ARABIC TEXT STEGANOGRAPHY USING MULTIPLE DIACRITICS

Adnan Abdul-Aziz Gutub, Yousef Salem Elarian, Sameh Mohammad Awaideh, Aleem Khalid Alvi

Computer Engineering Department, King Fahd University of Petroleum & Minerals, Saudi Arabia

ABSTRACT

Steganography techniques are concerned with hiding the existence of data in other cover media. Today, text steganography has become particularly popular. This paper presents a new idea for using Arabic text in steganography. The main idea is to superimpose multiple invisible instances of Arabic diacritic marks over each other. This is possible because of the way in which diacritic marks are displayed on screen and printed to paper. Two approaches and several scenarios are proposed. The main advantage is in terms of the arbitrary capacity. The approach was compared to other similar methods in terms of overhead on capacity. It was shown to exceed any of these easily, provided the correct scenario is chosen.

Keywords— Arabic; capacity; diacritic marks; steganography; text hiding.

1. INTRODUCTION

Since ancient times, people and nations seek to keep some information secure. Steganography is the approach of hiding the very existence of secret messages, hence securing them [16]. Steganography has gained much importance today, in the era of communications and computation. Figure 1, from [1], point out a classification tree of steganography.

The first category in the classification divides steganography according to the cover message type. We are proposing two approaches that would fit the text and image classes, according to these categorizations. The linguistic categorization exploits the computer-coding techniques to hide information [1]. Semagrams hide information through the use of signs and symbols. According to this second classification, we fit the text and visual semagrams class.

In the following section, we present some background information on Arabic script. In the next section, we review work related to Arabic script steganography. Next, the Approach section is devoted to describe our two approaches and compare them to each others. Afterwards, we show the results of some testing. Finally, we conclude, acknowledge and provide a list of references.

2. BACKGROUND ON ARABIC SCRIPT

The Arabic alphabet has Semitic origins derived from the Aramaic writing system. Arabic diacritic marks decorate consonant letters to specify (short) vowels [2]. Those marks, shown in Figure 2, normally come over/beneath Arabic consonant characters. Arabic readers are trained to deduce these [3, 4]. Vowels occur pretty frequently in languages. Particularly in Arabic, the nucleus of every syllable is a vowel [5]. Inside the computer, these are

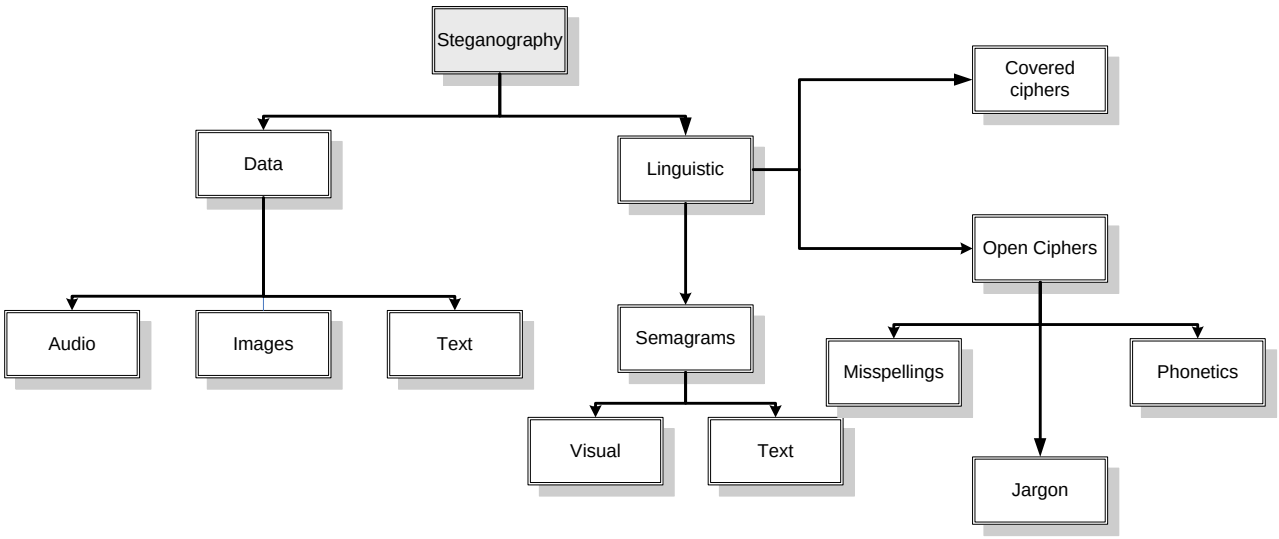


Figure 1. The classification tree of steganography [1].

represented as characters [6]. The use of diacritics is an optional, not very common, practice in modern standard Arabic, except for holy scripts.

Dots and connectivity are two inherent characteristics of Arabic characters. We describe them here for the convenience of Sections 3 and 5. We use the word dots to refer to any separate stroke that comes over or beneath otherwise identical glyphs to differentiate among them. This includes any single, double, and triple points, besides the zigzag shapes called Hamzahs, and Maddahs. Out of the Arabic basic alphabet of 28 letters, 15 letters have from one to three points [7, 10], four letters can have a Hamzah, and one can be adorned by Maddah [8]. Ancient Arabs used to omit and deduce dots in the same manner in which standard Arabic treats diacritics today.

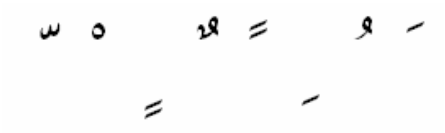


Figure 1. Arabic diacritic marks.

Connectivity is a result of the cursive nature of Arabic script. However, 8 out of the 28 Arabic letters do not connect to subsequent letters. Besides, even connectable letters do not connect to subsequent letters when the end of the word has been reached. These issues also restrict the insertion of the connectivity elongation redundant character, Kashida.

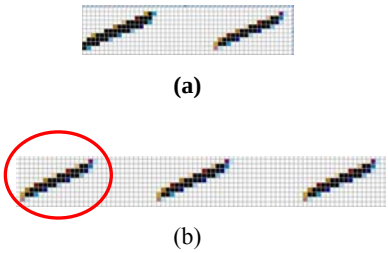
3. RELATED WORK

Little has been proposed on Arabic script steganography. Two inherent properties of Arabic writing, however, have been proposed: dots and connectivity. Dots are interesting for their frequent occurrences in Arabic text. A first proposal of their use has tackled the character design itself [9]. In this method, the position of dots is changed to render robust, yet hidden, information. The method needs special fonts to be installed and give different codes to the same Arabic letter depending on the secret bit it hides. A more practical way has been suggested in [10] and [15]. It distinguishes the secret-bit-hiding dotted letters by inserting Kashidah's before/after them. A small drop in capacity occurs due to restriction of script on Kashidah insertion from one side, and due to the extra-Kashidah's increasing the overall size of text, on the other side. A variation to the work of [10] and [15] that simply inserts a Kashidah after an extendible character to represent a binary bit, regardless of the previous character's dots, might achieve better capacity.

Aabed *et al*. [11] have made use of the redundancy in diacritics to hide information. By omitting some diacritics, meaningful streams in them can be kept. This paper shares the base idea and extends it to the usage of multiple instances of diacritic marks, benefiting from the display characteristics of such marks.

4. APPROACH

The idea emerges from the way how computers display/print Arabic diacritic marks. For most Arabic fonts, when the code of a diacritic mark is encountered, the image of the corresponding stroke is rendered to the screen/printer without changing the location of the cursor. Such displaying without displacing leads to the possibility of typing multiple instances of a diacritic in an almost invisible way. A computer program aware of the presence and meaning of such diacritics can detect and interpret them. For example, a program can be aware that a multiple diacritics exist in a message. It then can easily extract them, as Figure 3 suggests.



(a)

(b)

Figure 2. Diacritics of enciphered text: (a) before & (b) after in circle: uncovering the last extra diacritic.

We emphasize on the word almost when qualifying the invisibility of extra diacritics. This fact is because the multiple typing of a diacritic character might have an effect on the displayed/printed output in some fonts. In fact, fonts range from making all diacritics completely invisible to revealing them all in an apparent unfolded manner. In between, there are two interesting cases: the one of revealing only the first diacritic mark, and hiding extra strokes, and the one of darkening the diacritics with extra strokes.

We provide two approaches to exploit the ideas above: The textual approach and the image approach. Each approach has its advantages in terms of the typical steganography metrics [10]: security, capacity, and robustness [12]. Tradeoffs between the metrics in the approaches are discussed after their presentation. The textual approach chooses a font that hides extra (or maybe all) diacritic marks completely. It, then, uses any encoding scenario to hide secret bits in an arbitrary number of repeated but invisible diacritics. Clearly, a softcopy of the file is needed to retrieve the hidden information (by special software or simply by changing the font).

4.1 The Textual Approach

The Direct and Blocked Value Scenarios

There are several scenarios to make use of this approach. One extreme scenario of this method achieves an arbitrary capacity: The whole message can be hidden in a single diacritic mark by hitting (or generating) a number of extra-diacritic keystrokes equal to the binary number

representing the message. For example, to hide the binary string $(110001)_b = 49_d$, we can follow the step below with n = 50. We get the result in Table 1.

This number *n* might be huge! One solution can be to perform the previous scenario on a block of limited number of bits. For this scenario, consider the same example of $(110001)_b$ as a secret message, we repeat the first diacritic 3 extra times ($3 = (11)_b$); the second one, 0 extra times ($0 = (00)_b$); and the third one, 1 extra time ($1=(01)_b$). Figure 4 manifests the pseudo-code of such general case. A second scenario can be analogous to the run-length encoding (RLE) compression approach.

```
For block b_i containing a number n_d
   Repeat the i^th diacritic n_d times.
```

Figure 4. Pseudo-code for the value scenarios.

The RLE Scenario

In the RLE (run-length encoding) scenario, we repeat the first diacritic mark in text as much as the number of consecutive, say, *ones* emerging in the beginning of the secret message stream. Similarly, the second diacritic is repeated equivalently to the number of the consecutive *zeros* in the secret text. In the same way, all oddly-ordered diacritics are repeated according to the number of next consecutive ones, and all the evenly-ordered ones repeat according to the zeros. Figure 5 presents a pseudo-code describing this method.

```
While(secret.hasMore & cover.hasMore
   b = b^
   While(secret.b = b)
      Type diacritic
```

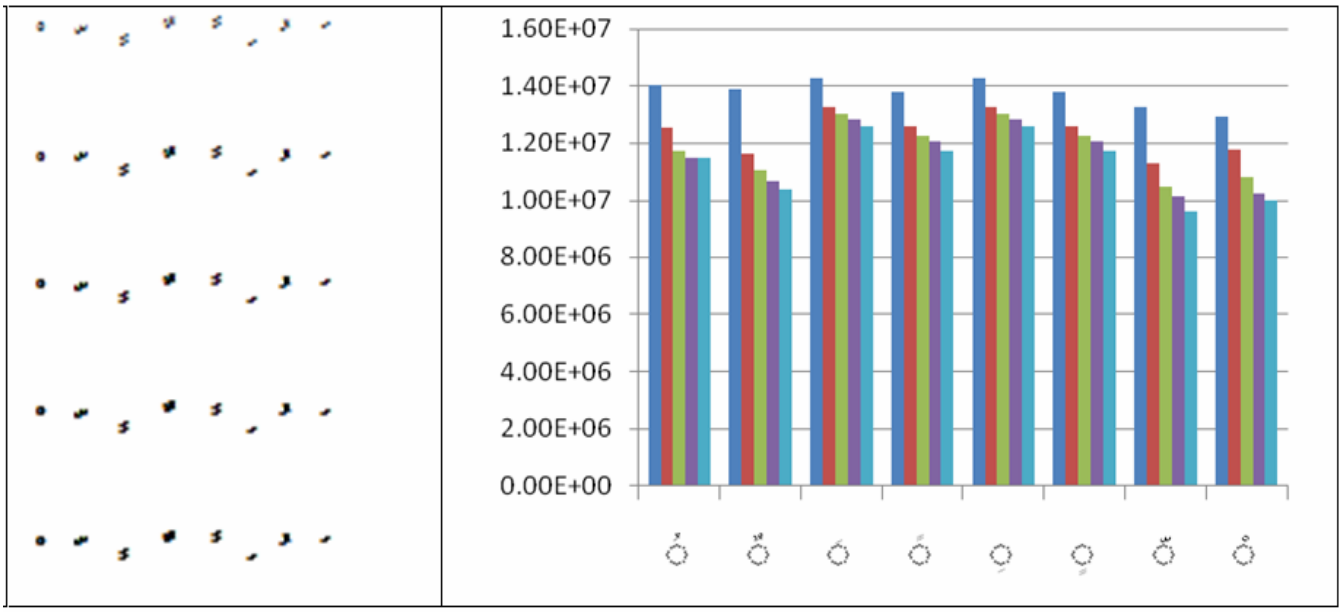
Figure 5. Pseudocode for the RLE scenarios.

For seek of completeness, we reproduce the example for the RLE case, as well. The algorithm will imply repeating the first diacritic 2 times (2 = number of 1's in $(11)_b$); the second one, 3 times (3 = number of 0's in $(000)_b$); and the third one, 1 time (for *1*).

4.2 The Image Approach

The image approach, on the other hand, selects one of the fonts that slightly darken multiple occurrences of diacritics. Figure 6 (a) shows how black level of the diacritics is darkened by multiple instances. Figure 6 (b) quantizes the brightness levels of such diacritics by adding the 24 colour-bits of each as one concatenated number. Notice that the less the brightness level, the more the darkness is.

This approach needs to convert the document into image form to survive printing. This step is necessary because the printing technology differs from the displaying technology in rendering such Arabic complex characters [12]. We found that printing doesn't darken



(a) **(b)**

Figure 6. (a) The image of diacritics, from a single instance in first row up to 5 repetitions in the fifth 2, 3, 4 and 5 times each (presented from the leftmost to the right most column of each diacritic).
(b) Quantization of the brightness levels of such diacritics by adding the 24 color-bits.

extra diacritic instances of text, even when the display does. This unfortunate fact reduces the possible number of repetition of a diacritic to the one that can survive a printing-and scanning process (up to 4 as the last two columns of the first diacritic in Figure 3 (b) suggest).

Table 1. Results of encodings of the binary value 110001 according to the two scenarios of the first approach

Scenario	Extra diacritics
1st scenario (stream)	49.
1st scenario block size=2	3 + 0 + 1 = 4.
2nd scenario (RLE start=1)	(2-1) + (3-1) + (1-1) = 3.

These limitations force us to stick to the first encoding scenario with a small block size (up to 2, perhaps). More catastrophically, yet, the size of the image containing text is, typically by orders, larger than that of the text it represents! However, if the media is paper, this capacity measure re-considers the number of characters in a printed page rather than the number of bits. This method can also be considered for printing watermarking. It's worth mentioning that to increase security it's best to transform the text or image into a common format, such as PDF, for example. This act not only hides some information regarding the original type and size of files, but also prevents from accidental or intentional font changes, which can have catastrophic impact on text messages.

Notes on the capacity, robustness and security of each approach are summarized in Table 2. The image approach has two entries: one assuming a softcopy of the document image is distributed and the other one assuming a printed version is. The text approach is not, generally, robust to printing. However, it is capable of achieving arbitrarily high capacities. The file size might deteriorate the security level, however, if this approach is abused. The image approach is, to some extent, robust to printing. The softcopy version is only mentioned for completeness. It has a very low capacity. Its security is also vulnerable since text isn't usually sent in images. The hardcopy version of the image approach intents to achieve robustness with good security.

Table 2. Comparison between the two approaches in terms of capacity, robustness and security.

Approach	Capacity	Robustness	security
Text + softcopy	High, up to infinity in 1st scenario.	Not robust to printing.	Invisible, in code.
Image + softcopy	Very low, due to image overhead.	Robust to printing.	Slightly visible.
Image + hardcopy	Moderate, 1st scenario, block of 2	Robust to printing.	Slightly visible.

5. COMPARISON TO SIMILAR TECHNIQUES

We compare the capacity of our approach to the dots approach [9] and to the Kashidah approach [10]. First, we need to note that in our, as well as the Kashidah approach, hiding a bit is equivalent to inserting a character (a diacritic mark in our case and a Kashidah in the Kashidah method). The dots approach doesn't suffer such increase in size due to hidden message embedding. In fact, the dotted approach can be viewed as an ideal (hence, unpractical) case for the Kashidah method.

Since there are several scenarios to implement all approaches, we count the number of usable characters per approach, independent from the scenario or the secret message to be embedded. For this goal to be realistic, we find utterances in the Corpus of Contemporary Arabic (CCA), by Al-Sulaiti [14, 15]. The corpus is reported to have 842,684 words from 415 diverse texts, mainly from websites. For the diacritic approach, the overhead is easy to estimate. Besides, it needs a diacratized text to experiment on. Hence, we use the not-heavily-diacratized sentence in Figure 7 to extract results.

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 the dots approach, dotted characters may contribute to p while undotted characters may contribute to q. For the Kashidah method, we study two cases: the case of inserting Kashidahs before, and the case of inserting them after, the required character. We count extendible characters before/after dotted characters for p and those before/after undotted characters for q. For both methods, we keep characters with Hamzahs in a separate class r so as to be added to p or q, whichever is more convenient. The last column assumes equiprobability between (p+r) and q. In our case, a diacritic mark can bare a secret zero or a secret one, hence $p = q$ and $r = 0$.

إِنَّ الحَمْدَ لِلهِ. نَحْمَدُهُ وَنَسْتَعِيْنَهُ وَنسْتَغْفِرُهُ.
وَنَعُوْذُ بِاللهِ مِنْ شُرُوْرِ أَنْفُسِنَا وَسَيِّئَاتِ
أَعْمَالِنَا.
مَنْ يَهدِهِ اللهُ فَلا مُضِلَّ لَهُ.
وَمَنْ يُضْلِلْ فَلا هَادِيَ لَهُ.
وأَشْهَدُ أن لا إله إلا الله وَحدَهُ لا شَرِيكَ لَه
وأَشْهَدُ أَنَّ مُحَمَداً عَبْدُهُ ورَسُولُهُ.

Figure 7. Moderately diacratized text to find utterances of bit-baring units in our method.

Table 3. Ratios of the usable characters for hiding both binary levels of the three studied approaches.

Approach	*p*	*q*	*r*	(*p*+*r*+*q*)/2
Dots	0.2764	0.4313	0.0300	0.3689
Kashidah-Before	0.2757	0.4296	0.0298	0.3676
Kashidah-After	0.1880	0.2204	0.0028	0.2056
Diacritics	0.3633	0.3633	0	0.3633

The figures in Table 3 are quite near. As pointed out previously, the dots approach is actually the ideal unpractical case for the Kashidah method. Hence, we discuss our and the Kashidah methods in depth, here. In a first glance, our approach might seem to outperform the Kashidah method for the restrictions on inserting a Kashidah are more than those on inserting a diacritic: Almost every character can bare a diacritic on it. (Although some rare times two diacritics are there, and some other rare times none is put). However, deeper tests reveal an inherent overhead to diacritics: they never come alone; but above/beneath another character. Hence, a somehow stable overhead of 2 bytes per secret-baring position is found in our approach.

The advantage of our work, however, is that each usable character can bare multiple secret bits with 1 character as overhead. Although this same overhead can be claimed in the Kashidah method, it can't really be applied for Kashidah becomes too long and noticeable.

6. CONCLUSION

This paper presents the two text and image approaches to hide information in Arabic diacritics for steganographic use. It presents a variety of scenarios that may achieve up to arbitrary capacities. Sometimes tradeoffs between capacity, security and robustness imply that a particular scenario should be chosen. The overhead of using diacritics was, experimentally, shown very comparable to related works. The advantage of the method, however, is that such overhead decreases if more than one diacritical secret bit is used at once.

7. ACKNOWLEDGEMENTS

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

Special appreciation to all the 2006 second term students of the course COE 509: Applied Cryptosystems - Techniques & Architectures for their valuable initiatives and positive cooperation.

REFERENCES

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

[2] A. Amin, "Off line Arabic character recognition: a survey," *International Conference on Document Analysis & Recognition*, Vol.2, pp. 596 – 599, 18-20 August 1997.

[3] K. Romeo-Pakker, H. Miled, Y. Lecourtier, "A new approach for Latin/Arabic character segmentation," *International Conference on Document Analysis & Recognition*, Vol. 2, pp. 874 – 877, 14-16 August 1995.

[4] F. S. Al-anzi, "Stochastic Models for Automatic Diacritics Generation of Arabic Names," *Journal Computers and the Humanities*, Vol. 38, No. 4, pp. 469-481, November 2004.

[5] Y. A. El-Imam, "Phonetization of Arabic: rules and algorithms," *Computer Speech & Language*, Vol. 18, No. 4, pp. 339-373, October 2004.

[6] G. Abandah, F. Khundakjie, "Issues Concerning Code System for Arabic Letters," *Direst Engineering Sciences Journal*, Vol. 31, No. 1, pp. 77-165, April 2004.

[7] G. Abandah, M. Khedher, Printed and handwritten Arabic optical character recognition –initial study, *A report on research supported by the Higher Council of Science and Technology*, Amman, Jordan, August 2004.

[8] M. S. Khorsheed , "Off-line Arabic character recognition –a review," *Pattern analysis & applications*, Vol. 5, pp. 31–45, 2002.

[9] M.H. Shirali-Shahreza, M.Shirali-Shahreza, "A New Approach to Persian/Arabic Text Steganography," *5th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2006)*, pp. 310-315, Honolulu, HI, USA, 10-12 July 2006.

[10] A. Gutub, M. 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.

[11] M.A. Aabed, S.M. Awaideh, M.E. Abdul-Rahman, A. Gutub, "Arabic diacritics based Steganography," *IEEE International Conference on Signal Processing and Communications (ICSPC 2007)*, pp. 756-759, Dubai, UAE, 24-27 November 2007.

[12] Correll, S. Graphite: an extensible rendering engine for complex writing systems.: *Proceedings of the 17th International Unicode Conference.* San Jose, California, USA, 2000.

[13] N. F. Johnson, S. Jajodia, "Exploring Steganography: Seeing the Unseen," *IEEE Computer*, Vol. 31, No. 2, pp. 26-34, 1998.

[14] L. Al-Sulaiti, Designing and developing a corpus of contemporary Arabic, MS Thesis, The University of Leeds, March 2004.

[15] A. Gutub, L. Ghouti, A. Amin, T. Alkharobi, and M. 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.

[16] F. Khan, A. Gutub, "Message Concealment Techniques using Image based Steganography", *The 4th IEEE GCC Conference and Exhibition,* Gulf International Convention Centre, Manamah, Bahrain, 11-14 November 2007.