A MELIORATED KASHIDA-BASED APPROACH FOR ARABIC TEXT STEGANOGRAPHY

Ala'a M. Alhusban and Jehad Q. Odeh Alnihoud

Computer Science Dept, Al al-Bayt University, Mafraq, Jordan

ABSTRACT

Steganography is an art of hiding a secret message within some cover media such as: images, audios, videos, and texts. Many algorithms have been proposed for Arabic text steganography exploiting some features of Arabic language such as: diacritics, extension character (kashida), and pointed letters. In this research we propose a new method to enhance a kashida-based methods for text steganography. In which each existing kashida can hide two bits instead of only one bit. In addition, security measures is increased through embedding the secret bits into the cover text by two different ways since the cover text is divided into two blocks; each one of them is being treated in different way. Moreover, the original kashida in the cover text is ignored by the extractor. A system is designed to achieve the embedding as well as the extracting with high degree of security through authentication operation used in its interface. The proposed approach is tested and compared with the most related kashida-based approaches in terms of capacity and the results are promising. Furthermore, it overcomes the limitations of other approaches, maintain a reasonable increases in the files size, and enhances security measures.

KEYWORDS

Arabic text, Steganography, Kashida, Pointed characters, Zero-width character.

1. INTRODUCTION

In general text is considered as one of the hardest cover media type, due to the limited options it has, these limitations related to the nature of the direct appearance of the letters in their original representations [14].

Many techniques have been introduced to hide a secret message in English language. However, fewer methods have been proposed for Steganography in Arabic language. These methods have mainly focused on some features of Arabic language such as diacritics, extensions (kashidas), pointed and un-pointed letters and other techniques such as line shifting as well as spaces between letters [5].

Kashida-based methods are concerned with the extension letter named kashida, which can be added to the majority of Arabic letters, depending on their positions in the word. Such extension cannot be added before the first letter of the word or after the last letter of the word, but it can rather be added between two letters. Moreover, there are certain characters that do not accept adding kashida after them: (ا،ء،أ،إ،آ،ؤ،ى،ئ،ر،ز،و), even if they come in the middle of the word [4].

There are two main types of Steganography; linguistic steganography and technical steganography. Linguistic steganography refers to any form of steganography that uses language in the cover. There are several examples of linguistic steganography. One of these is called open codes, in which the readable text can contain certain secret text which can be in certain places in the text, or can be hidden vertically or in reversed position. Technical steganography is the

DOI:10.5121/ijcsit.2017.9209

method of steganography which uses a tool, a device or a method to conceal the message. Unlike Linguistic steganography, dealing with the written words is not a must in Technical Steganography [1]. Steganography process consists of two phases: embedding (hiding the secret message) and extracting (retrieving the secret message) [11].

2. LITERATURE REVIEW

In [2], Gutub and Fattani proposed adding an extension to a pointed letter to represent a secret bit of value (1) and adding an extension to a non-pointed letter to represent a secret bit of value (0). By applying this method, you may add kashidas before or after the letters. Both options of adding extensions before and after letters can be used within the same document but in different paragraphs or lines. This method has some problems such as the large number of kashidas required to represent the secret message, since a kashida represents only one bit, so that the size of the document will increase significantly. Furthermore, extensive use of kashidas affects security measure.

In [3], Shahreza and M. Shahreza used a method that is structured as follows; If the representative forms of "Lam" (its Unicode is 0644) and "Alef" (its Unicode is 0627) letters are used in writing "La" word, the text viewer sees the "La" word in its special form ("ﻟﺎ"). In this case, we conclude that bit (1) is hidden. However, by writing "La" in the normal form instead of inserting the Arabic extension character between "Lam" (its Unicode is FEEO) and "Alef" (its Unicode is FE8E) characters, it is written by using the code of the correct shape of each character ("لا"). We, therefore, conclude that bit (0) is hidden. This method is so limited because it is implemented only within the "La" word only.

In [4], Al-Nazer and Gutub proposed a steganography method that utilizes the extension character (kashida) in Arabic language has been built to hide a secret message. The main goal is to maximize the capacity by utilizing all possible locations for kashida in Arabic letters. A kashida is put where it is applicable and the bit representation of the secret message has a value of (1). In other words, if the extendable character must carry a secret bit of value (1), the kashida is inserted. Otherwise, it is not inserted. Unfortunately, this method uses a kashida to hold a single bit every time.

In [5], Al-Haidari. *et. al.* developed an approach, in which the number of kashidas in one word is delimited by some equations to improve security. This method enhances the security over the previous kashida -based methods, but it still stores one bit in each kashida.

In [6], Gutub. *et. al.* use a secret key to generate random kashida characters added to words where a secret message is then embedded in the words as a watermarking code. The first addition of random kashidas is designed for a confusion purpose to ensure security. After that, selected kashidas are embedded based on the needed secret data to form the watermarking process in order to save its applications. This method uses extra kashidas to ensure security. Nonetheless, this maximizes the size of the cover text needed to hide the secret message.

In [7], Mahfoodh. *et. al.* presented method added one kashida to represent the secret bit (0) and two consecutive kashidas when the hidden bit is (1). The number of kashidas needed using this approach is huge. Besides the noticeable increasing in the file size, the extensive use of kashidas affects the security.

Odeh and Elleithy [8], proposed a letter connectivity and extension to hide one bit approach. Moreover, they used zero-width letter, which is a letter used to connect two letters with a small

effect on the shape of the two connected letters and no effect on the meaning of the word, to hide two bits in each connective character. Table 1 shows the coding deployed in this method.

Table 1. Coding of method

Extension	Zero Width	Code	Letter effect
No	No	00	No effect
Yes	No	01	Extension
No	Yes	10	Zero Width
Yes	Yes	11	Extension + Zero Width

This method merges two techniques to achieve the goal of hiding a secret message in some cover media prior to the encoding process. The first technique is to add a kashida and the second is to add a letter named 'zero-width letter', which has its own Unicode symbol (U+200D). This character causes the previous letter to appear as a connected letter. In this study, we develop a new method in which a letter can hide up to two bits by using only kashidas instead of using kashidas technique as well as zero-width technique.

In [9], Alginahi. *et. al*. inserted kashidas before a specific list of characters. If a kashida is inserted before any of these characters, it means that there is a secret bit (1), and if there is one of these characters without a kashida before it, this means that there is a secret bit (0). This method gives better security than others, but it provides less capacity due to the addition of kashidas before a specific number of characters as well as the representation of only one bit at a time.

In [10], Odeh. *et. al.* applied four scenarios randomly in order to improve data privacy. These four scenarios are. First scenario, adding Kashida after pointed letters to hide one, otherwise, hide zero. Second scenario, adding Kashida after non-pointed letters to hide one, otherwise, hide zero. Third scenario, adding Kashida after letters to hide one, otherwise, hide zero. Fourth scenario, adding Kashida after letters to hide zero, otherwise, hide one. Then, random selection is deployed to select one of these scenarios. However, each one of these scenarios can hide one bit at a time (either 1 or 0) in each extendable character.

In [12], Alginahi. *et. al.* encoded the original text depending on a predefined key. A kashida is placed after a set of characters considering whether the letter is assumed to have high frequency of recurrence in general or not. Each letter hides only one bit at a time.

In [15], Jabri and Ibrahim converted the secret message to an encrypted bits set. Then, they exploited the spaces and the extendable characters to hide the encrypted bits. If the secret bit to be embedded is one and the current character in the cover text is an extendable character, this letter is extended by a kashida, and no kashida is added if the secret bit to be embedded is zero. In the same way, if the current character in the cover text is a space between two words and the secret bit to be embedded is one, an additional space is added after the original one, and no additional space is added if the secret bit to be embedded is zero. This method uses the extendable characters as well as the spaces between letters to hide one bit at a time.

3. THE PROPOSED APPROACH

3.1. EMBEDDING PHASE

The proposed method depends on the nature of the letter before and after the kashida instead of the nature of just one of the surrounding letters as used in the previous methods.

In Arabic language, 15 letters out of 28 alphabet letters are pointed by one, two or three points. Hence, we can use the pointed letters to hide some secret bits and the unpointed letters to hide some other bits. In this study, the nature of the letter, if it is a pointed letter or an unpointed letter, is exploited to hide some secret bits in an extension (kashida) between two letters. It is used as a formatting technique in order to lengthen a specific letter without having any change on the meaning of the word. For example, the word "كان" consists of three letters ("ن","ا","ك"). If we add a kashida after the first letter, the word will be:"كـان". The word still has the same meaning it had before adding the kashida despite the fact that it consists now of four letters ("ن","ا","-","ك").

The embedding process starts with dividing the cover text into two blocks in which each block is being dealt with using a special table. The dividing process is achieved by counting the number of words of the cover text (m words), then dividing the number of words by two. After that, we take the integer number resulted (n). The words from 1 to n are then being treated using a special table, while the words from n+1 to m are being treated using another special table. In addition, the secret message is divided into small blocks. Each one of them consists of two bits. Table 2, shows the coding used for the first block.

Table 2. Embedding Code of the First Block

Secret Bits	**The Letter before Kashida**	**The Letter after Kashida**
00	Pointed	Pointed
01	Pointed	Unpointed
10	Unpointed	Unpointed
11	Unpointed	Pointed

While for the second block, Table 3 is deployed.

Table 3.Embedding Code of the Second Block

Secret Bits	**The Letter before Kashida**	**The Letter after Kashida**
00	Unpointed	Unpointed
01	Unpointed	Pointed
10	Pointed	Pointed
11	Pointed	Unpointed

For instance, to hide the secret message: "110100010011" in the cover text:
" بلغت الاستثمارات السعودية في الأردن نحو عشرة مليارات دولار ", we divide the secret message into blocks, each two bits by their own:

Table 4. Secret Message

110100010011					
11	01	00	01	00	11

The cover text is divided into two blocks by dividing the number of the words of the cover text by two:

Table 5. Cover Text

(cover text) بلغت الاستثمارات السعودية في الأردن نحو عشرة مليارات دولار	
بلغت الاستثمارات السعودية في (block no.1)	الأردن نحو عشرة مليارات دولار (block no.2)

The embedding process starts from right to left by taking the first two bits which are "11", and then we search for an un-pointed letter followed by a pointed letter in the first block of the cover text, to put a kashida between them, since "11"means that there must be a kashida between an un-pointed letter and a pointed letter (according to the table specified for the first block). The first case we find is that the letter "ل" is followed by the letter "غ", and thus we insert a kashida between the two letters and move on to the next two bits which are "00". We then search for a pointed letter followed by another pointed letter starting from the letter which is placed after the kashida we have inserted "غ". The first un-pointed letter followed by a pointed letter we find here is the letter "غ" followed by the letter "ت". Then, we continue to the next two bits "01" and search for a pointed letter followed by an un-pointed letter. Here, we find that the letter "ث" is followed by the letter "م", so we insert a kashida between them. After that, we take the next two bits which are "00", and search for a pointed letter followed by another pointed letter in the cover text to put a kashida between them. The first case we find is that the letter "ي" is followed by the letter "ة", so we insert a kashida between the two letters and move on to the next two bits which are "01". We now search for a pointed letter followed by an un-pointed letter. While we search, we reach the end of the first block without finding a pointed letter followed by an un-pointed letter, so we move to the second block. According to the table specified for the second block, "01" means that there must be a kashida between an un-pointed letter and a pointed letter. The first case we find is the letter "ع" and the letter "ش", so we put a kashida between them. After that, we continue to the last two bits "11" and search for a pointed letter followed by an un-pointed letter. Here, we find that the letter "ش" is followed by the letter "ر". Consequently, we insert a kashida between them. At this point, the embedding process comes to an end since all the secret bits have been embedded within the cover text.

- **Original Kashida Case:**

In order to distinguish between the original kashida in the text and the added kashidasin the embedding process, an additional kashida is added after any original kashida in the sentence. For example, if the cover text is " ينـادي الإسلام للسلام بين الناس، ويدعو إلى التسامح ونبذ التعصب", and there is an original kashida in the word "ينـادي" between the letters "ن" and "ا", the system, in this case, adds another kashida after this kashida in order to notify the extractor to ignore it when extracting the hidden bits. If we enter this sentence as the cover text and enter the secret message "01", "01" means that there must be a kashida between a pointed letter and an un-pointed letter, according to the table specified for the first block. The first case the extractor finds is the letter "ن" followed by the letter "ا" in the first word ("ينـادي"), but since there is an original kashida between them, it adds another kashida after the original one to inform the extractor that this is an original kashida, then continues searching for a pointed letter followed by an un-pointed letter. It finds the letter "ن"

followed by the letter "ا" in the fifth word ("الناس"), so it adds a kashida between them. Since there are no more bits in the secret message, the embedding process ends.

- **Embedding Algorithm:**

The embedding process deals with four groups of letters:
- Pointed letters that accept a kashida after them (PL1):
(ش، ث، ي، ق، ت، ن، ف، غ، ظ، ض، خ، ب)
- Pointed letters that accept a kashida before them (PL2):
(ة، ش، ث، ي، ق، ت، ن، ف، غ، ظ، ض، ز، ذ، خ، ج، ب)
- Un-pointed letters that accept a kashida after them (UPL1):
(ص، ط، ع، ك، ل، ه، س، م، ح)
- Un-pointed letters that accept a kashida before them (UPL2):
(ك، ل، م، س، ه، و، ص، ط، ع، ر، د، ح، أ، ا، آ)
Figure 1, shows the proposed embedding algorithm.

Emb_Alg

Input : Cover Text (CT), Secret Message (SM)
Output: Stego Text (ST)

1) Let i:= -1, c:= -1, m = No. of words in CT
2) If m:= 1
   Block 1 := one word
   Block2 := empty
   Else
   Block 1 := words(1 To n = trun(m/2))
   Block2 := words (n+1 To m)
3) i:= i + 2
   Read 2 bits (b1, b2) ∈SM, at positions i, i+1.
4) c:= c + 1
   read 2 characters $(x1, x2) \in$ CT, at positions c, c+1.

If $(x1 \&\& x2) \in block1$ **then** {
If $(b1 == 0 \;\&\&\; b2 == 0)$ *then*{
If $x1 \in PL1 \;\&\&\; x2 \in PL2)$ *then* {
- insert (-) after x1
- If (i := length (SM) || c:= length (CT) then EXIT
Else go to (3) } }
Else go to (4)
Else if $(b1 == 0 \;\&\&\; b2 == 1)$ *then*{
If $x1 \in PL1 \;\&\&\; x2 \in UPL2)$ *then*{
-insert (-) after x1
- If (i := length (SM) || c:= length (CT) then EXIT
Else go to (3) } }
Else go to (4)
Else if $(b1 == 1 \;\&\&\; b2 == 0)$ *then*{
If $x1 \in UPL1 \;\&\&\; x2 \in UPL2)$ *then*{
-insert (-) after x1
- If (i := length (SM) || c:= length (CT) then EXIT

Else go to (3) } }
Else go to (4)

Else

If $x1 \in UPL1 \&\& x2 \in UPL2)$ *then*{
-insert (-) after x1
- If (i := length (SM) || c:= length (CT) then EXIT
Else go to (3) }
Else go to (4) }*// End of Block 1*

Else // x1 && x2 belongs to block 2

If $(b1 == 1 \&\& b2 == 0)$*then*{
If $x1 \in PL1 \&\& x2 \in PL2)$ *then* {
- insert (-) after x1
- If (i := length (SM) || c:= length (CT) then EXIT
Else go to (3) } }
Else go to (4)

Else if $(b1 == 1 \&\& b2 == 1)$*then*{
If $x1 \in PL1 \&\& x2 \in UPL2)$ *then*{
-insert (-) after x1
- If (i := length (SM) || c:= length (CT) then EXIT
Else go to (3) } }
Else go to (4)

Else if$(b1 == 0 \&\& b2 == 0)$*then*{
If $x1 \in UPL1 \&\& x2 \in UPL2)$ *then*{
-insert (-) after x1
- If (i := length (SM) || c:= length (CT) then EXIT
Else go to (3) } }
Else go to (4)

Else

If $x1 \subset UPL1 \&\& x2 \subset PL2)$ *then*{
-insert (-) after x1
- If (i := length (SM) || c:= length (CT) then EXIT
Else go to (3) }
Else go to (4) }*// End of Block 2*

Figure 1. Embedding Algorithm

3.2. EXTRACTING PHASE

In a similar way to the embedding process, the extracting of the secret message from the stego text starts by dividing the message (stego text) into two blocks in which each block is being considered using a special table. The dividing process is achieved by counting the number of words of the cover text (m words). Then we divide the number of words by two to get the integer number (n). The words from 1 to n are then being treated using a special table and the words from n+1 to m are being treated using another special table. The extractor starts dealing with the first block by searching from right to left for a kashida. If a kashida is found, the extractor takes the letter before the kashida as well as the letter after it and looks in the extracting table specified for the first block in order to extract the two bits which the founded kashida represents. These two bits are inserted in the secret message obtained. Then, the extractor continues searching for another kashida in the first block. If the extractor finds a kashida, it looks for the nature of the letters surrounding the kashida and looks in the first

block extracting table to extract the two bits that the kashida represents. If the first block is finished, the extractor moves to the second block and starts searching for kashidas in the same way. If any kashida is found, the extractor looks at the table of the second block to interpret what the kashida represents. For example, if we have the stego text: " الـسلام عليـكم ورحمـة الـله وبركاتـه", this sentence consists of five words. Each one of these words has one kashida (the word"الـسلام " has a kashida between the letters "ل" and "س". The word "عليـكم" has a kashida between the letters "ي" and "ك". The word "ورحمـة" has a kashida between the letters "م" and "ة", and the word "الـله" has a kashida between the letters "ل" and "ل"). The decoder divides this sentence into two blocks by dividing the number (5) by (2). The integer number it gets is (2), so the first block consists of the first two words and the second block consists of the rest words of the sentence.

The first block"الـسلام عليـكم" is treated using Table 6.

Table 6. Extracting Table of the First Block

Secret Bits	The Letter before Kashida	The Letter after Kashida
00	Pointed	Pointed
01	Pointed	Un-pointed
10	Un-pointed	Un-Pointed
11	Un-pointed	Pointed

The Extracting process starts from the right to the left searching for a kashida. If the kashida is found and the surrounding letters of it belong to one of the two sets (pointed or un-pointed letters), it looks for the two bits this kashida represents. The first kashida we find here is placed in the first word which belongs to the first block ("الـسلام") between the letter "ل" and the letter "س", which are an un-pointed letter and another un-pointed letter, so the extractor looks into the table to find the two bits represented. These two bits are "10", so these bits are inserted in the secret message extracted by the extractor. After that, we move to the left, searching for another kashida. We will find another one in the second word ("عليـكم") between the letter "ي" and the letter"ك", which are a pointed letter and an un-pointed letter, according to the table. This reflects the two bits: "01". This is the last existing kashida in the first block. Consequently, the next kashidas found will be extracted using Table 7, which specified for the second block:

Table 7. Extracting Table of the Second Block

Secret Bits	The Letter before Kashida	The Letter after Kashida
00	Un-pointed	Un-pointed
01	Un-pointed	Pointed
10	Pointed	Pointed
11	Pointed	Un-pointed

The first kashida found in the second block is placed in the word "ورحمـة" between the letters"م" and "ة", which represents the two bits: "01". Since it is located between an un-pointed letter and a pointed letter, the next kashida we find is the kashida in the word "الـله" between the letters "ل" and "ل". We add the two bits:"00" to the secret message extracted because the kashida was found between an un-pointed letter and another un-pointed letter. The last kashida in the second block is placed in the word "وبركاتـه" between the pointed letter "ت" and the un-pointed letter "ه", so we add "11" to the secret message extracted. The secret message obtained is: "11 00 01 01 10". Figure 2, shows the proposed extracting algorithm.

- **Extracting Algorithm:**

Input: Stego text, pointed characters list, un-pointed characters list
Output: Secret message
Abbreviations: pointed characters list: PL, un-pointed characters list: UPL, character: char

1. Get stego text
2. If the number of the words of the cover text is 1
   The first block of the cover text consists of the word number one
   The second block is empty
   Else
   Divide the cover text into two blocks from right to left:
   N:= number of the words of the cover text
   J:= integer value results from dividing $\lfloor N/2 \rfloor$
   The first block of the cover text: (words from 1 to J)
   The second block of the cover text: (words from J+1 to N)
3. While a character in the first block exists (from right to left):
   If the character is "_":
   If the character before it $\in$ PL
   If the character after it $\in$ PL
   Add 00 to the secret message
   Else if the character after it $\in$ UPL
   Add 01 to the secret message
   Else if the character before it $\in$ UPL
   If the character after it $\in$ UPL
   Add 10 to the secret message
   Else if the character after it $\in$ PL
   Add 11 to the secret message
4. While a character in the second block exists(from right to left):
   If the character is "_":
   If the character before it $\in$ UPL
   If the character after it $\in$ UPL
   Add 00 to the secret message
   Else if the character after it $\in$ PL
   Add 01 to the secret message
   Else if the character before it $\in$ PL
   If the character after it $\in$ PL
   Add 10 to the secret message
   Else if the character after it $\in$ UPL
   Add 11 to the secret message
5. Output: secret message

Figure 2. Extracting Algorithm

3.3. SYSTEM DESIGN

The implementation of the proposed method is represented by a system designed using c#. This system is called "ATSKS", which refers to "Arabic Text Steganography using kashida System". The system processes the operations of embedding as well as extracting, but before that there is a

hashing phase used to secure the system. If a user passes it correctly, they will enter the correct system. Otherwise, they will enter a fake system. Three tables are used in this system; one table for hashing and two for embedding as well as extracting.

4. EXPERIMENTAL RESULTS AND DISCUSSION

4.1. CAPACITY

The capacity ratio is computed by dividing the amount of hidden bytes over the size of cover text media in bytes as in [5]. One of the major contributions of the proposed method introduced in this research is to hide two bits using only one kashida depending on its position between two letters. This method is supposed to increase the capacity of the hidden message since a kashida can hide two bits. In addition, there is no need to merge two methods to hide two bits in one kashida. We have used 10 different essays as cover text files of various sizes to hide secret data. Then capacity ratio of each cover text file is computed. Table 8, shows the capacity ratio of the proposed method. We have also calculated the maximum possible capacity (optimistic case).

Table 8. Capacity ratio of the proposed method

Essay No.	Website Address	Cover Text Size (Bytes)	No. of Hidden Bits	Capacity Ratio (%)
1	http://motaded.net/show-2567052.html	362	352	12.15
2	http://www.shorouknews.com/news/view.aspx?cdate=04032016&id=944d3cee-c368-4864-8420-8b5e6839fb65	674	664	12.31
3	http://www.alanbatnews.net/print.php?nid=120388	924	896	12.12
4	http://assabeel.net/culture/item/161130	1036	1056	12.74
5	http://www.al-ayyam.ps/ar_page.php?id=109f47edy278874093Y109f47ed	1174	1192	12.70
6	http://www.alriyadh.com/1134698	1203	1224	12.72
7	http://www.elkhabar.com/press/article/101630	2061	2056	12.47
8	http://www.ahram.org.eg/NewsQ/483345.aspx	2361	2408	12.74
9	http://www.alwatannews.net/NewsViewer.aspx?ID=118537	5442	5544	12.73
10	http://www.alrai.com/article/771518.html	20435	20400	12.48

The capacity ratio of the proposed approach for each one of the essays doubles the capacity ratio of most of the previous kashida approaches that hided one bit at a time. For example, the extendable characters found in essay number 1 are 178. These characters are able to store 356 bits at best using the proposed method, while they can store only 178 bits at best using the previous kashida approaches which store 1 bit at each extendable location.

In contrast with [8], we do not need to add another character beside the kashida to represent two bits. Each single existing kashida using the proposed method hides two bits by its own without merging it with a zero-width character which is used in this study just to distinguish between the original kashida and the kashida used to store secret bits. Table 9, shows the capacity ratio values

of three other methods using the10 essays in Table 8 as cover text files. We have also calculated the maximum possible capacity (optimistic case):

Table 9. Capacity ratio of methods [9, 15]

Essay No.	Cover Text Size (bytes)	No. of Hidden Bits Using Method [9]	Capacity Ratio Using Method [9] (%)	No. of Hidden Bits Using Method [15]	Capacity Ratio Using Method [15] (%)
1	362	72	2.49	248	8.56
2	674	152	2.81	464	8.61
3	924	192	2.60	616	8.33
4	1036	232	2.80	720	8.69
5	1174	232	2.47	824	8.78
6	1203	216	2.24	856	8.89
7	2061	448	2.72	1488	9.02
8	2361	480	2.54	1472	7.79
9	5442	952	2.19	3824	8.78
10	20435	4712	2.88	14392	8.80

The average capacity ratio of method [9, 15] is 2.57%, and 8.63% respectively. The average capacity ratio of the proposed method is 12.52%. It is obvious that the capacity ratio of the proposed method outperforms the capacity ratio achieved by [9, 15].

We have calculated the results by taking all the extendable characters in the cover text. Then, we divided them by 8 to get the number of secret bytes that we may hide within the cover text. After that, we divided the size of secret message (bytes) by the size of cover text (bytes) to get the capacity ratio. By contrast, in [15], they took all the extendable characters as well as the whitespaces as secret message carriers. Then, they divided them directly by the total number of characters and whitespaces. It is obvious that this way of calculation is incorrect and yield to misleading results.

4.2. EMBEDDING RATIO

Another measurement used to evaluate the proposed approach is the "Embedding Ratio(ER)", as in [13] which is used to determine the total fitness of the hidden text that can be embedded in cover text:

$$ER = \frac{\text{total bits of stego text} - \text{totalbitsof covertext}}{\text{totalbitsof covertext} + \text{totalbitsofhiddentxt}} \text{ X } 100\% \quad \text{.....(1)}$$

$$ER = \frac{\text{total number of embedded bits}}{\text{total bits of expected stego text}} \text{ X } 100\% \quad \text{.....(2)}$$

We have calculated the embedding ratio of the same essays:

Table 10. Embedding Ratio Values of the Proposed Method

Essay No.	Cover Text Size (bytes)	Number of Embedded Bits	Embedding Ratio (%)
1	362	352	10.84
2	674	664	10.96
3	924	896	10.81
4	1036	1056	11.30
5	1174	1192	11.26
6	1203	1224	11.28
7	2061	2056	11.09
8	2361	2408	11.30
9	5442	5544	11.30
10	20435	20400	11.09

These embedding ratio values exceed the values of the embedding ratio for the previous methods which hide one bit at a time, because the proposed method uses less number of kashidas to embed the same number of secret bits within the same cover text. The following table shows the embedding ratio values of the same 10 essays used as cover text files which have been used in Table 10 for two other methods:

Table 11. Embedding Ratio Values of Three Previous Methods

Essay No.	Cover Text Size (bytes)	No. of Embedded Bits Using Method [9]	Embedding Ratio Using Method [9] (%)	No. of Embedded Bits Using Method [15]	Embeddi ng Ratio Using Method [15] (%)
1	362	72	2.43	248	7.88
2	674	152	2.74	464	7.92
3	924	192	2.53	616	7.69
4	1036	232	2.72	720	7.99
5	1174	232	2.41	824	8.06
6	1203	216	2.19	856	8.17
7	2061	448	2.65	1488	8.28
8	2361	480	2.83	1472	7.23
9	5442	952	2.14	3824	8.07
10	20435	4712	2.80	14392	8.09

The average embedding ratio of method [9, 15] is 2.54%, and 7.94% respectively. While, the proposed method achieves 11.12% as average embedding ratio. This was expected, since a capacity ratio affects positively or negatively the embedding ratio.

5. CONCLUSION

A new method to hide secret message within Arabic texts has been proposed. This method depends primarily on the nature of letters; whether they are pointed or un-pointed letters. We have exploited this feature of Arabic text to add a kashida (-). Since there are two cases of each letter; pointed or un-pointed, a table of four cases is used to add a kashida between two letters every time to hide two bits in each kashida. The most common kashida-based methods hided just one bit in each kashida or used a kashida as well as a zero-width character to hide two bits, while the

proposed method hided two bits using kashida without the need to add the zero-width character. Adding zero-width character increases the file size dramatically, which affects the security measures. By hiding two bits in each kashida, the capacity is remarkably increased as compared with some of the well-known kashida-based approaches. The dependency on the nature of both of the surrounding letters of the kashida as well as dividing the cover text into two blocks, each one is being dealt with in a different way, increases the security of the proposed method. Since, intruders are not able to expect the method of extraction. Furthermore, the original kashida case is considered. Ignoring the original kashida in the cover text affects the accuracy of extraction phase. A system of embedding the secret message within a cover text and extracting the secret message from a stego text has been built. This system has a hashing phase in order to distinguish between the authorized user and the un-authorized user, which may increase the robustness of the system. In some cases, the capacity ratio of the proposed method may affected by the sequences of the secret bits, and suitable appearances of the targeted kashida(-). This might be considered as limitation that may yield to undesirable results in such cases. As a future work, we should overcome this drawback by proposing a suitable method to make use of all kashidas irrespective to the sequences of the secret bits.

REFERENCES

[1] G. Kipper, (2003) Invistigator's Guide to Steganography, Auerbach Publications, ISBN 9780849324338, October 27.

[2] Gutub, A. and M. M. Fattani, (2007) "A Novel Arabic Text Steganography Method Using Letter Points and Extensions", International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol: 1, No: 3.

[3] M. S. Shahreza and M. H. Shahreza, (2008) "An Improved Version of Persian/Arabic Text Steganography Using "La" Word", Proceedings of IEEE 2008 6th National Conference on Telecommunication Technologies and IEEE 2008 2nd Malaysian conference on Photonics, 26-27 August, Putrajaya, Malaysia, pp: 372-376.

[4] A. Al-Nazer, A. Gutub, (2009) "Exploit Kashida Adding to Arabic E-Text for High Capacity Steganography", Proceedings of Third IEEE International Conference on Network and System Security, Pages: 447-451. DOI: 10.1109/NSS.2009.21.

[5] F. Al-Haidari, A. Gutub, K. Al-Kahsah, and J. Hamodi, (2009) "Improving Security and Capacity for Arabic Text Steganography Using 'Kashida' Extensions", Proceedings of IEEE/ACS International Conference on Computer Systems and Applications.pp: 396-399.DOI: 10.1109/AICCSA.2009.5069355

[6] A. Gutub, F. Al-Haidari, K. Al-Kahsah, and J. Hamodi, (2010) "E-Text Watermarking: Utilizing 'Kashida' Extensions in Arabic Language Electronic Writing", Journal of Emerging Technologies in Web Intelligence, Vol. 2, No. 1, Pages: 48-55. DOI: 10.4304/jetwi.2.1.48-55.

[7] A. Gutub, W. Al-Alwani, and A. Bin Mahfoodh, (2010) "Improved Method of Arabic Text Steganography Using the Extension 'Kashida' Character", Bahria University Journal of Information & Communication Technology Vol. 3, Issue 1, pp: 68-72.

[8] A. Odeh and K. Elleithy, (2012) "Steganography in Arabic Text Using Zero Width and Kashidha Letters", International Journal of Computer Science & Information Technology (IJCSIT), Vol. 4, Pages: 1-11. DOI: 10.5121/ijcsit.2012.4301.

[9] Y. M. Alginahi, M. N. Kabir and O. Tayan, (2013) "An Enhanced Kashida-Based Watermarking Approach for Increased Protection in Arabic Text-Documents", Proceedings of International Conference on Electronics, Computer and Computation (ICECCO). DOI: 10.1109/ICECCO.2013.6718288

[10] A. Odeh, K. Elleithy and M. Faezipour, (2013) "Steganography in Arabic Text Using Kashida Variation Algorithm (KVA)", Proceedings of the 2013 IEEE Systems, Applications and Technology Conference (LISAT). DOI: 10.1109/LISAT.2013.6578239.

[11] Mersal, S., Alhazmi S., R. Alamoudi and N. Almuzaini, (2014) "Arabic Text Steganography in Smartphone", International Journal of Computer and Information Technology (ISSN: 2279 – 0764), Vol. 03, No. 02, pp. 441-445.

[12]Y. Alginahi, M. Kabir and O. Tayan, (2014) "An Enhanced Kashida-Based Watermarking Approach for Increased Protection in Arabic Text- Documents Based on Frequency Recurrence of Characters", International Journal of Computer and Electrical Engineering, Vol.6, No. 5. DOI: 10.177706/ijcee.2014.v6.857.

[13] B. Osman, R. Din and M. R. Idrus, (2015) "Capacity Performance of Steganography Method in Text Based Domain", ARPN Journal of Engineering and Applied Sciences, Vol. 10, No. 3.http://repo.uum.edu.my/id/eprint/14832.

[14] R. Saluja, K. Kanwal and S. Dahyia, (2014) "Review on Steganography for Hiding Data", International Journal of Computer Science and Mobile Computing, Vol.3 No. 4, pp: 225-229. ISSN 2320–088X.

[15] R. Jabri and B. Ibrahim, (2016) "Capacity Improved Arabic Text Steganography Technique Utilizing 'Kashida' with Whitespaces", Proceedings of the Third International Conference on Mathematical Sciences and Computer Engineering (ICMSCE2016), pp: 38-44, Langkawi, Malaysia.

AUTHORS

Jehad Q. Odeh received his Ph.D. of Computer Science from University Putra Malaysia in 2004. Currently, Dr. Jehad is an Associate Professor at the Faculty of Information Technology in Al al-bayt University, Jordan. His research areas include image retrieval and indexing, image processing, and steganography. He has published a number of papers related to these areas, in addition to supervision on many MSc theses in different computer science fields.

Ala'a M. Alhusban received here MSc. of computer Science from Al al-Bayt university in 2017. Her research areas include text retrieval and indexing, and steganography.