Contents lists available at Crown Academic Publishing

International Journal of Engineering and Artificial Intelligence
Journal home page: http://wwwijeair.com

Data Hiding Enhancement in Arabic Text

Abdulaziz Saleh Al-Dini [1], Naziha Mohammed Al-Aidroos [2], Khalid Q. Sha'Afal [3]

[1] Information Technology Department, Faculty of Engineering & Information Technology, Al-Rayan University, Hadhramout, Yemen

[2] Computer Science Department, College of Computers & Information Technology, Hadhramout University, Hadhramout, Yemen

[3] Faculty of Education, Aden University, Aden, Yemen

Corresponding Author: aziz2014saleh@gmail.com

Original article Received 28 October 2020, Accepted 20 November 2020, Available online 30 January 2021

ABSTRACT

Recently, information security has become a very important topic for researchers as well as military and government officials. For secure communication, it is necessary to develop novel ways to hide information, for this purpose, steganography is usually used to send secret information to their destination using different techniques. The aim of this article is to provide a new text steganography method. Hidden information in text files is difficult to discover as text data has low redundancy in comparison to other mediums of steganography. Hence, we use an Arabic text to hide secret information using a combination of Unicode character's zero-width-character (ZWC), zero-width-joiner (ZWJ) and pseudo-space (PS) in the proposed algorithm. The experimental results show increasing in hidden data capacity per word in comparison to the recently proposed algorithms. The major advantage of our proposed algorithm over previous researches is the high visual similarity in both cover and stego-text that can reduce the attention of intruders.

Keywords: Information Hiding; Text Steganography; Capacity; Arabic Text; Zero-Width-Character (ZWC); Zero-Width-Joiner (ZWJ); Pseudo-Space (PS).

1. Introduction

The use of information technology is evolving, communication is increasing and the exchange of information requires that communication be more secure. One of the approaches to protect information when going over an untrusted channel is steganography (Ahmed Taha et al., 2018). It is defined as "the art and science of writing a hidden message in such a way that no one, apart from the sender and intended recipient, even realize there is a hidden message" (Al-Otaibi and Gutub, 2014). Steganography uses some safe carrier called stego media in which the message is embedded. It may use image, audio, video, or text to hide the existence of the message.

Of all carrier media, images are the most widely used carrier to hide data. However, the text has some advantages that make it comparable to images (Gutub and Fattani. 2007). It has less memory usage; it is faster than other methods and it is easier in communication (Chaudhary and Dave, 2016). :

Steganography mainly employs redundant data in the carrier media to hide the secret message. In most cases, text steganography is more challenging due to the lack of redundant data in text files (Shirali-Shahreza et al., 2008). In

addition, the structure of text documents is almost identical to their looks so any change may be visible. In general, text steganography depends mainly on the language characteristics.

It is known that the grammatical and orthographic characteristics differ from one language to another. Here, we have used an Arabic text as a cover media to hide secret information by applying our proposed algorithm as shown in **Figure 1** and produce a stego-object without change the visual appearance of the cover text. After embedding secret data into cover text, we have obtained a stego-text file (stego-object).

The stego-object file is transmitted over the internet from the sender to the receiver through several steps **Figure 1**.

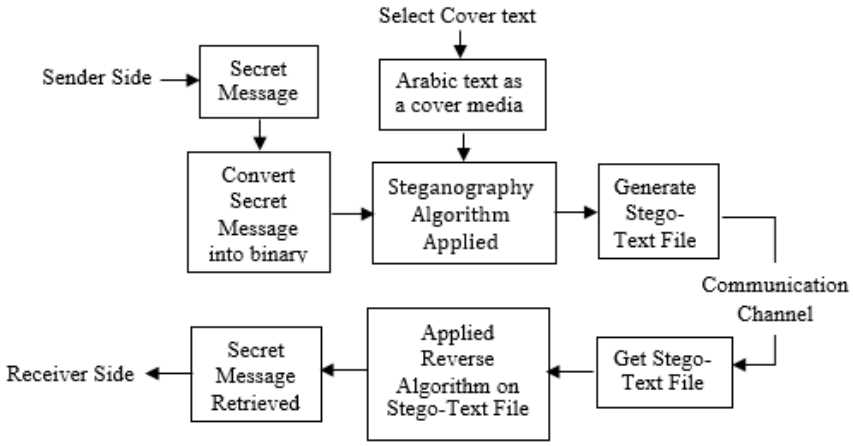

Figure 1. Text based steganography– a basic mechanism

In the proposed algorithm, we ensure steganography parameters like capacity, robustness successfully. This algorithm not only improves data security but also enhances data hiding capacity up to 1.8 times, which is higher than any other previously used text-based steganography technique and generates stego-text file as the same visual appearance of cover text that will reduce the attention of hackers and intruders.

2. Background

Generally, text steganography is the most challenging kind of steganography techniques, due to the relative lack of redundant information in text files as compared to images or audio. Recently, there have been several successful attempts to design text steganography schemes for many languages including English, Chinese, Hindi, Malayalam, Bengali and Urdu. However, Arabic is not as fortunate. In Arabic script, several researches have been done, e.g. dot steganography (Bensaad and Yagoubi, 2011), pointed letters with extension (Kashida in Arabic) (Gutub et al., 2007; Memon et al., 2008; Odeh and Elleithy, 2013; Odeh et al., 2013), fully vocalized Arabic texts (i.e. Arabic texts with diacritics) (Nofaie et al., 2016), steganography and pseudo space (Ahmadoh and Gutub, 2015), and pseudo connection (Shirali-Shahreza et al., 2008).

This section reviews some related work presented for Arabic text steganography.

2.1 Point Shifting Steganography

Third-order Arabic language has 28 letters. Authors categorized these letters into two groups. Pointed like 'ا' Alif; 'ر' Raa; 'و' Waaw, and un-pointed letters like 'ت' taa; 'ذ' Thaal; 'ز' Zaa respectively. Out of 28 letters, 14 are pointed letters. Authors used these pointed letters to hide secret data. By shifting points upwards to hide secret bit 1 and keep it normal distance to hide bit zero (Shirali-Shahreza, 2006). Drawback of this method is hidden information can be lost by using optical character recognition (OCR) software or retyping these documents.

2.2 Extension and pointed letters steganography

Out of 28 letters, 22 Arabic letters can be connected to next or previous letter of the word by Arabic extension. Gutub and Fattani (2007) used this extension with pointed letters to hide secret bit 1 and un-pointed letters to hide bit 0 as shown in Figure 2. Authors (Al-Nofaie et al., 2016) improved extension/Kashida approach by utilizing 'Kashida' with whitespaces between the words to enhance the hidden data capacity of cover text. After using extension in some words, length and size of cover text file increased **Figure 2** which becomes suspicious and gets attention of intruders.

Secret bit	110010
Original text	من حسن اسلام المرء تركه مالا يعنيه
Output text	من حسن اسلام المرء تركه مالا يعنيه 0 1 0 0 1 1

Figure 2. Data hiding with pointed and un-pointed letters by inserting Arabic extension

2.3 Using Pseudo-Space and Pseudo Connection Steganography

Shirali et al (2008) proposed a new approach of Arabic Unicode texts to hide data in Persian and Arabic languages by using pseudo-space and pseudo-connection also known as zero-width non-joiner (ZWNJ) and zero-width-joiner (ZWJ), respectively. This approach sorted out weather the letter of word is connected to next letter or not. If letter is connected to the next letter, it hided bit one by inserting ZWJ letter between two letters. However, if letter is not connected to the next letter, then it inserted ZWNJ between letters and do not insert any character to hide bit zero. This proposed method is only useful for non-diacritics Arabic text.

2.4 Using Arabic Diacritic Steganography

Arabic language mainly used eight diacritics like Fatha, Kasrah, Dhammah, etc. Diacritics are used on Arabic text to perform correct pronunciation of Arabic language. To embed secret information (Aabed et al., 2007) used fully diacritised Arabic text as a cover media. To hide bit 1 they found first Fatha in the Arabic text and hide on that letter. But they removed all other diacritics like Kasrah, Dhammah, etc., from cover text if appeared before first Fatha. However, bit 0 embedded in all non-Fatha diacritics. To increase the data hiding capacity (Ahmadoh and Gutub, 2015) improve the work of (Aabed et al., 2007) and proposed a new data hiding approach by using two diacritics 'Fateh' and 'Kasrah' on the basis of even and odd bits of the secret message. A lot of efforts have been done to hide secret data by using more than one Arabic diacritics like Fatha, reverse Fatha or diacritics shift (vertically upward) (Odeh and Elleithy, 2012). However, recently these methods are facing different challenges. Firstly, according to recent trends Arabic text is used without diacritics. Secondly, after removing some diacritics or its shift, file size is changed and it becomes suspicious.

2.5 Unicode Approach Steganography

In Arabic language, 22 letters can be connected to next or previous letter of the word. Therefore, one letter can be written in four different shapes as shown in **Table 1**. All of these connected shapes of the letter can be first, middle, last or an isolated format in the word. Each specific shape of letter has its unique Unicode. Shirali et al (2008) used these kinds of Arabic Unicode to hide secret information into Arabic cover text. This approach is suitable if the text appears in non-diacritics form. Unicode approach has also been used (Mohamed, 2014) to embed secret data 0 and 1 in run length encoding format. However, this approach is restricted to hide data using only in isolated letters.

Table 1. Letter shapes according to its position in the word

At the end of the word	In the middle of the word	At initial form of the word	Isolated form in the word
ـس	ـسـ	سـ	س

2.6 Using Kashida and Pseudo-Space

Kashida and pseudo-space (PS), extends (Al-Alwani, et al., 2007), which used only Kashida, and in (Al-Nofaie et al. 2016), which used Kashida with normal spaces to hide secret bits. Authors in (Al-Nofaie et al. 2019) used Kashida-PS, This approach check the bit, and letter of word is connected to next letter or not, if the bit is 1 and the letter is connected to the next letter, it hided bit 1 by inserting two kashida between two letters. However, if letter is not connected to the next letter, then it inserted two PS between letters. Otherwise, if the bit is 0 and the letter is connected to the next letter, it hided bit 0 by inserting one kashida between two letters. However, if letter is not connected to the

next letter, then it inserted one PS between letters, length and size of cover text file increased **Figure 3** which becomes suspicious and gets attention of intruders.

Secret bit	100101100101110011100011
Original text	احرص على ماينفعك واستعن بالله ولاتعجز
Output text	احرص علـى ماينفـعك واسـتـعن بـالله ولاتعجز
	100101100101110011100011

Figure 3. Data hiding with pointed and un-pointed letters by inserting Arabic extension

3. The Proposed Algorithm

3.1 Methodology

All private letters out of 28 Arabic letters are classified as special letters. These special letters are 'ا' Alif, 'د' Daal, 'ذ' Thaal, 'ر' Raa, 'ز' Zaa, and 'و' Waaw. These special letters be private in Arabic text in the following cases: if show at start of word or at the end of the word. These special letters cannot be connected with letter next to it if appeared in the center of the word. This special feature of these letters can be benefit as word divider. For example, in 'المرسلات' we can see that Arabic word divides into parts because:

'ا' Alif appear at start

'ر' Raa belongs to six defined letters so we can see that it is not combined to the next letter.

'ت' Taa appears as an isolated letter as before this 'ا' Alif belongs to six defined letters and it is second last letter of the word so the last letter will be isolated letter (Mohamed, 2014).

We used this performance of these special letters in our algorithm. Since, our algorithm based on the combinations of zero-width-character (ZWC), ZWJ and PS, therefore it consists of three different scenarios as follows:

- If the letters belong to special letters we insert combination of ZWC.
- If letters are combined to next letter or isolated letter, then we insert combination of ZWJ character between the letters to hide pair of secret message bits.
- If there are spaces between words, then we insert PS character between the words to hide one bit of secret message.

While pseudo-space (PS) is a character that does not appear in printing.

These combinations of ZWC, ZWJ and PSs increases the capacity of hidden information. This is because every character (letters and spaces) is exploited to hide a secret. Unlike the previous methods mentioned above. These combinations are made by deep analysis of Arabic text. As a result, after embedding secret message bits, shape and visibility of cover-text and stego-text remain same with higher capacity of hidden data. The other distinction of our proposed algorithm is that we used it on both diacritic and non-diacritics Arabic text.

Generated binary bits of secret text are calculated as:

$$Bsm = Tc \times 8 \quad (1)$$

Where Bsm = total no of bits to secret message, Tc refers to total no. of characters and 8 corresponds to every character's length.

3.2 Inserting Start-End Bits into Cover Text

How can we keep more secure our hided data in cover text from the hackers? For this, we embed some bits to keep track of start and end point. If we find start-end bits in text, it ensures the existence of secret message in text file. To make stego-text file more secure we further embed start-end point at different places more than one time in cover text file with the combination of ZWC and ZWJ with PS. The combination of ZWC and ZWJ with PS increase the security and confidentiality because it is hard to judge for hackers about start-end points due to similar combinations of ZWC and ZWJ with PS for hiding secret message. It also improves the data security.

3.3 How To Embed Secret Bits Into Cover Text?

To embed secret bits into cover text, first our algorithm read pair of bits of Bsm and then read next letter of cover text. If next letter is from special letters, then it applies zero width character's (ZWC) combination after special

letter. Otherwise, it applies ZWJ combination before next letter, if there is space between words will insert pseudo-space (PS). No action is taken if next letter is the last letter of the word, which indicates no data is hidden in the last letter of each word. Hided data's capacity in each word is calculated by using this equation $[(NL_w) - 1] \times 2$ (2)

Where NL represents number of letters in each word of cover text and w is word count range from 1 to n. For example, in 'المرسلات', this word contains eight letters, so its capacity will be: [8 – 1] × 2 = 14. It means an eight-letter word can hide 14 bits of secret message. Therefore, for n number of words:

Hided capacity of bits $= \left(\sum_{w=1}^{n}[(NL_w - 1) * 2]\right) + NS$ (3)

Where NS represents number of spaces in the cover text.

3.4 The Proposed Scheme

The proposed scheme is comprised of two phases, as follows:

A. The Embedding Phase:

The embedding phase is based on checking the cover text capacity, if it has capacity large than number of bits, it will insert start bits then read next character if it is space or not, if it is space it will check next bit and insert PS. If the character not space it will check if the special letter it will take next bit pair and apply ZWC, and if next letter is not special letter it will take next bit pair and apply ZWJ. Then it will check the all bits has hidden or else return previous steps until hidden all bits. The embedding phase algorithm is shown in **Algorithm 1**.

Algorithm 1. The Embedding Phase Algorithm

Input : Secret Message bits (M), Cover Text File (C)
Output : Stego_text(St)

1. Calculate the size of C according to equation (3).
2. Set i = 0 // Counter to hold the current secret bit being processed
3. Set j = 0 // Counter to hold the current character in the cover text file being processed
4. Set k = 0 // Counter to hold the current character in the Stego_text
5. Insert start bits.
6. For each bit (b_i) in the secret bit sequence
   If the current character in the cover file (C_j) is space
   then
      If $b_i = 1$
         Add two pseudo-space to Stego_text
      else // $b_i = 0$
         Add one pseudo-space to Stego_text
      i++; j++; k++.
   else if the current character in the cover file (C_j) is special letter
      generate ZWC(b_i , b_{i+1})
      i = i+2
      j++
      k++
   else // current character in the cover file (C_j) is not special letter
      generate ZWJ(b_i , b_{i+1})
      i = i+2
      j++
      k++
7. Repeat the step 6 until all M's bits are embedded.
8. Insert end bits for stop hidden secret message process
9. Generate the Stego_text St

B. The Extracting Phase:

The extraction algorithm of the proposed method is shows in **Algorithm 2.** The algorithm takes stego-text as an input and returns the secret message as an output.

The stego-text is sequentially processed as follows: for each character in a word, if it is space, it will check is their one pseudo-space will return bit 0, or bit 1 if their two pseudo-space, then check the next character if it special letter it will return bit pair of ZWC, or bit pair of ZWJ if the letter is not from special letters.

Algorithm 2. The Extracting Phase Algorithm

Input : Stego_text(St)
Output : Secret Message bits (M)

1. Set k = 0 // Counter to hold the current character in the Stego_text

2. Insert start bits.
3. For each character (C_k) in the Stego_text
   If (C_k) is space
      search for the number of pseudo-space
      and extract the corresponding secret bit
      add extract bit to (M)
      k++
   else if (C_k) is special letter
      use map-zwc combination
      and extract the corresponding secret bits
      add extract bits to (M)
      k++
   else // (C_k) is not special letter
      use map-zwc combination
      and extract the corresponding secret bits
      add extract bits to (M)
      k++
4. Return Secret Message bits (M)

3.5 Procedure of Hiding Secret Data into Cover Text

To hide secret data, we made an algorithm that embed secret data into cover text and generate stego-text file by using proposed combinations of ZWC, ZWJ and PS as shown in **Figure 4.**

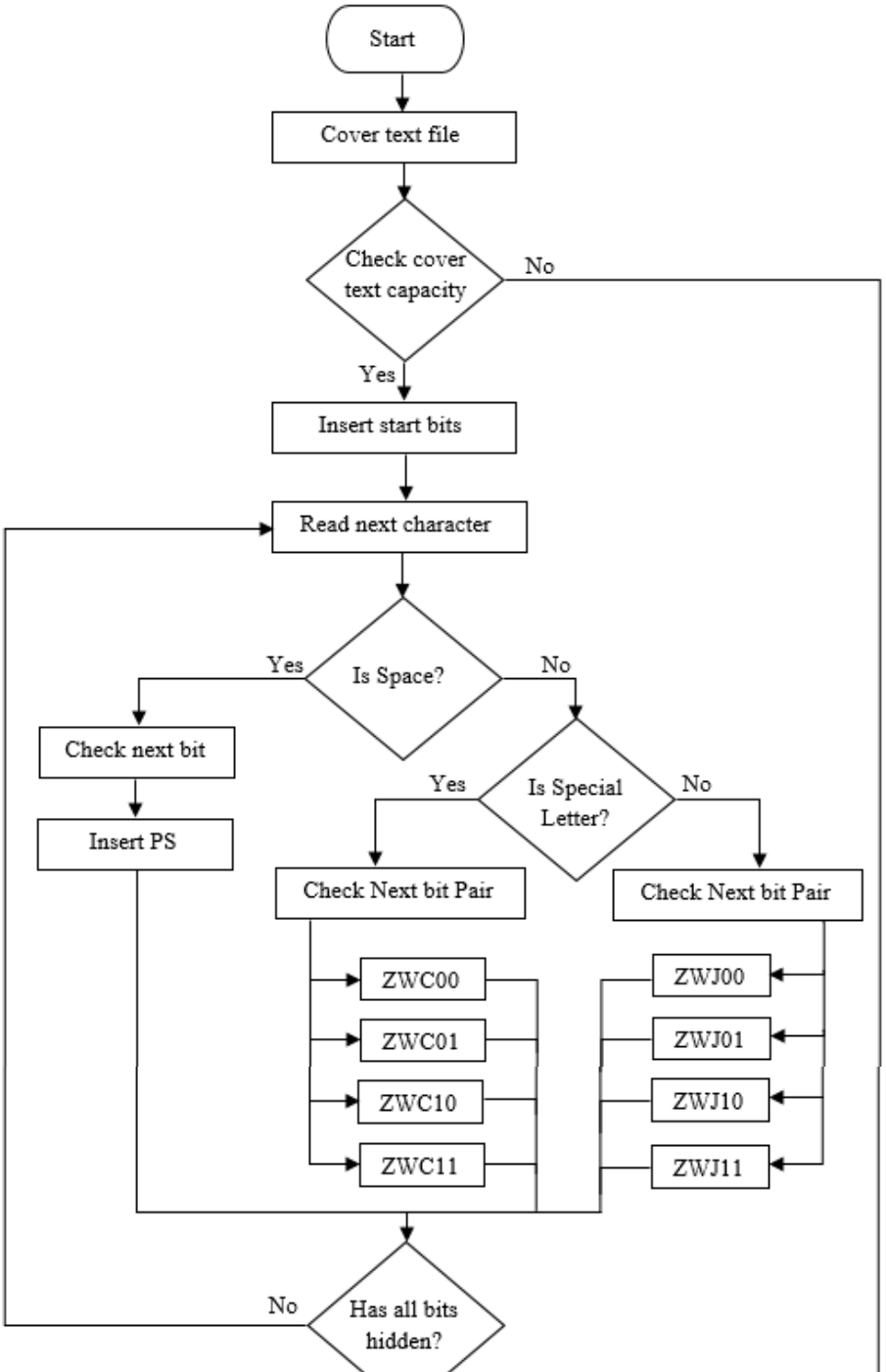

Figure 4. Procedural flow chart for embedding secret bits into Arabic cover text file

As example to understand the proposed algorithm, we selected this Arabic sentence 'انا عند ظن عبدي بي' as cover text and '100110111' (9 bits) data bits to hide. Following steps have been taken to hide above-mentioned data bits into cover text:

Step 1 At first, algorithm verifies cover text capacity. It will only hide data if capacity is more than bits count. As we can see, our selected Arabic sentence consists of five words. Each word has 3, 3, 2, 4, and 2 letters and 4 spaces, respectively. Therefore, according to equation (3), we can hide up to 22 bits in this selected Arabic verse as shown below:

[(3 - 1) * 2 + (3 - 1) * 2 + (2 - 1) * 2 + (4 - 1) * 2 + (2 - 1) * 2] + 4 = 22

It shows, our selected sentence/string has enough capacity to hide data. As our selected data ('100110111') which is being hided has only 9 bits, so it can be easily hided in the selected string. For this, we inserted some starting bits at the start of cover text and proceeded.

Step 2 At the next stage, algorithm read next letter of selected string. If next letter of selected string is among special defined letters then it inserts ZWC combination after received letter, otherwise it will insert ZWJ combination before received letter according to bit pair. Since it received 'ا' as next letter which is from the special defined six letters and the first bit pair is '10'. Therefore, our algorithm will add ZWC10 + 'ا'.

Step 3 Next letter of selected string is 'ن' and '01' is the next hidden bit pair. Which is 'ن' does not belong to the special defined letters. Therefore, our algorithm inserts ZWJ01 before the letter 'ن' as ZWJ01 + 'ن'.

Step 4 Next is 'ا' which is last letter of the word. As we defined in Section 3.3, our algorithm will skip it and move to next character.

Step 5 Next character is space. Our algorithm will insert two PSs, because the next bit is 1. Next hidden bit is '1'. Therefore, it adds two PSs in selected string.

Step 6 Next letter is 'ع' which is not from special letter. As defined in Section 3.3, our algorithm will insert combination of ZWJ. Next hidden bit pair is '01'. Therefore, it adds 'ع' + ZWJ01 in selected string.

Step 7 Next one is non special letter 'ن'. Therefore, cover text will be ZWJ11 + 'ن'.

Step 8 Next, algorithm did not find any bit pair in the selected data, which confirm whole data is embedded successfully in the cover text. After this, algorithm will insert end bits in the cover text to indicate hidden process is completed. In the end, it will add random ZWC and ZWJ combinations into the remaining cover text to create confusing situation for intruders in steg-analysis and message regeneration.

Step 9 At the end, we have a stego-text file, which contains completely hidden data. In stego-text file, visual appearance of text is exactly the same as cover text file which is the key strength of our proposed algorithm.

3.6 Extracting Algorithm to Retrieve Secret Message from Stego-Text

After creating stego-text file, it is important to know how to extract hidden message. For this, we adopt reverse procedure of embedding algorithm as shown in **Figure 5.**

Step 1 Firstly, algorithm will verify the existence of start/end bits. If start and end bits were found, then it starts regenerating secret message other it stops. From start bit it read next letter of the selected string.

Step 2 Next letter of stego-text is 'ا'and it is among the special defined. In this case, there will be a combination of ZWC after letter 'ا'as stated in Section 3.3. As a result, our algorithm will extract ZWC10 information and add '10' bit pair in the regenerated message string, i.e., 10.

Step 3 Next one is letter 'ن' which is connected to next letter. For this, it will extract a combination of ZWJ01 and add '01' bit pair in the regenerated message string, i.e., 1001.

Step 4 Next one is 'ا'and it is last letter of the word. For this, no action will be taken as stated earlier.

Step 5 Next character is space. For this algorithm will extract 1 bit and add 1 in regenerating string, i.e., 10011.

Step 6 Next word starts with letter 'ع'which is non-special letter. For this algorithm will extract ZWJ01 and add 01 in regenerating string, i.e., 1001101.

Step 7 Next one is a non-special letter 'ن'. For this, it will extract ZWJ11 and add 11 with regenerating string, i.e., 100110111.

Step 8 For the next letter'ـد' we find end bits instead of data bits which confirm the extraction of hided data. After this algorithm will stop further extraction process.

Step 9 Finally, we converted regenerated binary string into humane readable format. That is our secret message.

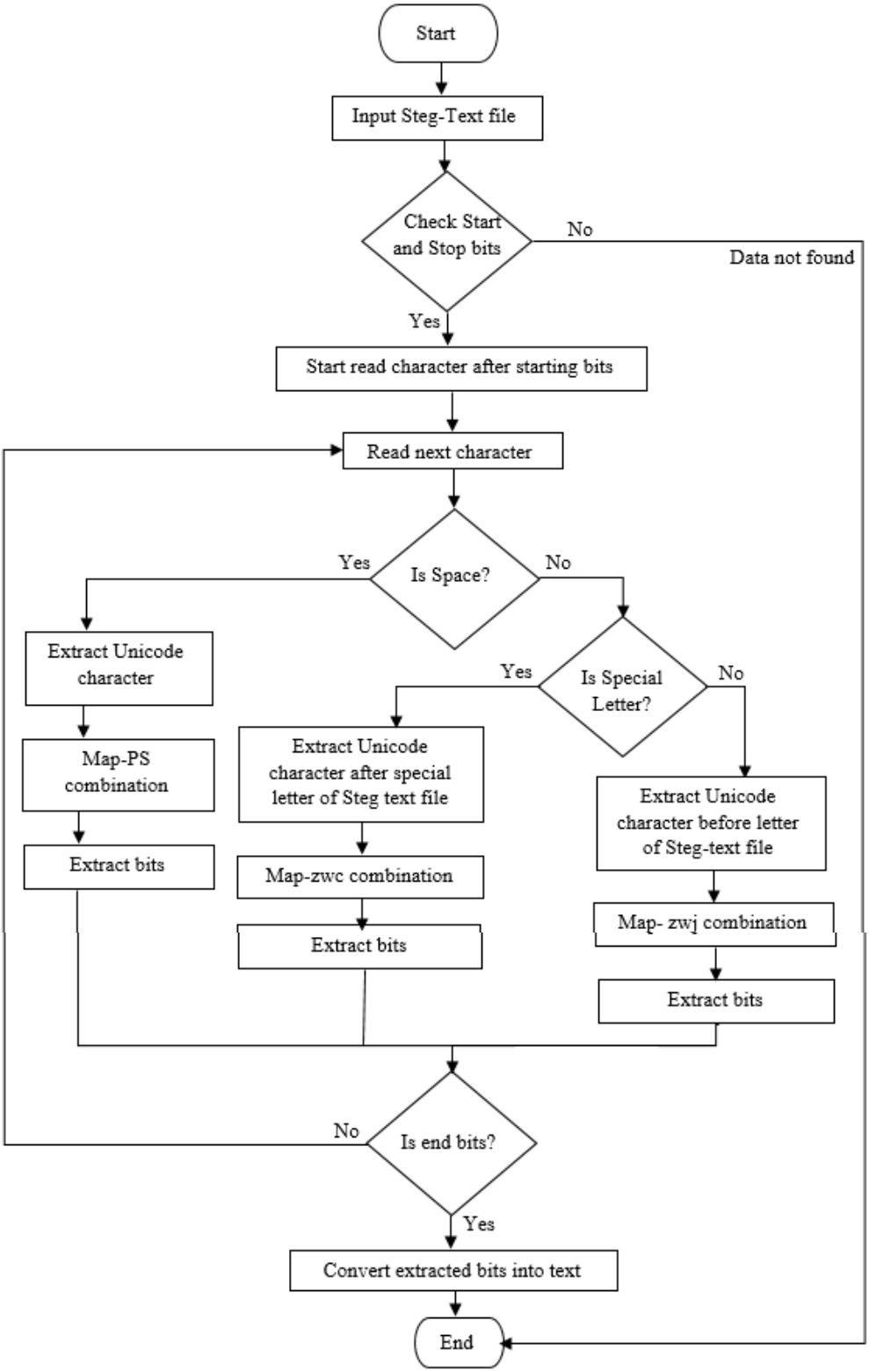

Figure 5. Reverse algorithm flow chart for extracting secret message from stego-text file

4. Results and Discussions

In this section, we described our experiments on Arabic text for data hiding process and their results. We applied our algorithm not only on diacritic but also on non-diacritic cover text strings for hiding secret data as shown in Table 2. As we can see clearly from Figure 5, data hidden capacity is higher than any other used approaches like inserting Kashida after pointed letters, Kashida variation algorithm, Unicode approach to hide information using isolated letters, Kashida-PS, etc., (Aabed et al., 2007; Gutub and Fattani, 2007; Mohamed, 2014; Odeh et al., 2013; Al-Nofaie et al., 2019). In all these mentioned proposed techniques, authors used minimum letter of each Arabic word to hide secret bit 1 or 0. Therefore, hidden data capacity per word very low as compared to our proposed approach. We used all letters except the last letter of the word to embed pair of bits in each letter of the cover text file. From experimental results of **Table 2**, we can see that against all recently data hiding proposed approaches, visual similarity of cover text and stego-text is compromised due to inserting Kashida before/after pointed letters or removing diacritics from the letter to hide secret data. While our proposed algorithm gives, better results and achieved same visual appearance of cover-text and stego-text file with significantly increased hidden data capacity per word.

Table 2. Results of data hiding capacity on different cover text using our algorithm and comparison with other used approaches

Input/output	*Selected string as a cover object and stego-object*	*No. of hidden bits capacity*	*Approaches*	*References*
Cover object	من حسن السلام المرء تركه مالا يعنيه	6 bits	Kashida approach	Gutub and Fattani (2007)
Secret information	110010			
Stego-object	يـعنـيه مـن حسن السلام المرء تـركـه مـالا			
Cover object	من حسن السلام المرء تركه مالا يعنيه	50 bits	Combinations of ZWJ, ZWC and PS	Proposed scheme
Secret information	01001000011001010110110110001101100011011110010100			
Stego-object	من حسن السلام المرء تركه مالا يعنيه			
Cover object	حَدَّثنا سُفيَانُ عَنْ يَحْيَى	8 bits	Diacritics approach	Aabed et al. (2007)
Secret information	11100111			
Stego-object	حَدَثنا سُفيَان عَن يَحْيى			
Cover object	حَدَّثنا سُفيَانُ عَنْ يَحْيَى	27 bits	Combinations of ZWJ, ZWC and PS	Proposed scheme
Secret information	0111100101100 10101110011010			
Stego-object	حَدَّثنا سُفيَانُ عَنْ يَحْيَى			
Cover object	اتق شر من احسنت اليه	9 bits	Kashida variation algorithm	Odeh et al. (2013)
Secret information	001011100			
Stego-object	اتـق شـر من احـسنت الـيـه			
Cover object	اتق شر من احسنت اليه	26 bits	Combinations of ZWJ, ZWC and PS	Proposed scheme
Secret information	01111001011001010111001011			
Stego-object	اتق شر من احسنت اليه			
Cover object	صلاح أمرك للأخلاق مرجعه فقوم النفس بالأخلاق تستق	5 bits	Unicode approach to	Mohamed (2014)

Secret information	11010		hide data in isolated letters	
Stego-object	صلاح أمرك للأخلاق مرجعه فقوم النفس بالأخلاق تستق			

Input/output	*Selected string as a cover object and stego-object*	*No. of hidden bits capacity*	*Approaches*	*References*
Cover object	صلاح أمرك للأخلاق مرجعه فقوم النفس بالأخلاق تستق	75 bits	Combinations of ZWJ, ZWC and PS	Proposed scheme
Secret information	011010010111010000100111 01110011001011 100000011011110110101101 1000010111001			
Stego-object	صلاح أمرك للأخلاق مرجعه فقوم النفس بالأخلاق تستق			
Cover object	احرص على ماينفعك واستعن بالله ولاتعجز	25 bits	Combinations of Kashida and PS	Al-Nofaie et al. (2019)
Secret information	10010110010111001111000011			
Stego-object	احـرص عـلـى ماينفـعـك واسـتـعـن بـالله ولاتـعجز			
Cover object	احرص على ماينفعك واستعن بالله ولاتعجز	57 bits	Combinations of ZWJ, ZWC and PS	Proposed scheme
Secret information	010010000110010101110110010111 000110110001101111100101101000			
Stego-object	احرص على ماينفعك واستعن بالله ولاتعجز			

5. Conclusion

We used Arabic text as a cover media to hide secret information by using combinations of ZWC and ZWJ with PS in our proposed algorithm. Combinations of ZWC and ZWJ with PS were made by deep analysis of Arabic text. To calculate data hiding capacity for each word, we formulated this equation $[(NL_w) - 1] \times 2$ Addition to number of spaces (NS). Our experimental results show that with the help of our algorithm we can hide higher capacity of data into the cover text than any other reported algorithms. We improved the security by placing start bits, end bits and random combinations of ZWC and ZWJ with PS into the cover text to create confusing situation for intruders. For the first time, we achieved same visual appearance of cover-text and stego-text file that will reduce the attention of hackers and intruders. As a future work, we will increase the data capacity ratio by using all letters of the word. In addition, we will use other features of Arabic language to enhance the hiding capacity. In addition, we have to improve more security phase by using some compression tools on data to reduce the size of text file.

6. References

A. Taha, A. S. Hammad and M. M. Selim, (2018). A high capacity algorithm for information hiding in Arabic text," Journal of King Saud University – Computer and Information Sciences, p. 8.

R. Alotaibi and L. Elrefai, (2017). Improved capacity Arabic text watermarking method based on open word space," ScienceDirect, pp. 50-62.

A. A.-A. Gutub and M. M. Fattani, (2007). A Novel Arabic Text Steganography Method using Letter points and Extension," in Proceedings of the WASTET International Conference on Computer, Information and System Science and Engineering (ICCISSE), Vienna, Austria.

S. Chaudhary, M. Dave, A. Sanghi and J. Manocha, (2016). An Elucidation on Steganography and Cryptography," in Proceedings of the Second International ACM Conference on Information and Communication Technology for Competitive Strategies (ICTCS '16), Udaipur, India, 43.

Shirali-Shahreza, Mohammad and Sajad, (2008). Steganography in TeX Document," in Proceedings of 3rd IEEE International Conference on Intelligent System and Knowledge Engineering (ISKE 2008), University Convention Center Xiamen, China.

M. L. Bensaad and M. B. Yagoubi, (2011). High Capacity Diacritics-based Method for Information Hiding in Arabic Text," in Proceedings of IEEE International Conference on Innovations in Information Technology (IIT), Abu Dhabi, UAE.

Memon, G. Ahmed, Kowhai, Karman, Kazi and Hameedullah, (2008). Evaluation of steganography for Urdu/Arabic text," J. Theor. Appl. Inf. Technol. (JATIT), p. 4.

A. Odeh and K. Elleithy, (2013). Steganography in Text by Merge ZWC and Space Character," in Proceedings of the 28th International Conference on Computers and Their Applications (CATA-2013), Honolulu, Hawaii, USA.

A. Odeh, K. Elleithy and M. Faezipour, (2013). Steganography in Arabic Text Using Kashida Variation Algorithm (KVA)," in Proceedings of the IEEE Conference on Systems, Applications and Technology (LISAT), Long Island, USA.

Nofaie, S. Meteb, Gutub and Adnan, (2016). Merging two steganography techniques," J. Comput. Sci. Comput. Math. (JCSCM), p. 6.

E. M. Ahmadoh and A. A.-A. Gutub, (2015). Utilization of Two Diacritics for Arabic Text Steganography to Enhance Performance," Lecture Notes on Information Theory, pp. 42-47.

Shirali-Shahreza, (2006). A new approach to Persian/Arabic text steganography," in 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06), IEEE.

M. A. Aabed, S. Awaideh, M. Elshafei and A. A. Gutub, (2007). Arabic diacritics based steganography," in IEEE International Conference on Signal Processing and Communications, ICSPC.

A. Odeh and K. Elleithy, (2012). Steganography in Arabic text using zero width and Kashidha letters," International Journal of Computer Science & Information Technology, p. 1.

A. Mohamed, (2014). An improved algorithm for information hiding based on features of Arabic text: a Unicode approach," Egyptian Informatics Journal, p. 79–87.

W. Al-Alwani, A. Bin-Mahfooz and A. Gutub, (2007). "A novel arabic text steganography method using extensions," World Acad. Sci. Eng. Technol, pp. 83-86.

S. Al-Nofaie, A. Gutub and M. Al-Ghamdi, (2019). "Enhancing Arabic text steganography for personal usage utilizing pseudo-spaces," Journal of King Saud University – Computer and Information Sciences, p. 13.