Improving Security and Capacity for Arabic Text Steganography Using 'Kashida' Extensions

Fahd Al-Haidari Adnan Gutub Khalid Al-Kahsah Jameel Hamodi

Collage of Computer Sciences & Engineering
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia
fahdhyd@kfupm.edu.sa, gutub@kfupm.edu.sa

Abstract— **Steganography is a method of hiding data within a cover media so that other individuals fail to realize their existence. In this paper, a new approach for steganography in Arabic texts is proposed. The main idea is that each Arabic word may have some characters which can be extended by 'Kashida'. The ranks 'locations' of such characters and the inserted Kashida, construct a coding method to represent a block of secret bits. Different scenarios have been proposed based on the maximum number of Kashida possible to be inserted per word. The approach was compared to some existing Arabic text steganography approaches in terms of capacity and security. It is shown that this proposed approach outperforms the others with interesting promising results.**

Keywords: Arabic text, Cryptography, Feature coding, Information security, Steganography, Text watermarking.

I. INTRODUCTION

With the tremendous development of Internet, large bulk of text data are transmitted and exchanged daily. The security subjects of such transmitted information on the internet have became more attracted, among these topics, nowadays the text steganography is a hotspot in the research area of information security [1, 2, 3, 4, 5]. Steganography is the process of hiding data inside other cover data in such a way that no one apart from the intended recipient knows its existence. It replaces unneeded bits in image, sound, and text files with secret data [12]. Applications of steganography include covert communications; watermarking and fingerprinting that seem to hold promise for copyright protection and tracing source of illegal copies [7].

There are three main issues to be considered when studying steganographic systems: capacity, security and robustness [3]. The capacity refers to the ability of a cover media to store secret date, and it can be measured by the amount of secret data (bytes) that can be hidden in a byte of a cover media. The security refers to the ability of an eavesdropper to figure, or suspect the hidden information easily. The robustness refers to the ability of protecting the unseen data from corruption especially when transmitted through the internet.

Text steganography is considered tricky [2] as it solely deals with text, which has less redundant information and is clear in its structure. These two features dramatically affect both the capacity and security [1].

Many stego approaches have been proposed to hide secret data in cover text. Most of them hide data by making minimal modifications to the painting of the characters or spaces. In [3], they proposed using the Kashida after a pointed letter to hide a secret bit of '1' and un-pointed letter to hide a secret bit of '0'. This approach introduced a new idea in the text steganography also for other languages using the Kashida as an extension symbol similar to Arabic as text formatting character. However, it has drawbacks in terms of capacity and security. The authors in [4] named feature coding, open spaces, word shifting and line shifting, as examples for these stego approaches. Although, such methods offered high capacity and more security, they suffered from low robustness, because their font is entirely computer dependent. In [5], authors proposed a new approach using multiple diacritics to hide data within Arabic text assuming several scenarios.

In this paper, we address these limitations found in [3] and introduce a modified approach using Kashida to hide information in Arabic text. We, in Section 2, introduce the background information about standard Arabic language and its related work. In Section 3, our proposed approach is introduced in details. The analysis and discussion are reported in Section 4 followed by the conclusion in section 5.

II. BACKGROUND

Arabic language has some unique features different than other languages including English. Arabic character has different shape depending on its position within the word. In addition, it has some pointed (doted) letters, with one, two, or three dots on top or bottom of some characters. Also, Arabic characters have some extra shape located top or bottom the shape of the character called "Diacritics" representing the vowel sounds. Furthermore, Arabic has an extra character called 'Kashida' used for text formatting that extends some characters with no change in the meaning. Strangely, with all these features, not much research

work is found available to utilize such capabilities for data security purposes [11,1], which made our interest to this work.

In [1], the doted letters is used to hide bits. They slightly shifted the dots up more than normal to represent the hidden bit '1' and kept the pointed character normal (without this extra shift up) if they hide '0'. In this method, capacity can be very high as each pointed characters can hide a bit of either '0' or '1'. On the other hand, robustness is weak. For example, retyping or scanning the text looses the hidden information. Also, this approach depends on using same fixed font, where using different font produce unknown letters.

In [3], the authors modified the approach in [1] by proposing to hide information in Arabic text utilizing the extension letter 'Kashida'. They proposed using 'Kashida' before or after pointed letters to hold the secret bit '1' and Kashida before or after un-pointed letters to hold '0'. This approach solved the problem of the negative robustness form the previous work as it is font independent. It utilized Kashida which doesn't have any affect in the Arabic text content. However, the number of hidden bits reduced since their condition restricted the amount of inserted hidden bits '1' and '0' by the pointed and un-pointed letters respectively. Furthermore; security of such approach is less since a word may have many inserted Kashidas representing secret bits and reflecting existence of hidden information in the text cover media.

Diacritics - or Harakat – are proposed to be used for Arabic text steganography in [4]. The diacritics are used to represent the hidden information. Here, the diacritic 'Fatha' is used to hide the secret bit '1' and the remaining diacritics represent bit '0'. This approach achieved high capacity and can be implemented easily. Another approach use diacritics in a multiple way [5]. They proposed generating a number of extra (duplicated) diacritic equal to the binary number representing the message as one block which is found unobservable. Another similar idea is to divide the message into small blocks 'ones and zeros' as a binary number. Then, generate number of extra diacritics equal to this binary number representing that block. Using diacritics may achieve high capacity in general; however, nowadays sending messages with diacritics raise suspicions, which reduce stego security. Also, investigating the diacritics standardization found that it is font dependent. Different Arabic fonts are having different diacritic encodings affecting robustness negatively. This review made our focus of this work to generate an Arabic stego method concentrate on Kashida as described in the following.

III. Proposed Approach

The main idea of our proposed approach is that a word in a cover text, that have some possible extendable characters, can represent some values within the range [0, *X*-1] by inserting at most a specific number of Kashidas. The number of such values can be given by the following equation:

$$X = \sum_{m=0}^{k} \binom{m}{n}, \qquad (1)$$

where n is the number of possible extendable letters, k is the maximum number of Kashidas allowed to be used per word. Equation (1) implies that a block of bits in secret data can be hidden in such word. The block size can be given by the following equation:

$$block\_size = \log_2 \sum_{m=0}^{k} \binom{m}{n}, \qquad (2)$$

For some values, the block size given in (2) will be incremented by 1 according to the value of such block. For example, consider a word with 3 possible extendable letters, and an applied approach uses at most two Kashidas per word. In such case, Equation (1) gives 7 possible represented values (0, 1...6). Whereas, Equation (2) gives a block size of 2 bits which represents 4 values (0, 1...3). Thus, the block size is adjusted to 3, when the value of such block is 4, 5 or 6.

Algorithm 1 and Algorithm 2 show the detailed proposed algorithm for hiding and extracting secret bits, respectively. The algorithms were implemented in C# .net environment.

Algorithm 1 Hiding secret bits

Input: message, cover text
Output: stego text
initialize *k* with maximum Kashidas used per word.

while data left to embed **do**
  get next *word* from cover text
  n ←the number of possible extendable letters.
  if *n* > 0 **then**
    s ← calculate the block size based on *n* and *k.*
    get the value of the next *s*-bit message block.
    determine the *positions* that represent the value.
    insert Kashida into the *word* at *positions.*
    insert the *word* into stego text.
  end if
end while

Algorithm 2 Extracting secret bits

Input: stego text
Output: message
initialize *k* with maximum Kashidas used per word.

while word left to process **do**
  get next *word* from stego text
  n ←the number of possible extendable letters.
  if *n* > 0 **then**
    s ← calculate the block size based on *n* and *k.*
    determine the *positions* of existed Kashidas.
    get the *value* represented by *positions*
    get the *s*-bits block representing the *value*.
    insert the *s*-bits block into message.
  end if
end while

IV. Implementation & Analysis

In this research, we evaluated three scenarios of our proposed approach, i.e. using at most 1, 2, or 3 Kashidas per word, with different cover text media sizes: 70627, 131233, 235125 and 525271 bytes. Files and information were hidden in the cover text documents by the use of the described algorithm, implemented in c# language. The implemented extractor program outputs the extracted data to a binary file. By comparing the output and input files, we observed that both files are identical.

We studied the three scenarios in terms of capacity and security. The capacity can be evaluated by the capacity ratio and useable characters ratio. The capacity ratio is computed by dividing the amount of hidden bytes over the size of covet text media in bytes as in [1]. To compute the usable characters ratio, we count the number of usable characters per approach, independent from the scenario or the secret message to be embedded. We use p for the ratio of characters capable of baring a secret bit of a given level, and q for the ratio of characters capable of baring the opposite level. In the case of our approach, an extendable character may hide a secret bit of '0' or '1'. Thus, such characters may contribute to p and q, i.e. p equals q.

V. Comparison & Remarks

We compare the capacity of our approach to the dots approach of [1], Kashida approach of [2, 3], and to the diacritics approach of [4, 5].

Table 1 shows the capacity ratio of our proposed approach for different scenarios, using one, two, and three Kashidas. The results show that capacity ratio increases when the number of Kashida per word increases. The scenario of using three Kashidas outperforms all other scenarios in our approach. In addition, the results show that our proposed approach using two or three Kashidas outperform the other existing approaches in the literature in terms of capacity, their reported results are shown in Table 2.

Table 1 Capacity ratio of our proposed approach

Approach	Cover Size	Capacity (%)	Average capacity (%)
One Kashida	70627	2.45	2.46325
	131233	2.465	
	235125	2.4526	
	525271	2.4854	
Two Kashidas	70627	3.285	3.303293
	131233	3.2916	
	235125	3.2612	
	525271	3.37537	
Three Kashidas	70627	3.73139	3.73715
	131233	3.71447	
	235125	3.7165	
	525271	3.78624	

Table 2 Reported Capacity Ratio in Literature

Approach	Cover Size	Capacity (%)	Average capacity (%)
Dots [1]	13619.2	1.172	1.37
	6983.68	1.467	
	6799.36	1.275	
	3604.48	1.529	
Kashida [3]	365181	1.215	1.22
	378589	1.172	
	799577	1.266	
	15112	1.244	
Diacritics [5]	318,632	3.25	3.27
	1,34,865	3.256	
	717,135	3.318	
	318,216	3.254	

Table 3 shows the usable characters ratio of our approach. The results show that our proposed approach outperforms the other approaches in terms of usable characters ratio. Our proposed approach shows a ratio of about 3.9, where as, the others show a ratio of about 3.6 as it is shown in Table 4.

Table 3 Usable Characters Ratio of Our Approach

File size (bytes)	p	q	(p+q)/2
70627	39.18	39.18	0.3918
131233	39	39	0.39
235125	38.64	38.64	0.3864
525271	39.2355	39.2355	0.392355

Table 4 Reported Usable Characters Ratio in Literature

Approach	p	q	r	(p+r+q)/2
Dots	0.2764	0.4313	0.0300	0.3689
Kashida-Before	0.2757	0.4296	0.0298	0.3676
Diacritics	0.3633	0.3633	0	0.3633

Figure 1 shows an example to clarify such situation, considering a word of 6 possible extension Kashidas and the secret bits as "001010". According to the existing Kashida approach in [3], 6 extension Kashidas will be inserted in this word. However, our proposed approach improves this issue by restricting the inserted Kashidas to be at most specific number per word. Using two Kashidas per word, shows adequate capacity ratio and implies more security compared to others.

Secret bits	001010
A word in the cover text	سنمتعهم
Output of proposed method	سنمتعـهم
Output of the method in [3]	سنمـتـعـهم

Figure 1 clarifying the improvement in the security

VI. Conclusion

The paper proposed a novel algorithm for text steganography in Arabic language. The technique can also be applied to similar language texts such as Urdu and Persian as some Semitic languages. The main idea is to use the locations of possible extendable characters within a given word in the cover text media to hide secret data bits. The secret data is represented within a word by inserting Kashidas after some extendable characters in the word.

We implemented different scenarios of this approach. We experimentally evaluated and compared the capacity of our work with the existing Arabic text steganography approaches in literature. It was shown that our proposed method is superior and provides better capacity than all others. Also, our proposed approach showed promise of more security compared to all the existing Kashida approaches as it restricts the number of Kashidas used per word adding more confusion. It is believed that this research will open new directions for promising techniques towards Arabic text steganography.

Acknowledgment

Authors would like to thank King Fahd University of Petroleum & Minerals (KFUPM), Dhahran, Saudi Arabia, for supporting this research work.

References

[1] M. Hassan Shirali-Shahreza, Mohammad Shirali-Shahreza, "A New Approach to Persian/Arabic Text Steganography," 5th IEEE/ACIS International Conference on Computer and Information Science, (ICIS-COMSAR 06), pp. 310- 315, July 2006.

[2] Adnan Gutub, Lahouari Ghouti, Alaaeldin Amin, Talal Alkharobi, and Mohammad K. Ibrahim, "Utilizing Extension Character 'Kashida' With Pointed Letters For Arabic Text Digital Watermarking", International Conference on Security and Cryptography - SECRYPT, Barcelona, Spain, July 28 - 31, 2007.

[3] Adnan Gutub and Manal Fattani, "A Novel Arabic Text Steganography Method Using Letter Points and Extensions", WASET International Conference on Computer, Information and Systems Science and Engineering (ICCISSE), Vienna, Austria, May 25-27, 2007.

[4] Mohammed Aabed, Sameh Awaideh, Abdul-Rahman Elshafei, and Adnan Gutub, "Arabic Diacritics Based Steganography", IEEE International Conference on Signal Processing and Communications (ICSPC 2007), Pages: 756-759, Dubai, UAE, 24-27 November 2007

[5] Adnan Gutub, Yousef Elarian, Sameh Awaideh, and Aleem Alvi, "Arabic Text Steganography Using Multiple Diacritics", WoSPA 2008 – 5th IEEE International Workshop on Signal Processing and its Applications, University of Sharjah, Sharjah, U.A.E. 18 – 20 MARCH 2008.

[6] Al-Sulaiti, Latifa; Atwell, Eric., "The design of a corpus of contemporary Arabic", International Journal of Corpus Linguistics, vol. 11, pp. 135-171. 2006.

[7] J.C. Judge, "Steganography: Past, Present, Future", SANS white paper, http://www.sans.org/rr/papers/ , November 30, 2001.

[8] K. Bennett, "Linguistic Steganography: Survey, Analysis, and Robustness Concerns for Hiding Information in Text", Purdue University, CERIAS Tech. Report 2004-13, 2004.

[9] D. Vitaliev, "Digital Security and Privacy for Human Rights Defenders," The International Foundation for Human Right Defenders, pp. 77-81, Feb. 2007.

[10] P. Wayner., "Disappearing Cryptography: Information Hiding: Steganography and Watermarking", Morgan Kaufmann, 2nd edition edition, 2002.

[11] Mahmoud, S.A.; Mahmoud, A.S., "Arabic Character Recognition using Modified Fourier Spectrum (MFS)," Geometric Modeling and Imaging--New Trends, 2006 , vol., no., pp. 155-159, 05-06 July 2006.

[12] Mohammad Tanvir Parvez and Adnan Gutub, "RGB Intensity Based Variable-Bits Image Steganography", *APSCC 2008 – Proceedings of 3rd IEEE Asia-Pacific Services Computing Conference*, Yilan, Taiwan, 9-12 December 2008.