Improved Method of Arabic Text Steganography Using the Extension ‘Kashida’ Character

Adnan Abdul-Aziz Gutub, Wael Al-Alwani, and Abdulelah Bin Mahfoodh

***Abstract* — In this paper a new Steganography approach for Arabic texts is reported. The approach hides secret information bits within Arabic letters benefiting from the redundant extension character "-" known as Kashida. To note the specific letters holding encoded secret bits, the method considers the number of the extension character inserted after any letter that can hold it. This paper introduces this new approach and discusses the additional features it possesses like bits optimization using mapping tables and dynamic assignment of letter codes. This approach is found attractive and can be modified to enhance the security and the capacity features in Arabic and other languages having similar texts such as Urdu and Persian.**

***Index Terms* — Arabic e-Text, Text Steganography, Information Security, Text Hiding.**

I. Introduction

The ever increasing number of networks users, e.g. the Internet, drives the process of enhancing security into more serious measures as more victims and attackers are brought into these networks. In the second quarter of 2008, more than 1.46 billion Internet users were exchanging data [4]; data that can be very sensitive for each user. So, data’s confidentiality must be preserved. One way to strengthen the confidentiality property of the exchanged data is steganography.

Steganography is described as the art and science of hiding data in an unremarkable cover object so that an eavesdropper will have no suspicions regarding this communication [7]. Steganography systems can be implemented in texts, pictures, and sound files. In this paper we will deal with Steganography in text files.

The steganography system has two inputs as shown in Figure 1: a plain text, “Cover Object”, and a message, “Secret Object”, which is confidential. Steganography algorithm comes into picture to do the embedding part for the two inputs, i.e. the “Cover Object” and the “Secret Object”, and then outputs the “Stego Object”.

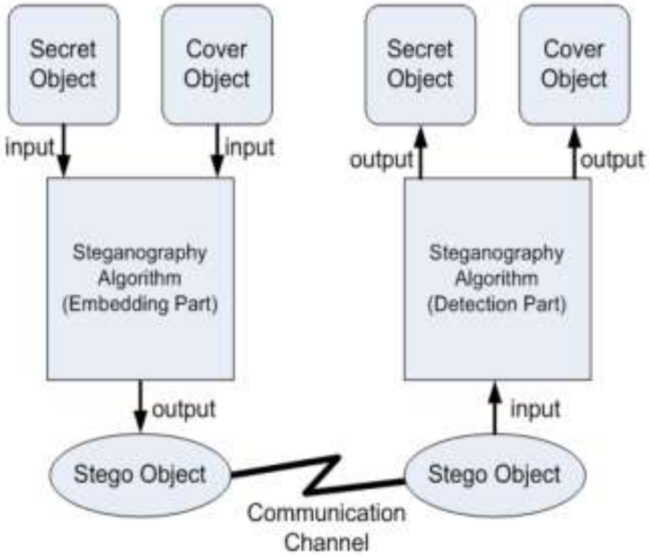

Fig. 1. Steganography system

The algorithm implemented in a steganography system has to consider three features: capacity, security, and robustness [3]. First, capacity is the amount of the Secret Object’s bits that can be hidden in the Cover Object. Second, security is to make an eavesdropper unable to detect the existence of the Secret Object’s data in the Stego Object. Finally, robustness is the amount of tolerance a Stego Object can have when modified by an attacker to keep the integrity of the embedded Secret Object valid [7].

Section 2 of this paper discusses related published work in Arabic text Steganography. Also, it lists advantages and disadvantages of each work. In section 3, our new Arabic text Steganography method is discussed. After that, section 4 introduces a comparison of two different approaches with our proposed approach. Finally, the paper ends with a conclusion in section 5.

II. Related Arabic Text Steganography Attempts

Recent works have focused on the development and the potential applications of Arabic script steganography. The first proposal in this field was done by Shirali-Shaherza [5, 12]. Their schema is based on hiding binary values into Arabic or Persian scripts using a feature coding method. This method depends on the points inherited in the Arabic, Urdu and

Adnan Abdul-Aziz Gutub
P. O. Box 6287
Center of Excellence in Hajj and Omrah Research
Umm Al-Qura University, Makkah 21955, Saudi Arabia
Email: aagutub@uqu.edu.sa

Wael Al-Alwani
Sulaimania, 30th street, Riyadh, Saudi Arabia
E-mail: wael.alalwani@gmail.com

Abdulelah Bin Mahfoodh
Olayya, Dabab street, Riyadh, Saudi Arabia
E-mail: bin.mahfoodh@gmail.com

Persian letters.

The points' location within the pointed letters hide information as follows: First, the length of the hidden information, i.e. the Secret Object, is looked at as binary with the first several bits. Then, the medium text, i.e. the Cover Object, is scanned. Whenever a pointed letter is detected, the location of the point may be affected if hidden binary value is one or zero. The location of the point is slightly shifted up if the hidden bit value is one as shown in figure 2; otherwise, the location remains unchanged [10].

This schema has many advantages; for example, it has high capacity in storing large number of hidden bits as the Arabic language has 15 letters out of 28 letters that have points. However, one of the main disadvantages of this method is in robustness. The output font is not standard. Moreover, the receiver will not be able to extract the secret message if the output font is not installed on his machine. Also, the hidden information can be lost in any retyping or scanning process.

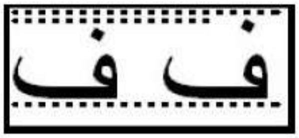

Fig. 2. Shifting the letter "Fa'a" point up to hide bit value equals 1

Another proposal [6] aims to utilize the advantages of diacritics in Arabic scripts [11] to implement steganography. There are eight different diacritical symbols in Arabic, and they are used in this approach to hide binary bits in the original cover text. The team, who proposed this approach, found that in Standard Arabic, the frequency of one diacritic, namely Fat'ha, is equal to the total frequency of the other seven diacritics. So, they assigned in this approach the diacritic Fat'ha to the bit value equals one, and the remaining seven diacritics were assigned to the bit value equals zero.

To implement this approach, a diacritized Arabic text is used as a Cover Object. Then, a computer program reads the first bit of data needed to be embedded. If the first bit was a one and the first diacritic in the cover media was Fat'ha, the diacritic is kept in the cover media and the index for both the embedded data and the cover media is incremented. However, if the diacritic is not Fat'ha and the bit value is one, the diacritic is removed from the cover media and, in the same time, the index of the cover media only is incremented to read the next diacritic. The same process is implemented for the bit value zero, except that a zero will search for all the other seven diacritics instead of the Fat'ha. An example to this approach can be seen in figure 3.

The use of diacritics in Standard Arabic language is optional [11]. Moreover, it is uncommon nowadays to write a diacritized Arabic text. Therefore, a drawback for this approach is the high probability of raising suspicions for an eavesdropper about the existence of a secret message when using this approach. Another important drawback is that the receiver has to have the original text so that the extracting algorithm can compare the diacritics in the Stego Object with the original Cover Object to extract the Secret Object.

Cover Object	حَدَّثَنَا سُفْيَانُ عَنْ يَحْيَى
Secret Object	E7 (= 11100111)
Stego Object	حَدَّثَنَا سُفْيَان عَن يَحْيَى

Fig. 3. Hiding "E7" using diacritics approach

The work in [2] presents another approach that uses diacritics in Arabic scripts. This approach uses the idea of the way how computers display and print Arabic diacritic marks. When the code of a diacritic mark is encountered in almost all Arabic fonts, the corresponding image is rendered and displayed as an output without changing the place of the cursor. This leads to the possibility of inserting multiple diacritic marks in an almost invisible way. Then, a computer program can detect the presence of these multiple diacritics, extract them, and interpret the way they are inserted in the text according to any encoding scenario. There are two main approaches presented in this same idea: The textual approach, and the image approach.

The textual approach which chooses a font that hides multiple diacritics according to many encoding scenarios such as: the direct and blocked value scenarios and the RLE scenario. The differences between these scenarios vary among capacity, robustness, and security.

The image approach selects one of the Arabic fonts that can darken multiple occurrences of diacritics. The produced document should be converted into image in order to determine the level of the brightness and encode it into the corresponded meaning. There are some limitations in this approach such as the block or file size, but in general it has advantages in capacity, robustness, and security.

Another proposed approach [1,9,13,14] uses the redundant Arabic extension character "-" which is called "Kashida". We will use the word Kashida through out the context of this paper to indicate this extension character. Arabic language has 28 main letters and a Kashida character which is considered as a redundant character meant for formatting the Arabic electronic typing. However, due to the nature of Arabic writing, the Kashida character cannot be added at the beginning or ending of words. It can be added between connected letters in a word. Note that adding the Kashida character does not affect the Arabic contextual meaning [1,9].

In this approach presented in [1] and [9], if a Kashida is used after a pointed Arabic letter in the cover object, then a secret bit which equals to one is hidden. On the other hand, a secret bit equal to zero is hidden if an un-pointed Arabic letter is followed by a Kashida as shown in figure 4. Using this character enhances the features of security and robustness. But, it might have some drawbacks in capacity of the cover medium if the size of secret bits in the Secret Object is large.

Secret bits	110010
Cover-text	من حسن اسلام المرء تركه مالا يعنيه
Steganographic text	مـن حسن اسلام المرء تـركـه مـالا يـعـنـيه ↑↑ ↑ ↑ ↑ ↑ 1 1 0 0 1 0

Fig. 4. Hiding secret bits using Kashida character

III. Proposed Arabic Text Steganography Method

The secret object is hidden in the form of zeros and ones which represents the 16-bit Unicode of each character (using the UTF-8 encoding scheme which uses 16 bits to represent one Arabic character [8]). A common drawback in all previous approaches is that they embed the bits without having some optimization. Meaning that, these approaches will embed the 16 bits of each character in the Cover Object. Our novel method benefits from the work with the Kashida character as proposed by Gutub and Fattani [1].

We will add one Kashida representing secret bit = 0 and two consecutive Kashidas when bit = 1. The Kashida will be placed after any letter that can hold it. The optimization part of the algorithm deals with the message to be hidden, i.e. the Secret Object. As mentioned before, Arabic language has 28 main letters. But there are special forms of a letter like in letter “Alef (ا)”: (أ , إ , آ , …) which are used in Arabic writing and each one has a UTF-8 representation. So, the number of letters and forms sums up to more than 32 and less than 64. Since each letter and form is represented by 16 bits, we used a mapping table in which each letter was assigned, instead, a 6-bit code to save 10 bits. In the mapping table, we assigned the 6-bit codes starting from 000000 and incremented by one to all letters and forms ordered alphabetically. As a result, saving 10 bits by implementing the mapping table means, enhancing the capacity feature of this method.

One concern is that when the Cover Object for example is one page long and the Secret Object is only one word, then, the Stego Object will only contain Kashidas in the first few lines. So, this will be suspicious for an eavesdropper who might infer the existence of a hidden message in the text.

So, we solved this by creating a special character named “finishing character” which has the code 111111 and it will be embedded just after the last letter of the message, i.e. the Secret Object. After this “finishing letter”, the algorithm will randomly add Kashidas to the whole text to enhance the security feature of this method. Finally, extracting the message from the Stego Object is done by collecting the Kashidas back and when 6 bits are collected, the algorithm checks the mapping table to determine the corresponding letter. When it detects the finishing character, it stops.

Figure 5 shows how the method works. The first letter in the Secret Object is “Ba’a” which has the code 000001 in the mapping table. We can see that the first and second letters in the first word in the Cover Object can hold Kashidas, so one Kashida is added after each one representing the first and second 0 bits of the code 000001, as shown in the Stego Object. The third letter can not hold a Kashida, so it is kept as is. The second and third words in the Stego Object hold the 4th, 5th, and 6th zero bit of the code, i.e. 000001. Finally, the fourth word is holding two consecutive Kashidas representing the remaining bit of the code which is 1, i.e. 000001.

Secret Object	بدأ اختبار
Cover Object	ميزة هذا النوع من المعالجات أنه يقضي مدة ثابتة في تنفيذ أي تعليمة، ومقدار هذا الوقت هو دورة واحدة يحدد زمنها أطول تعليمة من مجموعة التعليمات وهي تعليمة القراءة من وحدة الذاكرة، وهذا يعني سهولة كبيرة في تصميم المعالج
Stego Object	مـيـزة هـذا الـنـوع مــن الـمـعـالــجــات أنــه يـقـضــي مــدة ثـابـتــة فــي تـنـفـيـذ أي تـعـلـيـمـة، ومـقـدار هـذا الـوقــت هــو دورة واحـدة يـحـدد زمـنـهــا أطـول تـعـلـيـمـة مــن مـجـمـوعـة الـتـعـلـيـمـات وهـي تــعـلــيـمــة الــقــراءة مـن وحـدة الـذاكـرة، وهـذا يـعـنــي سـهـولـة كـبـيـرة فـي تـصـمـيـم الـمـعـالـج

Fig. 5. Embedding secret bits using Kashidas

We found that adding one Kashida when the bit of the Secret Object is 0 and two consecutive Kashidas when the bit is 1 after any letter or form that can accept Kashidas is probably suspicious. An eavesdropper would find Kashidas at any applicable position for a Kashida in the Stego Object.

We propose a double impact solution that can increasingly enhance the capacity and the security feature. We modified the algorithm of our method to avoid adding Kashida when the letter code in the mapping table has two consecutive zero bits. In other words, the code of letter “Ba’a” which is 000001 will not be embedded in the Cover Object as: 1 Kashida, 1 Kashida, 1 Kashida, 1 Kashida, 1 Kashida, 2 Kashidas. Instead, it will be embedded as: no Kashida, no Kashida, 1 Kashida, 2 Kashidas. Obviously, we saved two more bits (Kashida) locations, which mean that this method will use only 4 Kashida locations to embed the 6-bit letter code. This allowed the absence of Kashidas in the Stego Object to remove the suspiciousness that may arise, i.e. improving the security feature of the method.

A final modification has been made to enhance the capacity more. We modified the method to check the Secret Object, before embedding it, and make statistics on it to find the most frequent (existing) letters. Then, the method will assign the letter code which have more consecutive zeros in the mapping table to the most frequent letters. For example, if we find that the Secret Object has the letter “Lam” (ل) as the most frequent letter to occur and then the letter “Noon” (ن), then the method will assign the codes 000000 to “Lam” and 000001 to “Noon” in the mapping table. Therefore, we will enhance the capacity by using the frequency of the occurrence of letters. By listing all the codes combinations that use 6 bits, we found that there are 13 codes which can be embedded using 4 Kashida locations such as 000010 and 010000. Hence, the method can assign these codes to the most 13 occurring letters. We also found that 29 code combinations can be embedded using 5 Kashida locations. As a result, these 29 codes will be assigned to the rest of the Arabic letters that exist in the Secret Object. Note that calculating the frequencies is a dynamic process and each time the Secret Object changes, the calculation is applied and hence, the assignment of codes for most occurring letters is changed accordingly.

IV. Comparison With Published Methods

We tested three approaches to compare their results. The aim was to decide which approach efficiently maintains the capacity feature. The first approach was Shirali-Shaherza's approach and the second one was Gutub and Fattani's approach. The third approach was the enhanced version of our proposed approach which calculates statistics on the Secret Object to assign 6-bit codes, which need 4 or 5 Kashida locations, to the most frequently occurring letters.

The test was to embed a Secret Object which contains one five-letter word, i.e. 5*16 bits for each letter = 80 bits, in many different Cover Objects which contain, in average, 116 words. The results, as shown in table 1, show that using Shirali-Shaherza's approach, the Secret Object used, in average, 74.32% of the total capacity found in most of the Cover Objects. Gutub and Fattani's approach used, in average, 33.68%. Our enhanced version approach used, in average, 10.25% of the total capacity of each Cover Object which seems to mean that our approach outweighs the two other approaches in terms of capacity efficiency.

TABLE I
Results of comparing the different approaches.

Approach	Average Capacity Usage in Each Cover Object
Shirali-Shaherza (pointed letters)	74.32%
Gutub and Fattani (Kashidas)	33.68%
Our enhanced approach	10.25%

In terms of security feature, Shirali-Shaherza's method is probably the best method to maintain this feature especially if the points of the letters are so slightly shifted that is hard to notice them. Our enhanced approach is better than Gutub and Fattani's approach in terms of security feature because it inserts less number of Kashidas in different places that do not follow any kind of inserting patterns as used in Gutub and Fattani's approach, i.e. Gutub and Fattani's approach inserts Kashida after a pointed letter if the secret bit = 1 and after an un-pointed letter if the bit = 0.

V. Conclusion

This paper presents a very interesting and novel steganography method useful for Arabic and other similar languages. This method benefits from the feature of having the Kashida character in Arabic script "-". We use a mapping table in which each letter of the Arabic language is assigned a specific 6-bit code to save ten bits, as the letter's Unicode representation using UTF-8 consists of 16 bits. The Secret Object letters are transformed into secret bits using the corresponding code for each letter found in the mapping table. These secret bits are represented as follows: One extension letter will be inserted after a letter that can hold it from the Cover Object if the secret bit is 'zero'. The same process will happen if the secret bit is 'one', but with inserting two consecutive extension letters instead of one.

The method can be modified to remarkably enhance the capacity and to eliminate suspiciousness of an eavesdropper by not inserting extension letter to represent 2 consecutive zeros, if found, in the 6-bit code of a letter. Also, by checking the Secret Object before embedding it and make some statistics on it to determine the most frequent letters will make the method to be dynamic in regard of the assignment of codes in the mapping table, which will help to increase the capacity. This approach enhanced and featured security, capacity, and robustness, which makes it useful to help Arab users to exchange information through text documents and establish secure communication.

Acknowledgment

We would like to thank King Fahd University of Petroleum and Minerals (KFUPM) for partially hosting this research. Special thanks to Dr. Ahmad Al-Mulhem the instructor of the course COE499: Network Security Engineering for his positive cooperation. Thanks to Center of Research Excellence in Hajj and Omrah, Umm Al-Qura University (UQU), Makkah, for moral support toward the achievements in this work.

References

[1] 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.

[2] 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, United Arab Emirates 18 – 20 MARCH 2008.

[3] B. Chen and G.W. Wornell, "Quantization Index Modulation: A Class of Provably Good Methods for Digital Watermarking and Information Embedding," *IEEE Trans. Information Theory*, Vol. 47, No. 4, 2001, pp. 1423-1443.

[4] *Internet World Statistics Website*. Retrieved September 3, 2008, from <http://www.internetworldstats.com/stats.htm>.

[5] 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)*, July 2006, pp. 310- 315.

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

[7] N. Provos and P. Honeyman, "Hide and Seek: An Introduction to Steganography", *IEEE Security & Privacy*, May/June 2003, pp. 32-44.

[8] 'UTF-8', *Wikipedia The Free Encyclopedia*. Retrieved September 3, 2008, from < http://en.wikipedia.org/wiki/UTF-8>.

[9] Adnan Gutub, L. Ghouti, A. Amin, T. Alkharobi, and M.K. Ibrahim, "Utilizing Extension Character 'Kashida' With Pointed Letters For Arabic Text Digital Watermarking", *Proceedings of the International Conference on Security and Cryptography (SECRYPT)*, Barcelona, Spain, 2007.

[10] G. Abandah, and F. Khundakjie, "Issues Concerning Code System for Arabic Letters", *Dirasat Engineering Sciences Journal*, Vol. 31, No. 1, 2004, pp. 165-177.

[11] M. Al-Ghamdi, and M. Zeeshan, "KACST Arabic Diacritizer", *Proceedings of the First International Symposium on Computers and Arabic Language*, 2007.

[12] M.H. Shirali-Shahreza, and M. Shirali-Shahreza, "A New Approach to Persian/Arabic Text Steganography", *Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2006)*, Honolulu, HI, USA, 2006, pp. 310-315.

[13] Adnan Gutub and Ahmed Al-Nazer, " High Capacity Steganography Tool for Arabic Text using 'Kashida' ", *The ISC Int'l Journal of Information Security (ISeCure)*, Vol. 2, No. 2, Pages 109-120, July 2010.

[14] Adnan Gutub, Fahd Al-Haidari, Khalid Al-Kahsah, and Jameel Hamodi, "e-Text Watermarking: Utilizing 'Kashida' Extensions in Arabic Language Electronic Writing", *Journal of Emerging Technologies in Web Intelligence (JETWI)*, Vol. 2, No. 1, Pages: 48-55, February 2010.

Adnan Abdul-Aziz Gutub

Adnan Abdul-Aziz Gutub is currently working as the Director of the Center of Research Excellence in Hajj and Omrah (HajjCoRE) at Umm Al Qura University, Makkah Al-Mukarramah, all Muslims religious Holy City located within the Kingdom of Saudi Arabia.

Before this administrative position, he worked as Chairman of the Information Systems Department at the College of Computer & Information Systems, also within Umm Al Qura University – Makkah.

Adnan's rank is an associate professor in Computer Engineering previously affiliated with King Fahd University of Petroleum and Minerals (KFUPM) in Dhahran, Saudi Arabia. He received his Ph.D. degree (2002) in Electrical & Computer Engineering from Oregon State University, USA. He has his BS in Electrical Engineering and MS in Computer Engineering both from KFUPM, Saudi Arabia.

Adnan's research interests are in optimizing, modeling, simulating, and synthesizing VLSI hardware for crypto and security computer arithmetic operations. He worked on designing efficient integrated circuits for the Montgomery inverse computation in different finite fields. He has some work in modeling architectures for RSA and elliptic curve crypto operations. His interest in computer security also involved steganography such as simple image based steganography and Arabic text steganography.

Adnan has been awarded the UK visiting internship for 2 months of summer 2005 and summer 2008, both sponsored by the British Council in Saudi Arabia. The 2005 summer research visit was at Brunel University to collaborate with the Bio-Inspired Intelligent System (BIIS) research group in a project to speed-up a scalable modular inversion hardware architecture. The 2008 visit was at University of Southampton with the Pervasive Systems Centre (PSC) for research related to advanced techniques for Arabic text steganography and data security.

Adnan Gutub filled an administrative academic position at KFUPM; before moving to Umm Al-Qura University, he had the experience of chairing the Computer Engineering department (COE) at KFUPM from 2006 to 2010.

Wael Al-Alwani

Wael Al-Alwani holds a bachelor degree in computer engineering. He has graduated from KFUPM, Saudi Arabia in 2008.Wael has taken many courses related to Networks field and Information Security along with the Computer Engineering department professors in KFUPM.

Wael and Dr. Adnan Gutub (main author of this paper) have invented a novel idea on which StegoTech products and services are based. Wael has worked in an IT company where he was exposed to different technologies and had a great exposure on the industry world.

Wael holds many technical certificates (Sun, Cisco, Oracle). While in University, he was assigned as the president of the College of Computer Science and Engineering Student Club at KFUPM at which he successfully revived the club along with a carefully chosen team and ranked the 2nd best club in KFUPM among all 16 different clubs in one year. The club was at the very bottom of the list during the previous 6 years. He gained a lot of skills in managing people and compromising conflicted opinions.

Abdulelah Bin Mahfoodh

Abdulelah Bin Mahfoodh holds a bachelor degree in computer engineering from KFUPM, Saudi Arabia in 2008. He has taken many courses related to Information & Communications Security, specifically Network Security Engineering and Applied Cryptosystems: Techniques & Architectures (attended with Dr. Adnan Gutub - main author of this paper).

Abdulelah holds many technical certificates and he was very active through the Computer Student Club at KFUPM at which he successfully helped improving the club contribution along with the wonderful team. He gained a lot of skills in working with people in teams and compromising different conflicted opinions.