in position for bit 1 and leaving position empty for bit 0. Algorithm 1 shows the detailed proposed algorithm for hiding message.

Algorithm 1: Hiding secret message

```
Input : cover text, message
Output : Stego text
Generate Huffman code table from Arabic characters frequencies
Codedmsg <- huf.encode(msg)
ExtLoc <- 0
while ( CodedMsg.hasMoreBits)
   Bit <- CodedMsg.NextBit
   if (CoverTxt.hasMoreExtension)
      ExtLoc <- CoverTxt.NextExtenstion
   else
      Exit-output Message can not be embedded
   end if
```

Algorithm 1: Continued

```
    if (bit=1) then

      Insert one extension at ExtLoc
    else
      no insertion needed and skip ExLoc
    end if
end while
if (CoverTxt.hasMoreExtension)
   CoverLoc <- CoverTxt.NextExtenstion
   Insert two extensions at ExtLoc to indicate end of message
else
    Exit-output Message can not be embedded
end if
```

Extracting secret message: To extract the message from the cover text, we respectively investigate the words of the text. If a position with extension character found, it means bit 1 is hidden. If a position without extension character found, it means bit 0 is hidden. By putting all bits next to each other, we extract the hidden message in binary form than Huffman code algorithm is used to decode the binary form to its text message. Algorithm 2 shows the detailed proposed algorithm for extracting message.

Algorithm 2: Extracting secret message

```
Input: Stego text
Output: Message
ExtLoc <- 0
CodedMsg<- null
while (CoverTxt.NextExtenstion is not endExtension)
   if (CoverTxt.hasMoreExtension)
      ExtLoc <- CoverTxt.NextExtenstion
    else
      Exit-output Message can not be extracted
    end if
    if (ExtLoc has one extension) then
      append one to CodedMsg
    else
      append zero to CodedMsg
    end if
end while
Generate Huffman code table from Arabic characters frequencies
Msg <- huf.decode(CodedMsg)
```

RESULTS AND DISCUSSION

We tested the implemented system with different cover text file sizes in terms of capacity. The capacity can be measured by the capacity ratio which is computed by dividing the amount of hidden bytes over the size of the cover text in bytes (Shirali-Shahreza and Shirali-Shahreza, 2006). Our approach is compared to dots approach, kashida approach and to diacritics approach. Table 1 shows the capacity ratio and the average capacity of these approaches.

Table 2 shows the capacity ratio and the average capacity of our proposed approach. The results show that the capacity ratio of our approach is more than dots and kashida approaches and less than diacritics approach as shown in the Table 1 and 2. But the capacity ratio of our approach will be increased to about 5.0 if the Huffman compression is used and also will be increased more if more one kashida is used. In addition, the cover text has a prefect perceptual transparency if it is justified by insetting these extension characters and the suspicions will be raised if diacritics are used, since it is uncommon nowadays to send diacritized text.

Table 1: Dots, kashida, and diacritics capacity ratios

Approach	Cover size (byte)	Capacity (%)	Average capacity (%)
Dots	13619.2	1.172	1.37
(Bennett, 2004)	6983.68	1.467	
	6799.36	1.275	
	3604.48	1.629	
Kashida	365181	1.215	1.22
(Gutub and	378589	1.172	
Fattani, 2007)	799577	1.266	
	15112	1.244	
Diacritics	318,632	3.250	3.27
(Aabed and	1,334,865	3.256	
Awaideh, 2007)	717,135	3.318	
	318,216	3.254	

Table 2: our proposed approach capacity ratios

Cover size (byte)	No. of hidden bits	Capacity ratio (%)	Average capacity (%)
4,050	969	2.991	3.019
15,563	3754	3.016	
24,957	6040	3.025	
29,283	7156	3.044	

CONCLUSIONS

This study presents a steganography technique useful for Arabic language electronic writing. It benefits from extension characters Kashida to hold secret information. The nice thing about Kashida is that it can be used to justify the text and does not have any affect to writing contents. We allocate positions suitable for inserting extension characters in the words of the cover text. These positions are used to hide secret bit one by inserting an extension character or to hide secret bit zero by leaving extension position empty. The work was evaluated with different Arabic text steganography approaches and it was shown that the presented technique provides a better capacity and more security compared to other Arabic text steganography methods that uses extension characters Kashida since, it compresses the embedded message using Huffman code and reduces the number of inserted extensions by leaving extension positions empty to hide zero bits. This Arabic text steganography techniques can be also applied to other languages such as Persian and Urdu and can be used to embed messages in electronic documents with different Text Format such as *.txt,*.doc, email, etc. These

characteristics and features promises that the Arabic text steganography method attractive for information security.

REFERENCES

Aabed, M.A., S.M. Awaideh, A.M. Elshafei and A.A. Gutub, 2007. Arabic diacritics based steganography. Proceedings of the International Conference on Signal Processing and Communications, Nov. 24-27, Dubai, UAE, pp: 756-759.

Al-Muhtaseb, H.A., S.A. Mahmoud and R.S. Qahwahi, 2009. A novel minimal script for Arabic text recognition databases and benchmarks. Int. J. Circ. Syst. Signal Process., 3: 145-153.

Bennett, K., 2004. Linguistic steganography: Survey, analysis and robustness concerns for hiding information in text. CERIAS Technical Report, Purdue University, West Lafayette, IN 47907-2086. https://www.cerias.purdue.edu/assets/pdf/bibtex_archive/2004-13.pdf.

Gutub, A.A.A. and M.M. Fattani, 2007. A novel Arabic text steganography method using letter points and extensions. World Acad. Sci. Eng. Technol., 27: 28-31.

Gutub, A.A.A., L. Ghouti, A.A. Amin, T.M. Alkharobi and M.K. Ibrahim, 2007. Utilizing extension character Kashida with pointed letters for Arabic text digital watermarking. Proceedings of the International Conference on Security and Cryptography-SECRYPT, July 28-31, Barcelona, Spain, pp: 25-32.

Gutub, A.A., Y.S. Elarian and A.K. Alvi, 2008. Arabic text staganography using multiple diacritics. Proceedings of the 5th IEEE International Workshop on Signal Processing and its Applications, March 18-20, University of Sharjah, Sharjah, United Arab Emirates, pp: 1-5.

Morkel, T., J.H.P. Eloff and M.S. Olivier, 2005. An overview of image steganography. Proceedings of the 5th Annual Information Security South Africa Conference, June 29-July 1, Sandton, South Africa, pp: 1-12.

Shirali-Shahreza, M.H. and M. Shirali-Shahreza, 2006. A new approach to persian/arabic text steganography. Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science, July 10-12, Honolulu, HI, USA., pp: 310-315.

Shirali-Shahreza, M.H. and M. Shirali-Shahreza, 2008. Steganography in Persian and Arabic unicode texts using pseudo-space and pseudo connection characters. J. Theor. Applied Inform. Technol., 4: 682-687.