Journal of Computer Science

Original Research Paper

Arabic Text Steganography Using Unicode of Non-Joined to Right Side Letters

Atef A. Obeidat

Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, Salt, Jordan

Article history
Received: 16-05-2017
Revised: 19-06-2017
Accepted: 23-06-2017

Email: atefob@gmail.com,
dr.atefob@bau.edu.jo

Abstract: Steganography is a technique for hiding data in media in a way that makes its existence hard to detect. Text files are a preferable format for use in steganography due to the small storage size of such files. This paper presents an Arabic text steganographic algorithm based on Unicode. The algorithm imposes a minimal change on connected letters without any change in size and shape. The experiment resulted in a high capacity rate ratio of about 180 bit/KB and low modification rate less than 90 letters/KB.

Keywords: Information Security, Arabic Text, Steganography, Text Hiding, Unicode

Introduction

Recent years have seen a growing number of cyber security attacks on businesses, healthcare organizations and others that use the Internet to exchange information. Therefore, protecting information that is transmitted via the Internet has become increasingly important. One effective method to protect such data involves hiding it using another medium such as text. This method is called steganography as shown in (Kessler and Hosmer, 2011).

The goal of hiding confidential information in another medium is to avoid drawing attention to the process of transference. A wide variety of media formats can be used for hiding data; these include image, sound, video and text files (Xu *et al*., 2006; Sridevi *et al*., 2009; Por *et al*., 2012; Bawaneh and Obeidat, 2016; Obeidat and Bawaneh, 2016). In this study, Arabic text is used to hide secret data because it offers a wide range of data hiding options. The type of media file used to hold data plays an important role in determining the method used to hide secret data. Text files can be a better choice than other media because of their small storage size compared to other media. Hidden data is embedded in a cover text (i.e., Arabic text) to create a stego text. Stego text consists of cover text, embedded data (i.e., secret data streams) and a key. The key consists of one bit that is inserted at the first location in the secret data. It should be noted that the method of hiding data used in text steganography depends on the language of the text, because of the differences in the code used to represent characters in each language. For example, in the English language, all text characters are written individually and with same shape, no matter where they appear in a word; in Arabic, there are two groups of letters: The isolated and a connected ones. Every character has different shapes according to their location in a word.

Visual appearance or perceptual transparency, robustness against modifications and cover media capacity are the main criteria's that should be taken in our consideration when building a steganography system (Anderson and Petitcolas, 1998; Cvejic and Seppanen, 2002). Perceptual transparency refers to the ability to know hidden information. It represents a data security factor. A high degree of security can be achieved by minimizing the impact of hiding in the cover text. This can be done by reducing the change between cover text and stego text by making the change to parts of the text that are hard to detect. Robustness refers to the ability to protect secret data from destruction during the process of transference. The hiding capacity refers to the ability of cover media to store data, which can be measured by the amount of secret data (bits) that can be hidden per kilobyte of cover media.

One of the difficulties in designing a hiding system in text is the lack of redundant data compared to files from other media. Another difficulty related to Arabic text is the ability of the user to discover any changes in the text where secret data are hidden. This is because Arabic text depends on the context, grammar and letter shapes in different situations.

In this study, a novel method to hide secret data in Arabic text, is proposed. The method consists of the encoding secret bit stream and extracting secret bit

stream algorithms which will be discussed in the following Sections.

In contrast to recent works on text steganography methods, the proposed method can be characterized by following features: (a) The method uses the Unicode of non-joined to right side letters as a cover letters. Thus, it is a novelty of the proposed technique. (b) It is considered the three criteria for building a steganography system. Although the capacity of the texts used to hide data is high, the proposed method has satisfied the robustness against modification and visual appearance of the stego text of the identical to the cover text without attracting attention.

This paper is structured as follows: Section 2 reviews the related works of Arabic text steganography based on Unicode; the proposed method is discussed in detail in section 3; section 4 presents the experiment results; and conclusions are presented in section 5.

Review of Related Literature

Many of the text steganography techniques described in the literature are specific to the English language. However, a few studies do explore the use of text steganography techniques in Arabic languages and identify limitations related to security and capacity (Shirali-Shahreza and Shirali-Shahreza, 2008a; Odeh *et al*., 2012; Al-Nofaie *et al*., 2016; Kadhem and Ali, 2017; Malik *et al*., 2017).

Arabic text steganography methods can be classified into the following types as summarized in (Mohamed, 2014): Shifting points-based (Al-Nazer and Gutub, 2009; Odeh *et al*. 2012), Arabic diacritics-based (Harakat) (Bensaad and Yagoubi, 2011), Kashida-based (Al-Haidari *et al*., 2009; Al-Azawi and Fadhil, 2010), Unicode-based (Shirali-Shahreza and Shirali-Shahreza, 2008a; 2008b; Por *et al*., 2012; Mohamed, 2014) and linguistic-based (Desoky, 2009; Alabish *et al*., 2013).

The method proposed here belongs to the Unicode-based type of steganography. This uses Unicode letters or symbols to hide secret packets of bits.

A recent study by Shirali-Shahreza and Shirali-Shahreza (2008a) in the field of Unicode-based Arabic text steganography proposed a method for hiding information in Persian and Arabic Unicode text. This method uses two characters, a Zero-Width Non Joiner (ZWNJ) and a Zero-Width Joiner (ZWJ), to hide information. It provides a high capacity because it hides one bit in each letter, although it requires the content of the carrier text to be changed frequently by the addition of ZWJs or ZWNJs to hide 1s.

Mohamed (2014) proposed a steganographic method for Arabic text that uses an isolated letter in a word to hide data. This technique provides a relatively secure algorithm and low capacity in comparison with the proposed method.

Shirali-Shahreza and Shirali-Shahreza (2008b) proposed a method to hide data by changing the standard Unicode with the code for the contextual form of the letter. But when there is combination of codes to represent letters and codes to represent letter shapes in a single word, the text viewer does not select the correct shape of the represented letters automatically and shows them in an isolated form. Solving this problem requires a method to insert a ZWJ character between the two letters to connect them. This solution requires changing a very large amount of content in the cover text.

Por *et al*. (2012) proposed a method to encode external information by inserting Unicode of special characters into inter-sentence, inter-word, end-of-line and inter-paragraph spaces. Consequently, hiding data requires changing the content of the cover text.

Kadhem and Ali (2017) proposed method takes all the cover texts (characters, diacritics, spaces between word and special character) and utilizes the spaces after the cover text, that will increase capacity ratio but requires changing a very large amount of content in the cover text.

Al-Nofaie *et al*. (2016) proposed method to hide secret bits inside the Kashida as well as spaces between words. In spite of the method increases the capacity of hiding it requires change in cover text in addition need to increase in size.

Malik *et al*. (2017) presented method in which the secret data bits are embedded in the cover text by making it colored using a color coding table. The problem with this method is to change the color of cover text which attracts attention.

A secure, high-capacity method is proposed that overcomes the above limitations, avoids changing the content of the cover text and does not corrupt the shape of words.

Arabic Language and Unicode System

Characteristics of the Arabic Language

The Arabic language is used by a large number of people. The Arabic alphabet has 29 letters (Fig. 1). However, many linguists believe that Hamza and Aleph are one letter, which means the total number of letters in the Arabic alphabet is actually 28 (Unicode.org, 2016).

In Arabic, letters in printed text are connected to one another, while in English, each letter is written separately. The Arabic language can be divided into two groups of letters (Unicode.org, 2016): The connected and the isolated ones as shown in Fig. 2. The connected group contains 22 letters, each of which can take four different forms depending upon its location in a word as shown in Fig. 3: Isolated, final, middle and initial of a word. Each letter is connected to the succeeding letter in the same word. For example, the connected letter (Beh) can be written by four forms as shown in Fig. 3.

أ	ب	ت	ث	ج	ح	خ
Alef	*Beh*	*Teh*	*Theh*	*Jeem*	*Hah*	*khah*
د	ذ	ر	ز	س	ش	ص
Dal	*Thal*	*Reh*	*Zain*	*Seen*	*Sheen*	*Sad*
ض	ط	ظ	ع	غ	ف	ق
Dad	*Tah*	*Zah*	*Ain*	*Ghain*	*Feh*	*Qaf*
ك	ل	م	ن	هـ	و	ي
Kaf	*Lam*	*Meem*	*Noon*	*Heh*	*Waw*	*Yeh*

Fig. 1. Arabic letters

أ و ز ر ذ د	Isolated letters
ب ت ث ج ح خ س ش ص ض ط ع غ ف ق ك ل م ن ه ي	Connected letters

Fig. 2. Types of Arabic letters

General Unicode	Contextual forms				Name
	Isolated	Final	Middle	Initial	
0627 ا	FE8D ا	FE8E ا			Alef
0628 ب	FE8F ب	FE90 ب	FE92 بـ	FE91 بـ	Beh
062A ت	FE95 ت	FE96 ت	FE98 تـ	FE97 تـ	The

Fig. 3. Unicode Arabic letters from Alef to The

As for the remaining six letters of the Arabic alphabet, the letters Thal, Dal, Reh, Zain, Waw and Alef have only two variants: As the first letter in a word and as the last letter in a word (Fig. 3). These letters cannot be joined to the succeeding letter, even when they are inside a word. Each letter that comes after these six letters is written as an initial form and it must be disconnected with the previous letter. In addition, it is noted that the first letter in a word has the same properties. Those letters make a set of non-joined to the right side letters. This set of letters are used as hidden keys in the proposed steganographic method. This will enable us to avoid the previously discussed drawbacks of Unicode-based algorithms.

Unicode in Arabic Language

Unicode is a universal standard for encoding text or script characters. The Arabic script is used to represent many languages, including Urdu (the official language of Pakistan), Persian (the official language of Iran) and Pashto (the official language of Afghanistan) (Unicode.org, 2016). The Unicode Standard includes many groups of Extended Arabic letters belonging to some languages that are not well documented. The Unicode Standard covers in detail the mechanisms for implementation, including methods for connecting letters, displaying text from right to left and displaying bidirectional text (Gillam, 2002).

In this standard, each Arabic letter has a general code and then has different codes for each of its different presentation forms in a word. For example, the general code of letter Beh is 0628. The codes of different shapes are FE8F for an isolated form, FE90 for the final one, FE91 for the one at the beginning and FE92 for the one at the middle, as shown in Fig. 3. In the Unicode Standard, the general-letter code is used only to save data in digital media. A shape-determination routine (IBM_web_site, 2017) is used to display Arabic text by selecting the correct shape for the letter based upon its location in a word.

The Proposed Method

In this section, a new method is proposed, in which the letters are not joining with previous character are used as hidden keys for Arabic text written in the Unicode format. Therefore, the proposed method contains an algorithm to look for these letters in the words of Arabic text. The method distinguishes the cover letter in a text when satisfies one from the following situations: (a) The letter is located after an isolated letter. (b) The letter is located at an initial position in a word.

The significance of the new method is that it hides data in the cover text by using these types of letters without any noticeable change in the target word; it does not change the original character of the symbols. It also offers a high capacity due to the availability of high ratio of characters to carry hidden data. There is a low probability of detecting hidden data because this method does not change the shape of the letters; instead, it exchanges the standard code with the contextual code for carrier letters only when hiding 1s.

The proposed method will use those cover letters of Arabic text written in Unicode format to hide secret data. The hiding program aims to look for cover letters in the words of Arabic text. Secret data is hidden in the cover text by using these types of letters without any change in the appearance of words. The output of the hiding process is called stego text which is the cover text with change for some cover letters. The number of cover letters will be changed to hide the secret data is always less than 50% of the total number of cover letters. So the rate of modification the cover text will be less than 1/2 number of cover letter.

Simplifying the complexity of the algorithm, it changes the code of cover letter to hide one otherwise Unicode of a letter stays as it. For example, using the proposed algorithm, the Arabic sentence in Fig. 4 can hide a packet of 17 bits. This is the sum of the total number of cover letters in the Arabic sentence.

Hiding a Secret Code in a Plain Text

Suppose the cover media is the Arabic text as in Fig. 5. While "101" is the secret stream to be hidden in it. Applying the encoding algorithm as shown in Fig. 6. requires the following steps to be performed to hide the secret code:

Step 1: Apply the 1's complement for the secret stream (i.e., invert all bits), because the number (count) of 1s is more than the number of 0s in secret stream to reduce the character change. Then insert a bit with value "1" at the first bit of the secret stream to indicate the exiting of 1's complement. The secret stream thus becomes "1010".
Step 2: Read the first bit of secret data. Then identify the first cover letter in the text. In this case, the next hidden bit is "1", while the cover letter is Qaf.
Step 3: Because the hidden bit is "1", the algorithm will convert the general Unicode of the letter Qaf into the initial form.
Step 4: Identify the next cover letter Lam, to hide the second bit (i.e., "0"). In this case, the general Unicode of the letter Lam is not changed.
Step 5: Read the next hidden bit (i.e., "1"). According to the algorithm, the cover letter (i.e., Alef) will be read from the carrier text. In this case, the general Unicode of the letter Alef must change to isolated form.
Step 6: Read the next hidden bit (i.e., "0") and the next connected letter (i.e., Lam). In this instance, the general Unicode of the letter Lam is not changed.

From the previous example, the hiding process can be summarized using the proposed algorithm as follows: (a) To reduce the character change, count the number of 0s; if there are fewer 0s than 1s, 1's complement is applied to the secret packet. (b) to hide a bit of "0", a cover letter must be leaved unchanged from the next word in the text. (c) to hide a bit of "1", a cover letter must be identified from the text. The Unicode of this letter must then be changed from the general letter Unicode to the appropriate contextual Unicode according to type of letter. If the cover letter belongs to isolated group, its Unicode is changed to the isolated form else, the cover letter is a connected one then its Unicode is changed to initial form.

How to Extract the Secret Code from the Stego Text

To extract the secret message from the stego text produced in the previous example as shown in Fig. 5. The extraction algorithm as shown in Fig. 7 is used, this is done as follows:

Step 1: Identify the first cover letter in the text. In this case, the next connected letter is *Qaf*. Because its Unicode is changed, it means the hidden bit is "1".
Step 2: Identify the next cover letter, which in this case is *Lam in first word*. Because its Unicode is not changed, this means the hidden bit is "0".
Step 3: Then, read the next cover letter (i.e., *Alef*) from the second word. Because its Unicode is changed, this means the hidden bit is "1".
Step 4: Finally, find the next connected letter, which in this case is *Lam*. Because its Unicode is not changed, this means the hidden bit is "0".

Arabic sentence	قال السماء كئيبة وتجهما قلت ابتسم يكف التجهم في السما									
Words	السما	في	التجهم	يكف	ابتسم	قلت	وتجهما	كئيبة	السماء	قال
Non-joined to right side letters	ا ل	ف	ا ل	ي	ا ب	ق	و ت	ك	ا ل ء	ق ل
Maximum number of hidden bits	2	1	2	1	2	1	2	1	3	2

Fig. 4. The maximum number of hidden bits in an Arabic sentence

Arabic sentence	قال السماء كئيبة وتجهما			
Words	وتجهما	كئيبة	السماء	قال
Cover letters	و ت	ك	ا ل ء	ق ل
Unicode Carrier letters	\u0648 \u062A	\u0643	\u0627 \u0644 \u0621	\u0642 \u0644
Unicode after hiding	Not used	Not used	**\uFE8D** \u0644 **(not used)**	**\uFED7**\u0644
Secret stream	Not used	Not used	**1 0**	**1 0**

Fig. 5. Example for the encoding and extraction processes

Algorithm 1: Encoding secret bits

Input: *Carrier text (C), secret bit stream (B)*
Output: *Stego text (S)*

```
1.  If (#1s in B > #0s), then //Calculate the number of ones in the secret bit stream (B)
2.     Calculate 1's complement on B. // To reduce the rate of change in the code
3.     Insert "1" at the first location in B.
4.  Else
5.     Insert "0" at the first location in B
6.  End if
7.  Initiate the pointer of the reading from the carrier text to the first cover letter L
8.  While (not EOF(B) ) do      // While not reaching the end of the secret bit stream
9.     Read next bit(b) from B   // Where b may be "1" or "0"
10.    If (b== 0) then
11.       Do nothing.
12.    Else
13.       Change Unicode of L   //Convert from General Unicode to contextual
14.    End if
15.    Read next secret bit into B
16.    Read next cover letter into L   // When the letter is  not joining to right side
17. Loop step 8
18. End algorithm
```

Fig. 6. Algorithm for encoding secret bits

Algorithm 2: Extraction of secret bits

Input: *Stego text (S)*
Output: *Secret bit stream (B)*

```
1.  Initiate the pointer of the reading from S to the first connected letter, L.
2.  While (not EOF(S))  do        // While not reaching at the end of  the file of the Stego text
3.     Read next cover letter L    //When the letter is  not joining to right side
4.     If (Unicode (L) is not universal) then   //Extracting the next secret bit "0" or "1"
5.           Insert "1" into B
6.     Else
7            Insert "0" into B
8      End if
9.  Loop step 2
10. If (B[0]="1")  then          // If first bit of secret bit stream is "1"
11.        Apply 1's complement on B   // To reach to original secret data
12. End if
13  End algorithm.
```

Fig. 7. Algorithm for extracting secret bits

In the extraction process, the algorithm identifies cover letters in the stego text. If Unicode of cover letter is changed, this indicates a hidden bit with value "1" else hidden bit is "0". After extraction, the first bit should be checked. If it is "1", the algorithm will perform 1's complement on the secret data.

Encoding Algorithm

This subsection presents the proposed Arabic steganographic algorithm (i.e., the encoding algorithm) as shown in Fig. 6. The input of the algorithm consists of the cover text and the *secret bit stream*. The first bit is considered as a key to the data. If its value is "1", this means the 1's complement on the secret bit stream has been performed. If the value is "0", the secret bit stream is unchanged. The output is stego text with minimum change. The algorithm is stopped when the secret data is hidden.

Extraction Algorithm

The extraction algorithm is the reverse of the encoding algorithm. It discovers the data hidden in the

Stego text as shown in Fig. 7. The input of the algorithm is the stego text where its output is the secret bit stream. When the algorithm reads a connected letter, it verifies its Unicode. In case of the Unicode is changed, the algorithm will consider the letter represents "1" otherwise "0". If value of the first bit of secret data is "1", this means the one's complement on the secret bit stream has been performed.

Results of the Experiment

The experimental results are explained in this section. The experiment considers the aforementioned criteria: Capacity, visual appearance and robustness. Using the proposed algorithm, information is hidden in Arabic text using the Unicode Standard.

The algorithm is tested on different types of Arabic files. The experiment is applied on the files for computing the capacity ratio of the algorithm for hiding data. The capacity ratio (Kessler and Hosmer, 2011) is calculated using Equation 1:

$$CR = \frac{NH}{SC} \quad (1)$$

Where:

CR = The capacity ratio (b/KB)
NH = The number if hidden bits (b)
SC = The Size of cover text (KB)

The results of the experiment are shown in Table 1. The algorithm capacity is about 180 bit/KB. Files that contain Harakat have a reduced capacity of approximately 130 bit/KB, while the rate of modification for cover text is less than 90 letters/KB.

Table 1. Experimental results for capacity of the proposed algorithm

File name	Size (KB)	Capacity (bit)	Capacity ratio	Notes
File 1	23	4319	187	
File 2	12	1810	150	
File 3	4	801	200	
File 4	35	5050	144	With Harakat
File 5	12	1516	126	With Harakat
File 6	13	2369	182	
File 7	109	13381	122	With Harakat
File 8	20	3954	198	
File 9	8	1382	172	

Arabic sentence(cover text)	قال: الصبا ولّى! فقلت له: ابتسمْ لن يرجعَ الأسفُ الصبا المتصرما
Secret stream	"11001010111011001110"
Unicode's cover text	\u0642\u0627\u0644\u003a \u0627\u0644\u0635\u0628\u0627 \u0648\u0644\u0651\u0649\u0021 \u0641\u0642\u0644\u062a \u0644\u0647\u003a \u0627\u0628\u062a\u0633\u0652 \u0645\u0652 \u064a\u0631\u062c\u0639\u064e \u0627\u0644\u0623\u0633\u064f \u0627\u0644\u0635\u0628\u0627 \u0627\u0644\u0645\u062a\u0635\u0631\u0645\u0627
1's complement of secret stream with key ("1").	"10011010100010011000 1"
Unicode's stego- text	(IBM_web_site 2017)(IBM_web_site 2017)(IBM_web_site 2017)(IBM_web_site 2017)(IBM_web_site 2017)(IBM_web_site 2017)\uFED7\u0627\u0644\u003a \u0627\uFEDF\u0635\u0628\u0627 \uFEED\u0644\u0651\u0649\u0021 \uFED3\u0642\u0644\u062a \u0644\u0647\u003a \uFE8D\u0628\u062a\u0640\u0640\u0633\u0645\u0652 \u0644\u0646 \u064a\u0631\uFE9F\u0639\u064e \u0627\u0644\u0623\uFEB3\u0641\u064f \uFE8D\u0644\u0635\u0628\u0627 \u0627\u0644\u0645\u062a\u0635\u0631\uFEE3\u0627
Stego-text	قال: الصبا ولّى! فقلت له: ابتسمْ لن يرجعَ الأسفُ الصبا المتصرما

Fig. 8. Results of the proposed algorithm to test the visual appearance experiment

The capacity of the proposed algorithm is high compared to similar method (Mohamed, 2014), in which isolated letters only is used to store secret data and these letters are appear a little in the Arabic text.

In addition, the proposed algorithm is tested on Arabic sentence to hide the secret stream "11001010111011001110" in it as shown in Fig. 8. The result of experiment showed how the proposed algorithm can hide the secret data without attracting attention in the text. This shows that the proposed algorithm adopted a visual appearance criterion.

The Robustness criterion was verified by re-sending the stego text (Fig. 8) via e-mail many times. In addition, we opened it in several editors and browsers for the text files and then the same secret data was retrieved from the stego text.

Conclusion

A new steganographic text method is presented here for Arabic text or for other languages written in Arabic letters, such as Urdu, Persian and Pashto (Unicode.org 2016). Many features for the proposed algorithm can be concluded from experiment results: (a) The proposed algorithm has a high capacity, with a ratio of about 180 bit/KB. (b) the proposed algorithm hides information without change in the carrier text, which means that the steganographic text generated by the proposed algorithm will not attract attention; consequently, the algorithm satisfies the requirements for security. (d) The technique for hiding data is robust because the steganographic text cannot be changed during the process of transference and secret data remains hidden within the text. (d) The algorithm can withstand traditional attacks because it hides secret data with minimal change on the carrier text. (e) The algorithm works without any change in the size of the text and uses only the Unicode for each letter; consequently, the algorithm can be used with any Arabic text without any specific format.

Acknowledgment

This research is supported by Dept. of Information Technology, Al-huson University College, Al- Balqa Applied University.

Ethics

This article is original and contains unpublished Material, the corresponding author confirms that no ethical issues involved.

References

Al-Azawi, A.F. and M.A. Fadhil, 2010. Arabic text steganography using kashida extensions with huffman code. J. Applied Sci., 10: 436-439. DOI: 10.3923/jas.2010.436.439

Al-Haidari, F., A. Gutub, K. Al-Kahsah and J. Hamodi, 2009. Improving security and capacity for Arabic text steganography using 'Kashida' extensions. Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications, May 10-13, IEEE Xplore Press, pp: 396-399. DOI: 10.1109/AICCSA.2009.5069355

Al-Nazer, A. and A. Gutub, 2009. Exploit kashida adding to Arabic e-text for high capacity steganography. Procedings of the 3rd International Conference on Network and System Security, Oct. 19-21, IEEE Xplore Press, pp: 447-451. DOI: 10.1109/NSS.2009.21

Al-Nofaie, S., M. Fattani and A.A.A. Gutub, 2016. Capacity Improved Arabic Text Steganography Technique Utilizing 'Kashida' with Whitespaces. Proceedings of the 3rd International Conference on Mathematical Sciences and Computer Engineering, Feb. 4-5, Langkawi, Malaysia, pp: 38-44.

Alabish, A., A. Goweder and A. Enakoa, 2013. A universal lexical steganography technique. Int. J. Comput. Commun. Eng., 2: 153-157. DOI: 10.7763/IJCCE.2013.V2.159

Anderson, R.J. and F.A. Petitcolas, 1998. On the limits of steganography. IEEE J. Selected Areas Commun., 16: 474-481. DOI: 10.1109/49.668971

Bawaneh, M.J. and A.A. Obeidat, 2016. A secure robust gray scale image steganography using image segmentation. J. Inform. Security, 7: 152-164. DOI: 10.4236/jis.2016.73011

Bensaad, M.L. and M.B. Yagoubi, 2011. High capacity diacritics-based method for information hiding in Arabic text. Proceedings of the International Conference on Innovations in Information Technology, Apr. 25-27, IEEE Xplore Press, pp: 433-436. DOI: 10.1109/INNOVATIONS.2011.5893864

Cvejic, N. and T. Seppanen, 2002. Increasing the capacity of LSB-based audio steganography. Proceedings of the IEEE Workshop on Multimedia Signal Processing, Dec. 9-11, IEEE Xplore Press, pp: 336-338. DOI: 10.1109/MMSP.2002.1203314

Desoky, A., 2009. Listega: List-based steganography methodology. Int. J. Inform. Security, 8: 247-261. DOI: 10.1007/s10207-009-0079-0

Gillam, R., 2002. Unicode Demystified: A Practical Programmer's Guide to the Encoding Standard. 1st Edn., Addison-Wesley Professional, Boston, ISBN-10: 0201700522, pp: 853.

IBM_web_site, 2017. Automatic shaping program.

Kadhem, S.M. and M. Ali, 2017. Proposed hybrid method to hide information in Arabic text. J. Theor. Applied Inform. Technol., 95: 1466-1478.

Kessler, G.C. and C. Hosmer, 2011. An overview of steganography. Adv. Comput., 83: 51-107. DOI: 10.1016/B978-0-12-385510-7.00002-3

Malik, A., G. Sikka and H.K. Verma, 2017. A high capacity text steganography scheme based on LZW compression and color coding. Eng. Sci. Technol. Int. J., 20: 72-79. DOI: 10.1016/j.jestch.2016.06.005

Mohamed, A., 2014. An improved algorithm for information hiding based on features of Arabic text: A Unicode approach. Egypt. Inform. J., 15: 79-87. DOI: 10.1016/j.eij.2014.04.002

Obeidat, A.A. and M.J. Bawaneh, 2016. A novel FLV steganography approach using secret message segmentation and packets reordering. Int. J. Res. Comput. Applic. Robot., 4: 44-54.

Odeh, A., A. Alzubi, Q.B. Hani and K. Elleithy, 2012. Steganography by multipoint Arabic letters. Proceedings of the IEEE Long Island Systems, Applications and Technology Conference, May 4-4, IEEE Xplore Press, pp: 1-7. DOI: 10.1109/LISAT.2012.6223209

Por, L.Y., K. Wong and K.O. Chee, 2012. UniSpaCh: A text-based data hiding method using Unicode space characters. J. Syst. Software, 85: 1075-1082. DOI: 10.1016/j.jss.2011.12.023

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

Shirali-Shahreza, M. and S. Shirali-Shahreza, 2008b. High capacity Persian/Arabic text steganography. J. Applied Sci., 8: 4173-4179. DOI: 10.3923/jas.2008.4173.4179

Sridevi, R., A. Damodaram and S.V.L. Narasimham, 2009. Efficient method of audio steganography by modified LSB algorithm and strong encryption key with enhanced security. J. Theor. Applied Inform. Technol., 5: 668-671.

Unicode.org, 2016. The Unicode standard. Unicode.org.

Xu, C., X. Ping and T. Zhang, 2006. Steganography in compressed video stream. Proceedings of the 1st International Conference on Innovative Computing, Information and Control, Aug. 30-Sept. 1, IEEE Xplore Press, pp: 269-272. DOI: 10.1109/ICICIC.2006.158