

A MULTI-LAYER ARABIC TEXT STEGANOGRAPHIC METHOD BASED ON LETTER SHAPING

A.F. Al Azzawi

Department of Software Engineering, Philadelphia University,
Amman, Jordan

ABSTRACT

Text documents are widely used, however, the text steganography is more difficult than other media because of a little redundant information. This paper presents a text steganography methodology appropriate for Arabic Unicode texts that do not use a normal sequential inserting process to overcome the security issues of the current approaches that are sensitive to steg-analysis. The Arabic Unicode text is kept within main unshaped letters, and the proposed method is used text file as cover text to hide a bit in each letter by reshaping the letters according to its position (beginning, middle, end of the word, or standalone), this hiding process is accomplished through multi-embedding layer where each layer contains all words with the same Tag detected using the POS tagger, and the embedding layers are selected randomly using the stego key to improve the security issues. The experimental result shows that the purposed method satisfied the hiding capacity requirements, improve security, and imperceptibility is better than currently developed approaches.

KEYWORDS

Arabic text, multi-layers, Unicode, hiding information, text steganography

1. INTRODUCTION

The image is the most well-liked carrier media for steganography due to its abundance out there on the web. Another excuse is that the images have an excellent quantity of a redundant area that is the perfect place to embed data. However, text steganography isn't commonly most popular because of the issue find a redundant area to embed bits in text files [1, 2]. The text document structure is often terribly the same as what's seen, whereas, in all alternative cover media varieties, the structure is completely different than what we tend to observe, creating the concealment of data straightforward with none notable alteration. The advantage to favoring text steganography over alternative steganography media since it is less memory occupation and easier in communication [3]

The text is one among the oldest media applied in steganography; before the electronic age, telegrams, letters, and books hid secret information among their texts. Text steganography refers to the concealing of data among the text [2] and will involve something from changing the format of the text to changing words among a text to generating arbitrary character series or utilizing of context-free grammars to come up with clear interesting texts [2]. Compared to the alternative media, text steganography needs a sound algorithm that may work below the constraint of low hiding capability, and the goal of this work is to make a secure high capability text steganography system for the Arabic language.

2. RELATED WORKS

Different methods are presented on hiding information in Arabic texts: The Dot steganography methodology is used the number of points within Arabic letters to hide data, the dot letters are employed to embed bits by slightly shift dots up to represent the embedding the bit "1" and leave the pointed letter to embed the bit "0", the robustness of this methodology is weak since it used the identical fixed font [2]. The extension methodology considered two options, the existence of the points within the letters and the repetitive extension character (Kashid) written as «←» to embed secret data bits. It uses the pointed letters with Kashid to carry the bit "1" and the unpointed letters with Kashid to carry the "0", the authors in [4] suggest an algorithm to insert Kashida letters according to four insertion scenarios and select one of the four randomly. The authors in [5], propose an algorithm utilizing of both the Kashida and small space character to hide three bits to achieve high capacity ratio, and these approaches also are helpful to different languages having a similar text to the Arabic language like Urdu and Persian scripts [6]. A hybrid Arabic text steganography method with cryptography was developed, the embedding message is hidden using a Kashida text steganography[7], and improving the hiding secrets utilizing the redundant extension Kashida character and embedding sensitive data within whitespace [8]. The capacity is also enhanced in [8] by using Kashida and the whitespace characters between words for hiding, a Kashida will be inserted to hide bit equals to one until no more additions can be handled, and white spaces are inserted between words to hide bit equals to one also, otherwise a zero bit is hidden if the location of Kashida is left empty or whitespace is kept as it is. The work in [9] proposes an algorithm that uses a small space character and Arabic extension character (Kashida) to hide only one bit while each space is used to hide three bits. Figure 1 shows an example for hiding a message using kashida-based approach.

Figure 1. Example of the Kashida method.

Secret bits	0100000101011001010000010100000100
Cover text	البر حسن الخلق والائم ما حاك في نفسك
Stego- text	البر حسن الخلق والائم ما حاك في نفسك

But the most disadvantages of kashida-based strategy is the opportunity to notice the kashida insertion and the capability of this strategy is also variable and depending on the ability of the letter to be extended with kashida[10].

Arabic diacritics methods utilize the diacritics of Arabic text to implement text Steganography[11]. Different methods are implemented to use eight different Arabic diacritical characters found in Arabic text. One of these methods, used Fatha diacritics to represent a bit equal to one and used another diacritic to represents a bit equal to zero[12], The work [13] exploits to use two Diacritics (Kasrah and Fatah) to hide information and adjusting the previous works that uses only Fathah and the work utilizing from including or omitting diacritics to hide bits as figure 2 shows an example of embedding bits using Diacritic-based approach [14]. These approaches provide a good capacity, does not need a big computational ability, and may be achieved manually, but are not imperceptible, not appropriate for sensitive text[15] and suspicions raise since it's uncommon today to transfer discretized text[16].

Cover text:	مُسْتَقْعَلٌ
Secret bits:	0 1 1 0 0 1
Stego-text:	مُسْتَقْعَلٌ

Figure 2. Example of the diacritics method.

In Arabic Unicode text steganography, the Unicode characters are used for hiding information by utilizing of two characters, zero with joiner character (ZWJ), zero with non-joiner character (ZWNJ) that respectively forces Arabic letters to be joined, or forces them from joining together. In this method, zero with joiner is used to hide the bit 1 and zero with non-joiner is used to hide the bit 0. Other methods are used the Zero-Width Character (ZWC) Unicode to hide information through inserting them in Arabic text without occupying any more space[17], and lossless compression is used to get high embedding capacity in [18], but actually, the compression algorithm leads to increase the embedding capacity.

The authors in [19] introduce a watermarking approach for hiding in Arabic text by utilizing a small space called a pseudo-space to isolate the connected letters, the bit 1 is hidden by adding pseudo-space after pointed letter, otherwise, the bit 0 is hidden by inserting pseudo-space after un-pointed letter and before normal. Figure 3 shows an example for hiding a message using Unicode-based methods where the Unicode ZWJ and ZWNJ are used.

Watermarking bits	1001011
Original text	البر حسن الطلق والإتم ما حاك في نفسك
Output text	البر حسن الطلق والإتم ما حاك في نفسك
	↑ ↑ ↑ ↑ ↑ ↑ 1 1 0 1 0 0 1

Figure 3. Example of the pseudo-space method.

This methodology is improved by authors [20] to induce a higher capacity, and better imperceptibility property and recommend to insert before/after a space depending on the dot letters or to insert three different little pseudo-space or zero-width space to enhance the capability. However, the most disadvantages of this methodology, the cover size is going to be grown rapidly in embedding a long secret message.

3. BACKGROUND

This section gives a review of the fundamental concepts to achieve our goal and to implement an Arabic text steganography method considering utilizing from a Part Of Speech (Stanford-pos tagger-full-2018) [21], replacing the original Unicode letter with corresponding a suitable Unicode that keeps the letters with the same shape, a natural Language processing Toolkit (NLTK), and Python programming language are used as a development environment to satisfy the most common criteria in an efficient design for developing a text steganography algorithm.

3.1. NATURAL LANGUAGE PROCESSING USING PYTHON

To achieve our goal, Natural Language Processing Toolkit (NLTK), and Stanford-pos tagger(full-2018) are used to implement the proposed method. The NLTK is a set of libraries that provides many functionalities, i.e tokenization, part of speech tagging, syntax analysis, and semantic analysis It is written using Python programming language by Steven Bird and Edward Loper at Pennsylvania University[22]. In this work, NLTK is used to split Arabic text file to a sequence of words, and Stanford-pos tagger is used to classify the cover text words according to its part of speech.

3.2. PART OF SPEECH

Valuable information about a word and its neighbors may be given through using a Part-of-speech (POS), Knowing a POS of the word and its neighbor words are important to deal with the grammar structure around the word [23]. In this work, Stanford-pos tagger(full-2018) is used to detect POS for the Arabic words found in the cover text and to construct multi-embedding layers.

3.3. ARABIC TEXT SHAPING

The letters within Arabic have completely different shapes per its positions within the word. The variety of those shapes are determined by its position within the word, for instance, the letter «ع» is written as «ع» at the start of the word, as «ع» within the middle of the word, as «ع» at the tip of the word, and as «ع» within the isolated type [24]. Furthermore, the Arabic text is kept in unique unshaped letters represented by a single Unicode for all shapes (beginning, middle, end, and isolated) and these unshaped letters are only rendered according to its position through system software.

Figure 4 shows the fact that Arabic text-letters are shown differently although the same Unicode code used for all shapes in the text file, and system software is used to shape the Arabic letters to their correct typographically based on the place of a letter (beginning, middle, end, or isolate).

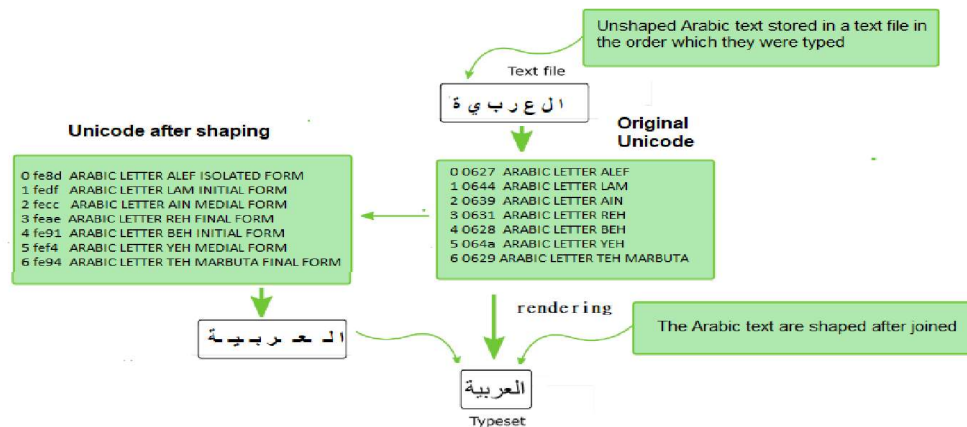


Figure 4. Arabic text shaping.

Arabic text letters may be shaped correctly according:

1. Determine the position of the letter in the word, and select a suitable form (isolate form, beginning form, middle form, and a final form). Special cases must take into account such as the letter may have an initial form, but it is in the middle of a word, For example, if Ein «ع» comes after Alef «ا», Ein «ع» has an initial form «ع» and not a middle form «ع».
2. Generate a Look up shaping table, where each row of the table represents an unshape letter with all its shape forms.
3. Map down the unshape letter to its shape form using the shaping table.
4. The proposed Arabic text steganography method is utilizing each letter in the cover text to guarantee high embedding capacity. The manipulation of the hidden message is done by reshaping the letter of the cover text to satisfy imperceptibility requirement and kept the cover text size as it.

4. PROPOSED APPROACH

The proposed approach can be used to hide any type of information as a sequence of bits in a text file. For instance, The secret message is inverted to a sequence of bits, and the hiding bits are embedded in Arabic document letters by replacing the original letters with other Unicode for the same letter that gives the same shapes.

At first, the NLTK Arabic tokenizer and Stanford-pos tagger(full-2018) tagging tool is used to split the cover document cover file into a word list where each word is associated with its POS, all words with the same POS will be grouped into a layer, number of words in each layer is computed, and the layers will be sorted in ascending order according to the number of words found in the layer. Each POS represents an embedding layer and the capacity of the document cover file is determined using the number of letters found in all words found in the document through the following equation :

$$No. of embedding bits = \sum_{i=1}^N len of word_i \quad \forall word \in Document file$$

The maximum number of embedding layers can be selected from the total number of layers, a list of non-repeated random integers from 0 to the maximum layer is generated to be used in encoding or decoding processes, and the layers are selected in sequence that appears in the random layer list and a word is selected from that layer to encode or decode a part of the secret message. As long as the binary sequence of the secret message remains, the processes are done for each word in the layers. When there is no more binary sequence secret message remains to hide the cover text contains the secret message and became stego Document text, and the secret stego key is generated at embedding phase from layer indices and length of the secrete message, and it is used to detect the integrity of document text at extracting process. The following two sections explain the detailed proposed algorithms for embedding and extracting the secret message.

4.1. EMBEDDING ALGORITHM

Input: Arabic Cover text file, Secret message to hide

Output: Stego text file, stego key

1. Read the text file and converts it to a string denoted by covertxt
2. The Arabic NLTK tokenizer module divides the covertxt into a list of words, denoted by Wordlist = (Word₁, ..., Word_m)
3. The Arabic tagging module in Stanford-pos tagger assigns a POS for each Word_i in the Wordlist and generate a Tagging list,
Tagginglist = ((Word₁, Tag₁), ..., (Word_m, Tag_k)), where Word has a POS Tag_i.
4. The cover text Tagginglist is divided into a list of layers, denoted by

Layerlist=(Layer₁, Layer₂, ..., Layer_N), where N is a number of POS, and Layer_i is a layer that has all words with the same POS.

5. Sort the Layerlist according to layer length in ascending order where

$$\text{Length of a layer } j = \text{No. of words} \in L_j$$

6. Select K embedding layers, and generate a random list containing these layers indices, I.e SelLayerindices = (10, K-1,0 ..., 4), where each index \in SelLayers < K.

7. Generate a word list to be used to embed secret message according to the selected layers indices sequence:

UsedWords= { {WL₁₁, WL₁₂, ..., WL_{1k}}, {WL₂₁, WL₂₂, ..., WL_{2k} }, {WL₃₁ , WL₃₂, ..., WL_{3k} } ..., ... }

where WL_{ij} is a word at location i of layer j.

8. Read the secret message and convert it to a binary string BinStr.

9. For each Word in the UsedWords list perform:

 Get a sequence of bits SeqBits from BinStr such that len (SeqBits) = len (Word)

 For each Bit, letter in SeqBits, Word perform:

 if Bit=zero then

 add original Unicode of Letter to Stegotxt

 else

 detect the letter position in the Word

 get Unicode of the letter corrodng to its position type add it to Stegotxt

10. Produce a key from No. of used letters to hide secret message and SelLayerindices list.

4.2. EXTRACTING ALGORITHM

Further Input: Stego text file, stego key

Output: Secret message

1. Read stego text file and converts it into a string denoted by Stegotxt

2. The Arabic NLTK tokenizer module divides the Stegotxt into a list of words, denoted by Wordlist = (Word₁, ..., Word_m)

3. Generate the binary word list

BinWordList = ((Word₁, BinWord₁),..., (Word_m, BinWord_k)), where BinWord_i is a binary string hidden in BinWord_j, using the following:

For each Word in the Wordlist list perform:

For each letter in Word perform:

 if letter= original Unicode then

 add zero to BinWord

 else

 detect the letter position in the Word

 convert Unicode of a letter to its original Unicode and add one to BinWord

 associated word with its BinWord as (word, BinWord)

4. The Arabic Stanford-pos tagger assigns a POS to each Word in the BinWordList and generates an extended Tagging list,

ExtendedTagginglist = ((Word₁, Tag₁ , BinWord₁),..., (Word_m, Tag_k, BinWord_k)), where Word_i has a POS Tag_j.

5. The cover text ExtendedTagginglist is divided into a list of layers according to Tagging, which is denoted by

Layerlist=(Layer₁, Layer₂, ..., Layer_N), N is the number of POS, and the Layer_i contains all words with the same POS.

6. Sort the Layerlist according to layer length in ascending order where

$$\text{Length of a layer } j = \text{No. of words} \in L_j$$

7. Use stego key to generate the list SelLayerindices of K embedding layers that containing the indices of these layers.

8. Generate a word list to be used to embed secret message according to the selected layers indices sequence:

UsedWords= { {WL₁₁, WL₁₂, ..., WL_{1k}}, {WL₂₁, WL₂₂, ..., WL_{2k}}, {WL₃₁, WL₃₂, ..., WL_{3k}}...,... }

where WL_{ij} is a word at location i of layer j.

9. For each Word in the UsedWords list perform:

Get a string of bits that associate Word in the ExtendedTagginglist and add it the binary secret message (BinStr)

10. Convert the binary string BS to Unicode representation.

4.3. EXAMPLE FOR THE PROPOSED APPROACH

Consider the document cover file contains the following text

"إخفاء المعلومات هي طريقة لحجب البيانات في وسيط رقمي لتبادل المعلومات بين طرفين", and the binary secret message "1100010110111000101111100100101011000101011".

The following shows the steps of executing the algorithm in section 4.1 :

1. The steps 1, 2, and 3, read the cover text from the word document file, divide the cover text into a list of words, detect POS for the words, and generate a sequence of words with associated POS as shown below:

[("إخفاء", 'NN'), ("المعلومات", 'DTNNS'), ("هي", 'PRP'), ("طريقة", 'NN'), ("لحجب", 'NN'), ("البيانات", 'DTNNS'), ("في", 'IN'), ("وسيط", 'NN'), ("رقمي", 'JJ'), ("لتبادل", 'NN'), ("المعلومات", 'DTNNS'), ("بين", 'NN'), ("طرفين", 'NNS')]

2. The steps 4 and 5, detect all layers and sort these according to its length, where each layer indicated by single POS and contains words with POS as shown below:

Layer List = [('NN', (6, 0)), ("إخفاء", "طريقة", "لحجب", "وسيط", "لتبادل", "بين"), ('DTNNS', (3, 0)), ("المعلومات", "البيانات", "المعلومات"), ('PRP', (1, 0)), ("هي"), ('IN', (1, 0)), ("في"), ('JJ', (1, 0)), ("رقمي"), ('NNS', (1, 0)), ("طرفين")]

Six layers are found, the first layer has "NN" POS with 4 words, the second layer has "NNP" POS with 3 words and so on.

3. The steps 6 and 7, select a number of embedding layers and generate random indices for the selected layers, and assume four layers from layer list as shown below:

Selected Layers = [('NN', (6, 0)), ("إخفاء", "طريقة", "لحجب", "وسيط", "لتبادل", "بين"), ('DTNNS', (3, 0)), ("المعلومات", "البيانات", "المعلومات"), ('PRP', (1, 0)), ("هي"), ('IN', (1, 0)), ("في")]

and selected layers will arrange according to according to random layer indices will be:

Arranged Selected Layers = [('NN', (6, 0)), (" , "لتبادل", "وسيط", "لحجب", "طريقة", "اخفاء", "بين", "بين"),
 ('IN', (1, 0)), ("في",
 ('DTNNS', (3, 0)), ("المعلومات", "البيانات", "المعلومات",
 ('PRP', (1, 0)), ("هي")]

Each time a word is selected for concealment from a layer, starting from the first layer and then the next subsequent layer is selected, if a layer has no more words, a move is done to the next. The sequence of words for the example will be:

Embedding word list = [("اخفاء", "في", "المعلومات", "هي") → From layer [0, 3, 1, 2]
 ("طريقة", "البيانات",) → From layer [0, 3]
 ("لحجب", "المعلومات",) → From layer [0, 3]
 ("وسيط",) → From layer [0]
 ("لتبادل",) → From layer [0]
 ("بين",) → From layer [0]

4. The step 9, takes each time a word from embedding word list and a chunk of bits from the binary message with length equal, each bit from secret message will be hidden in a letter. If the bit is zero, the letter will be in the original unshaped Unicode the letter will be in a shape Unicode depends on its location (at the beginning, middle, at the end or isolated). Table 1 shown the words of cover text before and after embedding with its Unicode, and the stego-text will be identical to cover text without adding any more bytes.

Table 1. Cover text with its Unicode before and after embedding.

Bits	Before Embedding		After Embedding			
	Word	Unicode for Unshaped letters	Unicode for shaped letters		Word	
"11000"	اخفاء	0627 ALEF KHAH 0641 ف FEH ALEF 0621 ء HAMZA	خ 062e	fea7 ALEF ISOLATED KHAH INITIAL 0641 ف FEH 0621 ء HAMZA	خ 0627 ALEF اخفاء	
"10"	في	0641 ف FEH YEH	ي 064a	fed3 FEH INITIAL 064a ي YEH	في	
"110111000"	المعلوما ت	0627 ALEF LAM 0645 م MEEM AIN 0644 ل LAM WAW 0645 م MEEM ALEF 062a ت TEH	ل 0644 ع 0639 و 0648 ا 0627	0627 ALEF INITIAL 0645 م MEEM MEDIAL FEE0 MEDIAL FINAL 0645 م MEEM 062a ت TEH	fedf LAM fecc م AIN feee و WAW 0627 ALEF	المعلوما ت
"10"	هي	0647 ه HEH	ي 064a	feeb ه HEH INITIAL 064a ي YEH	هي	

		YEH				
“11111”	طريقة	0637 ط TAH REH 064a ي YEH QAF 0629 ة TEH MARBUTA	ر 0631 ق 0642	fec3 ط TAH INITIAL FINAL fef3 ي YEH INITIAL MEDIAL fe94 ة TEH MARBUTA FINAL	ر feae REH ة fed8 QAF	طريقة
“10010010”	البيانات	0627 ا ALEF LAM 0628 ب BEH YEH 0627 ا ALEF NOON 0627 ا ALEF TEH	ل 0644 ي 064a ن 0646 ت 062a	fe8d ا ALEF ISOLATED LAM 0628 ب BEH MEDIAL 0627 ا ALEF NOON fe8e ا ALEF FINAL TEH	ل 0644 ي fef4 YEH ن 0646 ت 062a	البيانات
“1011”	لحجب	0644 ل LAM HAH 062c ج JEEM BEH	ح 062d ب 0628	fedf ل LAM INITIAL HAH fea0 ج JEEM MEDIAL BEH FINAL	ح 062d ب fe90	لحجب
“000101011”	المعلومات	0627 ا ALEF LAM 0645 م MEEM AIN 0644 ل LAM WAW 0645 م MEEM ALEF 062a ت TEH	ل 0644 ع 0639 و 0648 ا 0627	0627 ا ALEF LAM 0645 م MEEM MEDIAL 0644 ل LAM WAW FINAL 0645 م MEEM ALEF FINAL fe95 ت TEH ISOLATED	ل 0644 م fecc AIN و feee ا fe8e	المعلومات

The step 10, generates the key from the number of used letters for embedding and the random indices for embedding layer order, as shown below:

32, [0, 3, 1, 2]

Thus, the hiding capacity of the example will be:

No of characters in cover text=79

No of letters=67

capacity = (67)/(79)*100=84.8

4.4. IMPLEMENTATION

A multi-layers Arabic Text Steganography based word tagging and letter reshaping are implemented by using Python programming language, Natural Language Toolkit (NLTK), and Stanford-pos tagger (full-2018), where NLTK is used to tokenize Arabic text and Stanford-postage is used to classify Arabic words found in the cover text according to POS tagging.

The complete implementation for the method to gather with its test cases and execution traces are found in the link

<https://drive.google.com/drive/folders/1Nu64Dxu0wZRcQkLFnbpCmUuTFvaTZQVO?usp=sharing>.

5. EVALUATION OF THE PROPOSED ARABIC TEXT STEGANOGRAPHY

Many methods are proposed for Arabic text steganography using different approaches such as Kashida, Pseudo-space, and Diacritics methods [5, 6, 10, 11, 14, 15, 16]. But Kashida with Moon and Sun Letters method [25], Open Word Space method (mix different spaces with normal space) [10] and Kashida and small space method [5] are achieved more better than others based on the common criteria embedding capacity, invisibility, security, and robustness[5, 20]. The comparison of this work is done based on these methods and for best comparison, cover texts with the identical size are taken from Open Source Arabic Corpus (OSAC, 2010) [26].

Hiding capacity ratio is a major performance for evaluating a steganography algorithm, where the capacity ratio is computed by [27]

$$Capacity\ ratio = \frac{number\ of\ hiding\ bits}{Length\ of\ stego\ text} \times 100\%$$

The proposed method embeds more bits than Open Word Space method and Moon and Sun letters method, but less than the Kashid and space method as shown from table 1 and figure 5. Kashid and space method utilizes from Arabic Kashida extension characters and the little space characters where Kashida character or space character is a combination of two approaches and used to insert a character into the cover text to hide either 1 bit or 3 bits which increases the cover size unacceptably affecting its quality, while the proposed method may hide a bit in each letter without inserting any character and its capacity also will be increased rapidly if it combined with other approaches.

Table 2. Capacity Comparisons with other methods using different text samples.
CL= Cover Text Length

Method	Maximum Secret Message Length				Capacity Ratio%			
	Test1 CL 155	Test2 CL 600	Test3 CL 1100	Test4 CL 1602	Test1 CL 155	Test2 CL 600	Test3 CL 1100	Test4 CL 1602
Proposed	133	503	902	1340	0.86%	0.84%	0.82%	0.84%
Kashid and space	144	528	992	1490	93%	88%	90.18%	93%
Open Word Space	96	408	720	1040	61%	68%	65.4%	65%
Moon and Sun letters	91	405	716	1029	58%	67%	65%	64%

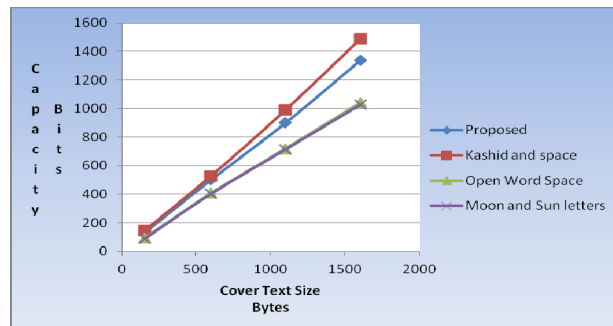


Figure 5. Capacity Comparisons chart for different text samples.

The proposed methodology has more imperceptibility than other methods due its visual similarity between the stego text and the cover text and table 3 shown the result of applying our proposed method and alternative methods[5] to the identical cover text, cause of its manipulation of hidden message is done through reshaping the letters of the Arabic cover text that kept the cover text as it.

Table 3. Applying text steganography methods to same Arabic text.

Method	Stego text
Proposed method	”تنقسم علوم الصحة الى قسمين دراسة جسم الإنسان والبحث لتعزيز معرفتنا بالآليات التي يعمل بها الجسم الحي وممرضاته وعلم الصحة التطبيقي الذي يعتم بتطبيق هذه المعرفة“
Kashid and space	”تنقسم علوم الصحة إلى قسمين دراسة جسم الإنسان والبحث لتعزيز معرفتنا بالآليات التي يعمل بها الجسم الحي وممرضاته وعلم الصحة التطبيقي الذي يهتم بتطبيق هذه المعرفة“
Open Word Space	”تنقسم علوم الصحة إلى قسمين دراسة جسم الإنسان والبحث لتعزيز معرفتنا بالآليات التي يعمل بها الجسم الحي وممرضاته وعلم الصحة التطبيقي الذي يهتم بتطبيق هذه المعرفة“
Moon and Sun letters	”تنقسم علوم الصحة إلى قسمين دراسة جسم الإنسان والبحث لتعزيز معرفتنا بالآليات التي يعمل بها الجسم الحي وممرضاته وعلم الصحة التطبيقي الذي يهتم بتطبيق هذه المعرفة“

There is a safety level of security that keeps attackers from recognizing the secret message visually or from extracting it from stego text[7]. This measure relies upon embedding capacity, invisibility, and robustness. A productive steganography approach must give an ideal balance between these criteria. If an approach gives high capacity, the secret message is absolutely imperceptible, and robustness is high, then the algorithm security can be calculated by the following:

The main disadvantages of other methods: the kashida and discretize characters are noted especially at inserting along secret message, and suspicions are raise considering it's not common today to send discretized text or using a lot of kashida extensions letters in Arabic text, and the cover text may be enlarged in size unexpectedly in case of inserting any kind of character (space, kashida, and discrete)

Besides that, the other methods considered to embed first bit(s) of the secret message at beginning position of the cover text, the next bit(s) of the message will be embedded at the following position and so on and the message will be embedded in only one stream, which leads that the message is destructed more easily than distributed in the cover text. In the proposed approach, even if the hidden message is discovered, it cannot be decoded or dropped from the stego text, since it is used a multi-layer embedding way to hide information through classifying text words into its POS tags, construct an embedding layer for all words with the same tag, and using these layers randomly in inserting message process. In the light of this evidence, our method provides a good balance between embedding, imperceptibility, and security criteria without affecting the Arabic text quality.

6. CONCLUSIONS

This work is proposed an Arabic text steganography method by reshaping the Arabic letters through changing its Unicode from unshaped to shaped that provides similar letters shape to embed the secrete message. This proposed method doesn't add any more bytes to the cover text when embedding whereas most of the different strategies hide info through modification the size of cover text by adding Arabic kashida character, ZWJ/NZWJ Unicode, discretizes, little space, pseudo-space, and zero width space with traditional word space, and suspicions might be simply raised since it possible to notice the kashida or the discretize characters in kashida-based and discretize-based strategies, and therefore the hiding capability of these strategies are variable that depends on the flexibility of the Arabic letter to be extended with another character (kashida, and

discretize), and it's uncommon these days to send discretized text, whereas all of these different strategies might enlarge the covert text unexpectedly. The proposed method is strong against attack because of multi-layer and randomization are used throughout the embedding process, and also the secret message is inserted according to the detected POS in different word locations of the cover text, and a private secret key is used to prove the authenticity. As a future work, this technique may be used with other compression algorithms similar to the Huffman algorithm rather than using binary bits to represent the concealment message. The proposed technique may be also joined with other Arabic text steganography strategies found in the literature to get higher capacity.

REFERENCES

- [1] Morkel, T., J.H.P. Eloff and M.S. Olivier (2005) "An overview of image steganography", Proceedings of the Fifth Annual Information Security South Africa Conference: (ISSA2005).
- [2] Bennett, k., (2004) "Linguistic steganography: survey, analysis, and robustness concerns for hiding information in text", CERIAS Tech. The report, Purdue University, West Lafayette: IN 47907-2086.
- [3] Shirali-Shahreza, M., and M. Shirali-Shahreza (2006) "A New Approach to Persian/Arabic text steganography", 5th IEEE/ACIS International Conference on Computer and Information Science: pp. 310-315.
- [4] Odeh, A., Elleithy, K., & Faezipour, M. (2013) "Steganography in Arabic text using Kashida variation algorithm (KVA)", In Systems, Applications, and Technology Conference (LISAT), 2013 IEEE Long Island (pp. 1-6). IEEE.
- [5] Taha, A., Hammad, A. S., & Selim, M. M. (2018) "A high capacity algorithm for information hiding in Arabic text", Journal of King Saud University-Computer and Information Sciences.
- [6] Gutub, AAA, L. Ghouti, AA. Amin TM., Alkharobi, and MK. Ibrahim (2007) "Utilizing extension character Kashida with pointed letters for Arabic text digital watermarking". International Conference on Security and Cryptography - SECRYPT, Barcelona, Spain, July 28 – 31.
- [7] Malalla, A. P. D. S. (2016) "Improving Hiding Security of Arabic Text Steganography by Hybrid AES Cryptography and Text Steganography", International Journal of Engineering Research and Applications, 6(6), 60-69.
- [8] Al-Nofaie, S., Fattani, M., & Gutub, A. A. A. (2016) "Capacity Improved Arabic Text Steganography Technique Utilizing Kashida with Whitespaces", In The 3rd International Conference on Mathematical Sciences and Computer Engineering (ICMSCE2016) (pp. 38-44).
- [9] Ala'a, M., & Alnihoud, O. (2017) "AMeliorated KASHIDA-BASED APPROACH FOR ARABIC TEXT STEGANOGRAPHY", Int. J. Comput. Sci. Inf. Technol. (IJCSIT), 9(2).
- [10] Alotaibi, R. A., & Elrefaei, L. A. (2018) " Improved capacity Arabic text watermarking methods based on open word space", Journal of King Saud University-Computer and Information Sciences, 30(2), 236-248.
- [11] Aabed, M.A., Awaideh, S.M., Elshafei, A.R.M., Gutub, A.A. (2007) "Arabic diacritics based steganography", In ICSPC 2007. IEEE International Conference on Signal Processing and Communications, IEEE, pp. 756–759. Alginahi,
- [12] 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 (ICSPC 2007), November 24-27, Dubai, UAE: pp: 756-759.

- [13] Ahmadoh, E. M., & Gutub, A. A. A. (2015) "Utilization of two diacritics for Arabic text steganography to enhance performance", *Lecture Notes on Information Theory* Vol, 3(1).
- [14] Bensaad, M. L., & Yagoubi, M. B. (2011) "High capacity diacritics-based method for information hiding in Arabic text", In *Innovations in Information Technology (IIT)*, 2011 International Conference on (pp. 433-436). IEEE.
- [15] Kamaruddin, N. S., Kamsin, A., & Hakak, S. (2017) "Associated diacritical watermarking approach to protect sensitive Arabic digital texts", In *AIP Conference Proceedings* (Vol. 1891, No. 1, p. 020074). AIP Publishing.
- [16] Gutub, A.A., Y.S. Elarian and A.K. Alvi, (2008) "Arabic Text Steganography Using Multiple Diacritics". 5th IEEE International Workshop on Signal Processing and Its Applications (WoSPA 2008), 18-20, University of Sharjah, Sharjah, United Arab Emirates.
- [17] Odeh, A., & Elleithy, K. M. (2012) "Steganography in text by merge zwc and space character"., *CAINE-2012 25th International Conference on Computer Applications in Industry and Engineering*.
- [18] Aman, M., Khan, A., Ahmad, B., & Kouser, S. (2017) "A hybrid text steganography approach utilizing Unicode space characters and zero-width character", *International Journal on Information Technologies and Security*, 9(1), 85-100.
- [19] Reem A. Alotaibi, Lamiae A.Elrefai. (2016) "Utilizing Word Space with Pointed and Unpointed Letters for Arabic Text Watermarking", In *Proceedings of the IEEE 18th International Conference on Computer Modelling and Simulation (UKSimAMSS)*, Cambridge, UK, April 2016, 111–116.
- [20] Alotaibi, R. A., & Elrefaei, L. A. (2018) "Improved capacity Arabic text watermarking methods based on open word space", *Journal of King Saud University-Computer and Information Sciences*, 30(2), 236-248.
- [21] Stanford-postagger (2018) "<https://nlp.stanford.edu/software/tagger.html>", last visit, December, (2018).
- [22] Hardeniya, N., Perkins, J., Chopra, D., Joshi, N. and Mathur, I. (2016) "Natural Language Processing: Python and NLTK". Packt Publishing Ltd.
- [23] Jurafsky, D. and Martin, J.H., (2016) "Speech and language processing". London: Pearson.
- [24] Shirali-Shahreza, M.H. and M. Shirali-Shahreza (2008) "Steganography in Persian and Arabic Unicode texts using Pseudo-Space and Pseudo connection characters", *Journal of Theoretical and Applied Information Technology*.
- [25] Shaker, A. A., Ridzuan, F., & Pitchay, S. A. (2017) "Text Steganography using Extensions Kashida based on the Moon and Sun Letters Concept", *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 8(8), 286-290.
- [26] Saad, M. K. and Ashour, W. (2010) "OSAC: Open Source Arabic Corpus", *Proceedings of the 6th International Conference on Electrical and Computer Systems (EECS'10)*, Lefke, North Cyprus, pp. 1-6.
- [27] Alginahi, Y. M., Kabir, M. N., & Tayan, O. (2013) "An enhanced Kashida-based watermarking approach for Arabic text-documents", In *Electronics, Computer and Computation (ICECCO)*, 2013 International Conference on (pp. 301-304). IEEE.

- [28] Taleby Ahvanooy, M., Li, Q., Shim, H.J. and Huang, Y. (2018) "A Comparative Analysis of Information Hiding Techniques for Copyright Protection of Text Documents". Security and Communication Networks.