Utilizing Diacritic Marks for Arabic Text Steganography

ADNAN A. GUTUB, LAHOUARI M. GHOUTI, YOUSEF S. ELARIAN,
SAMEH M. AWAIDEH, ALEEM K. ALVI

College of Computer Sciences and Engineering;
King Fahd University of Petroleum & Minerals;
Dhahran 31261; Saudi Arabia; Email: gutub@kfupm.edu.sa

ABSTRACT

Arabic diacritic marks represent efficient carriers to hide information into plain text. The ability of diacritics to invisibly superimpose them on each other, when typed multiple times consecutively, empowers them for robust data hiding applications. In this paper, we propose two different algorithms to map secret messages into repeated diacritics in a non-wasteful fashion where the number of extra diacritics is defined in fixed and variable size fashions. Therefore, the size of the outputted text is decided by the encoding flexibility. Both steganographic algorithms are characterized by several advantages over their existing counterparts. Finally, we provide a detailed performance analysis of both algorithms in terms of embedding capacity, robustness and file size measures.

Keywords: Arabic text; capacity; data hiding; diacritic marks; steganography.

INTRODUCTION

Since ancient times people have recognized the importance of information (and communication) secrecy and security. Since then, several approaches have been successfully adopted and used for covert communications and information exchange. *Steganograhy* - or the art of covert writing () - has emerged as the most efficient means for such purposes in hostile moments such as war times. As such, stenography aims at concealing the very existence of the *covert* messages from enemies, attackers and hackers alike which would ensure the secrecy, and hence the security, of these covert messages. The "*plain messages*" (or plaintext in reminiscence of cryptography) are *discreetly* manipulated to seamlessly carry the secret (or covert) messages. Moreover, nowadays with the emergence of the *Internet*, the Global Information Highway, steganography has been the focus of active research to develop novel and innovative techniques to serve such purposes. Figure 1, as represented by Dmitri (2007), gives an outline where a detailed hierarchical classification of steganography is given.

In Figure 1, the adopted classification categorizes the different steganograhic techniques according to the type of the *cover* message. In this paper, we restrict our work to the class of text-based steganography. The linguistic-based techniques exploit the computer-coding techniques to hide the covert messages (Dmitri 2007). Information hiding through the use of signs and symbols belongs to the semagram-based steganography class.

Arabic diacritic marks have been efficiently used for information hiding (Shirali 2006, Aabed *et al.* 2007, Gutub & Fattani 2007, Gutub *et al.* 2008]. In this case, extra instances of some diacritics are inserted in the text to enable the embedding of the covert message. The diacritic multiple insertion is carried out in an invisible fashion where the inserted diacritics are superimposed on each other without affecting their initial positions. The covert information is carried by the multiple instantiations (or insertions) of these diacritics. It is obvious to note the increase in size of the resulting cover message in terms of file size.

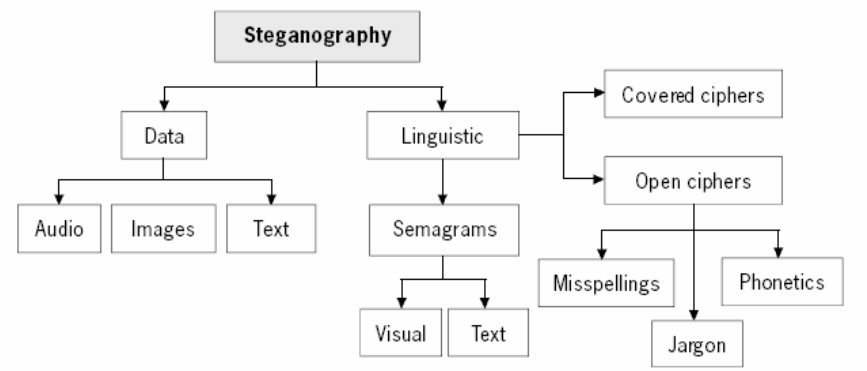


Figure 1: Classification tree of steganography (Dmitri 2007).

The paper is organized as follows. Some background information on Arabic diacritics is given in Section 2. In Section 3, an account of the existing work on Arabic and several non-Arabic scripts' steganography is given. The two proposed steganographic algorithms (fixed-length and variable-length encoding) are detailed in Section 4. The new concept of *wasting property* is established in Section 5. Several issues related to the properties of the cover message are discussed therein. In Section 6, we discuss the proposed techniques used for the mappings of the covert message. Furthermore, a qualitative analysis of the mapping schemes is given. A detailed description and performance analysis of the steganographic prototypes, developed based on the proposed algorithms, are given in Section 7 in a quantitative manner. Finally, Section 8 concludes the paper by summarizing the proposed steganographic algorithms.

BACKGROUND ON ARABIC DIACRITIC MARKS

In Arabic scripts, Arabic diacritics (or consonants) decorate letters in a way to modify their pronunciations (Gutub *et al.* 2008). *Fathah*, *Dammah* and *Kasrah* specify the 'a', 'u' and 'i' short vowels, respectively. *Fathatan*, *Dammatan* and *Kasratan* are the three respective variations with *Tanween* (translated as *nunation*, or *n'ing*). These may occur at the end of some Arabic nouns to make a pleasant *n* sound, just like the *n* in "an" (in contrast to "a") (Tanween - online). The *Sukun* diacritic explicitly indicates the end of a syllable (absence of a vowel) (Sukuun & Shadda – online). Finally, *Shaddah* replaces a double-consonant and introduces a termination (Al-Ghamdi & Zeeshan 2007). These

marks are written over or beneath Arabic characters. It should be noted that for digital representation purposes, they are represented (encoded) as characters (Abandah & Khundakjie 2004). The six diacritics of Arabic script are shown in Figure 2 (Aabed *et al.* 2007) ordered from right to left following the order mentioned above.

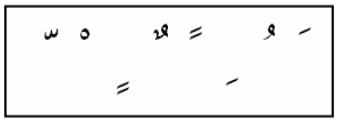


Figure 2: Arabic diacritic marks.

As in most languages, vowels occur frequently in the Arabic language. Particularly in Arabic, the nucleus of every syllable is a vowel (El-Imam 2004). The frequent occurrence of these vowels would imply an abundance of diacritics in any Arabic script. However, it should be noted that the use of diacritics is optional and even not a common practice in modern standard Arabic (Aabed *et al.* 2007). But it should be noted that for Holly scripts, mainly Quran, and in situations where the subtle alterations of diacritics would lead to crucial differences and meanings being distorted. In fact, Arabic readers are trained to deduce diacritic marks (Shirali 2006, Gutub & Fattani 2007). To illustrate the effect of omitting the vowel diacritics, consider reading the English sentence shown in Figure 3 (Sakhr Software Co – available online).

Knowing the facts mentioned previously, diacritics might be abundant provided that appropriate texts, manuscripts and documents are considered. Results of a simple Arabic content query in Google® are summarized in Table 1.

Just to feel the task, read the following English sentence:
"jst t fl th tsk, rd th fllwng nglsh sntnc"

Figure 3: An English sentence and its non-vowelized version - reproduced in part from (Sakhr Software Co).

Table 1: Per-diacritic statistics.

Diacritic	Number of page references by Google	Numbers of diacritics in Musnad Al-Imam Ahmed (available online)
Fathah	1,220,000	1,679,820
Dammah	854,000	367,224
Kasrah	683,000	472,101
Fathatan	1,500,000	18,752
Dammatan	603,000	24,096
Kasratan	1,040,000	50,820
Sukun	643,000	459,566
Shaddah	1,130,000	300,906

The number of retrieved documents for each diacritic mark (shown in Column 1) independently is summarized in the second column of Table 1. For comparison purposes, the individual frequencies in Musnad Al-Imam Ahmed (available online) are presented in the third column of Table 1. We used

Musnad Al-Imam Ahmed because it is well known as one of the largest religious books available online. It is quite interesting to notice the abundance of these diacritics in both types of documents which supports our opinion about the appropriateness of the considered documents for a successful implementation of the proposed steganographic schemes.

RELATED WORK

Following the successful applications of steganography in English, other Latin and non-Latin texts, information hiding for Arabic text begun to emerge. Because of the peculiar nature of Arabic texts, three inherent properties of Arabic script have been considered for steganography: *dots*, *connect-ability* and *diacritics*. Shirali (2006) presented a new method proposed for hiding information in Persian and Arabic Unicode texts. In their work (Shirali 2006), two special characters, the zero width non joiner (ZWNJ) and zero width joiner (ZWJ) characters, respectively, are used to hide information in Persian and Arabic Unicode text documents. It is known that in Persian and Arabic texts, most letters are normally connected together in words. However, the ZWNJ character prevents the Persian and Arabic letters from joining and the ZWJ character forces them to join together. More specifically, Shirali (2006) hide secret information in the points' location within the pointed letters. First, the hidden information is looked at as binary with the first several bits (i.e. for example, 20 bits) to indicate the length of the hidden bits to be stored. Then, the cover medium text is scanned. Whenever a pointed letter is detected, its point location may be affected by the hidden info bit. If the value of the hidden bit is one, then the letter point is slightly shifted up; otherwise, the point location remains unaffected. This point shifting process is shown in Figure 4 (Shirali 2006) for the Arabic letter '*Fa*'. It should be noted that in order to create more perceptual transparency, the points of the unaltered letters are randomly changed.

Figure 4: Point shift-up of Arabic letter '*Fa*'(Shirali 2006).

It should be noted that this method can hide one bit in each Persian/Arabic letter which would enable a high capacity for text hiding. Also, this method does not yield visible alterations on the cover text which would achieve a desirable perceptual transparency. However, the algorithm, proposed by Shirali (2006), requires special fonts to be installed and gives different codes to the same Persian/Arabic letter depending on the secret bit being hidden. Therefore, it suffers from being of practical standard use (Gutub *et al.* 2007).

Samphaiboon and Dailey (2008) propose a steganographic scheme for Thai plain text documents. In the Thai TIS-620 system, the standard Thai character set, vowel, diacritical, and tonal symbols are *redundantly* composed in a very special way (Samphaiboon & Dailey 2008). Samphaiboon and Dailey (2008) exploit such redundancies as the compound characters combining vowel and diacritical symbols

in the TIS-620 system. Their proposed scheme is characterized by unnoticeable textual modifications and achieves an embedding capacity of 0.22% (Samphaiboon & Dailey 2008). Thai characters can be placed vertically at four levels: 1) top level; 2) above level; 3) base-line level; and 4) below level, as shown in Figure 5.

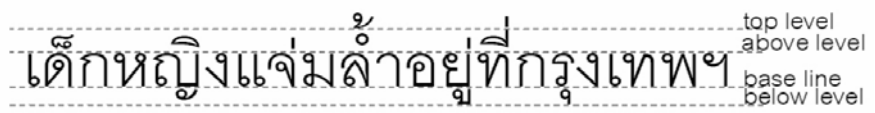


Figure 5: Four levels of vertical placement of Thai symbols (Samphaiboon & Dailey 2008).

This scheme is a blind one since the original cover text is not required for decoding. Moreover, this embedding scheme allows 2.2 bytes of covert text per kilobyte of cover text on average. However, it should be noted that this scheme applies only to Thai text and cannot be generalized easily.

Amano and Misaki (1999) propose a feature calibration scheme for steganographic uses in Japanese texts by embedding and detecting watermarks in document images. A calibration method is proposed to use the difference between two features extracted from two sets of partitions arranged symmetrically. In this method, the average width of character strokes is used as a feature. Figure 6 shows a typical two-set partitioning as proposed by Amano and Misaki (1999). Figure 7 shows the plain text in (a), the cover text after 1-bit embedding in (b), and the cover text after 0-bit embedding (c). It is clear from Figure 7 (b and c) that the proposed partitioning layout is able to counterbalance the variations of the stroke width feature with respect to the font face, font size, contents printed, and re-digitization.

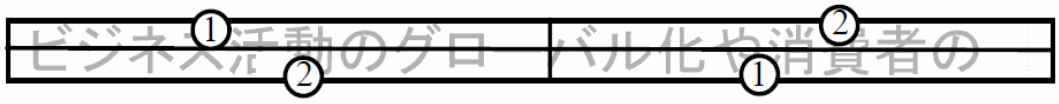


Figure 6: Partitioning and Grouping into two sets (Amano & Misaki 1999).

インターネットやCD-ROMの普及によってディジタルコンテンツを容易に配布,流通させることが

可能になった.このことは,インターネットを利用した情報配信サービスなどの新たなアプリケー

(a)

インターネットやCD-ROMの普及によってディジタルコンテンツを容易に配布,流通させることが

可能になった.このことは,インターネットを利用した情報配信サービスなどの新たなアプリケー

(b)

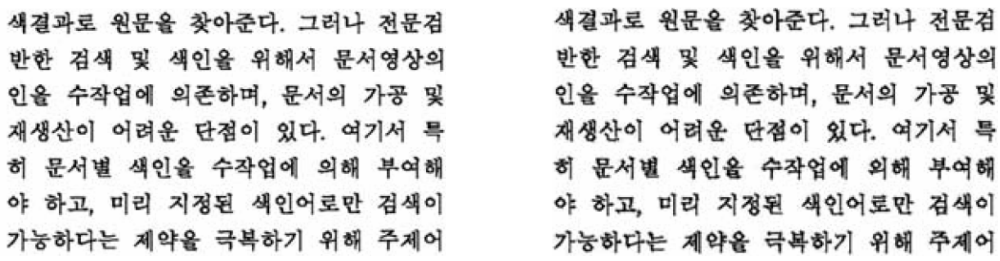
インターネットやCD-ROMの普及によってディジタルコンテンツを容易に配布,流通させることが

可能になった.このことは,インターネットを利用した情報配信サービスなどの新たなアプリケー

(c)

Figure 7: Steganograhy example in Japanese text. (a) Plain text. (b) Cover text with 1-bit embedded. (c) Cover text with 0-bit embedded (Amano & Misaki 1999).

Kim and Oh (2004) propose a novel algorithm for watermarking grayscale Korean text document images. The algorithm embeds the watermark signals through edge direction histograms. The algorithm exploits the sub-image consistency concept. Text sub-images are assumed to have similar-shaped edge direction histograms. They showed the assumption validity over a wide range of document images. Each (or more) text block is designated as a *mother block*. A reference edge direction histogram is provided by the unaltered mother block(s). Then, each of the remaining blocks (referred as a child block) is modified by the embedding process to have an edge direction histogram different from the mother block(s) depending on the bit value (0 or 1). Figure 8 shows an example of plain and cover Korean texts images.



(a) **(b)**

Figure 8: Steganograhy example in Korean text. (a) Plain text. (b) Cover text (Kim and Oh 2004).

PROPOSED ALGORITHMS

In an attempt to alleviate the limitations of Arabic-based steganographic algorithms (Shirali 2006), we will propose the use of the *connectivity property*. Gutub & Fattani (2007) and Gutub *et al.* (2007) presented methods for utilizing the Kashida's for steganography. Kashida (the Arabic redundant extension character used for justifying or beautifying a text) before/after dotted/un-dotted characters has overcome the need for requiring new fonts being installed. Gutub & Fattani (2007) proposed a steganographic scheme that hides secret information bits within Arabic letters by exploiting their inherited points. To locate a specific letter carrying the secret bits, the pointed letters and the redundant Arabic extension character (Kashida) are considered. A one-bit is secretly hidden in a pointed letter with extension, and an un-pointed letter with extension is used to host a zero-bit. Note that the letter extension doesn't have any effect on the content writing nor meaning. It has a standard character hexadecimal code: 0640 in the Unicode system (Gutub *et al.* 2007). In fact, this Arabic extension character in electronic typing is considered a redundant character only for arrangement and format purposes. The embedding scheme is illustrated in the example shown in Figure 9.

Secret bits	110010
Cover-text	من حسن اسلام المرء تركه مالا يعنيه
Steganographic text	من حسن اسلام المرء تركه مالا يعنيه 1 1 0 0 1 0

Figure 9: Steganography example adding extensions after letters.

In Figure 9, a secret information code 110010 will be discreetly embedded into a plain Arabic text. The least significant bits are considered first. The first secret bit '0' will be hidden in an un-pointed letter. The cover text is scanned from right to left due to Arabic regular writing/reading direction. The first un-pointed letter in the cover-text is the '*meem*' letter. This '*meem*' will carry the first secret bit '0' noted by adding an extension character after it. The second secret bit, '1', cannot be hidden in the second letter of the cover text, '*noon*' because this letter position cannot allow extension (based on Arabic language writing rules). The next possible pointed letter to be extended is '*ta*'. Note that a pointed letter '*noon*' before '*ta*' is not utilized due to its unfeasibility to add an extension character after it. The technique, proposed by Gutub & Fattani (2007) can be applied, in a straightforward fashion, to texts having similar approaches of pointing and extending letters such as the Persian and Urdu texts. However, a reduction in hiding capacity occurs due to the restricting rules on the usage of Kashida and due to the *wasting property* described in the following Section.

Diacritic marks in Arabic text have been proposed for steganography by Aabed *et al.* (2007). Their approach makes usage of eight different diacritical symbols in Arabic to hide binary bits in the original cover media. The embedded information is then extracted by reading the diacritics from the document and translating it back to a binary representation. Fully diacritized Arabic texts are used as cover media. Then, the first bit of the secret data is compared with the first diacritic in the cover media. If, for example, the first secret bit is one and the first diacritic is a '*fatha*', the diacritic is kept on the cover media and an index for both the embedded text and the cover media is incremented. If, however, the first diacritic was not a '*fatha*' then it is removed from the cover media and the index for the cover media is incremented to explore the next diacritic. The same approach is repeated until the next '*fatha*' is found. A secret zero-bit is embedded in the same way except it will use the remaining seven diacritics other than '*fatha*'. The embedding process is illustrated in Figures 10-a and 10-b.

حَدَّثَنَا سُفْيَانُ عَنْ يَحْيَى عَنْ مُحَمَّدِ بْنِ إِبْرَاهِيمَ التَّيْمِيِّ عَنْ عَلْقَمَةَ بْنِ وَقَّاصٍ قَالَ سَمِعْتُ عُمَرَ رَضِيَ اللَّهُ عَنْهُ يَقُولُ
سَمِعْتُ رَسُولَ اللَّهِ صَلَّى اللَّهُ عَلَيْهِ وَسَلَّمَ يَقُولُ إِنَّمَا الْأَعْمَالُ بِالنِّيَّةِ وَلِكُلِّ امْرِئٍ مَا نَوَى فَمَنْ كَانَتْ هِجْرَتُهُ إِلَى اللَّهِ
عَزَّ وَجَلَّ فَهِجْرَتُهُ إِلَى مَا هَاجَرَ إِلَيْهِ وَمَنْ كَانَتْ هِجْرَتُهُ لِدُنْيَا يُصِيبُهَا أَوْ امْرَأَةٍ يَنْكِحُهَا فَهِجْرَتُهُ إِلَى مَا هَاجَرَ إِلَيْهِ

Figure 10-a: Example of a Standard Arabic text with full diacritics placement.

حَدَثنا سُفْيَان عَن يَحْيى عنْ محمَدِ بْن إبرَاهيمَ التيْمي عَنْ علْقمة بْن وقَاص قالَ سمِعْت عمَر رضِيَ اللهُ عَنْهُ يَقول
سَمعت رَسولَ الله صَلَى اللهُ عَليه وَسلم يقُولُ إنما الْأعمَال بالنيَة وَلِكل امرئ مَا نوى فمنْ كانتْ هجرَته إلى الله
عَزّ وجلّ فهجرَته إلى مَا هَاجَر إليْهِ ومنْ كانتْ هجرَته لدنيا يصيبهَا أوْ امرَأةٍ ينْكحهَا فهجْرَتهُ إلى ما هاجر إليه

Figure 10-b: Example of using Standard Arabic diacritics to encode a pseudo-random sequence.

Figure 10-a shows a standard Arabic text full of diacritics (the plain text in this case). The secret message consists of a pseudo-random sequence represented in hexadecimal format as:

E7 - 30 - E9 - 1C - A4 - FC - B8 - B9 - AF - 1F - 0B - D9 – 22

The resulting cover media (or text) is illustrated in Figure 10-b. Notice the clear differences in diacritics between the two texts illustrated in Figures 10-a and 10-b, respectively. These differences indicate that

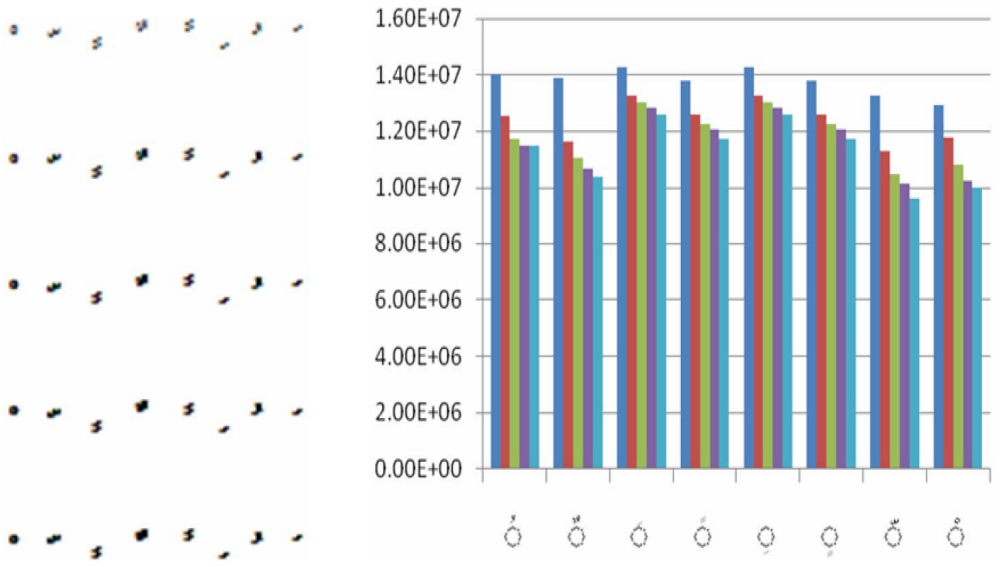
the secret message has been effectively and discreetly hidden in the cover text. It should be noted that the same cover media can be reused more than once if needed. However, unless a method is used to reinsert the removed diacritics, hiding capacity will decrease drastically every time a new message is embedded into the cover text. Zitouni *et al.* (2006) propose the use of maximum entropy to restore the missing Arabic diacritics. Another approach using hidden Markov models (HMMs) to restore the vowels is proposed by Gal (2002). Diacritic restoration can also be done manually if necessary, and in any case, a new cover media, already diacritized, can be used instead. Borrowing ideas from the computer representation (display/print) of Arabic diacritic marks; Gutub *et al.* (2008) propose two different approaches: 1) textual; and 2) image. In the first approach, the whole secret message can be hidden in a single diacritic mark by hitting/typing (or generating) a number of extra-diacritic keystrokes equal to the binary number representing the secret message. For example, to hide the binary string $(110001)_b = 49_d$, the repetition rate is $n = 50$. This number seems quite high for such approaches. One solution could consist of performing the same algorithm on a block of limited number of bits. For illustration, let's consider the same secret message of $(110001)_b$, then we will 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 ($1=(01)_b$). Moreover, Gutub *et al.* (2008) propose a variant for the same approach similar to the run-length encoding (RLE) algorithm used in data compression. In the RLE variant, the first diacritic mark is repeated in the cover 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 are repeated according to the number of zeros. For illustration, the RLE variant to embed the same secret message would imply repeating the first diacritic 2 times (2 = number of 1's in the sequence $(11)_b$); the second one 3 times (3 = number of 0's in the sequence $(000)_b$); and the third one 1 time. Table 2 summarizes the results of information hiding and encodings of the binary sequence 110001 using the two variants (block and RLE encoding) of the textual approach.

Table 2: Embedding results of the binary sequence 110001 using the two variants of the textual approach.

Scenario	**Extra diacritics**
Textual approach: Variant 1 (All stream)	**49**
Textual approach: Variant 1 (Block Size = 2)	3 + 0 + 1 = **4**
Textual approach: Variant 2 (RLE Encoding)	(2-1) + (3-1) + (1-1) = **3**

In the second approach, image-based, one of the fonts that slightly darken multiple occurrences of diacritics is selected. Figure 11 (a) shows the darkening of the black level of the diacritics by multiple instances. The brightness levels of such diacritics are quantized by adding their 24 color-bits representation as a concatenated sequence. Notice that the less the brightness level, the more the darkness is. It should be noted that image conversion is necessary in this approach to withstand the printing effects. This conversion is necessary given the major differences between the printing and display technologies used in rendering such Arabic complex characters (Correll 2000). It is interesting to note that the printing process doesn't darken extra diacritic instances of text, even when they are by the

display process. This non-uniformity in the effects reduces the possible number of repetition of a diacritic to such a level that will allow the diacritic simultaneously withstand both printing and scanning processes. This limitation favors the use of the textual approach over the image counterpart. More specifically, the block-based variant with a small block size (say 2). In addition to this, the size of the image file containing the cover text is usually much larger than that of the text-based representation. Finally, it is worth mentioning that to transform the text into an image format, such as the portable document format (PDF), which will prevent from accidental or intentional font manipulation, would hinder the security of the embedded information.



(a) **(b)**

Figure 11: (a) Image representation of Arabic diacritics. (b) Quantization of brightness levels of selected diacritics by adding the 24 color-bits.

Table 3 gives a summary of the capacity, robustness and security of the proposed approaches. It should be noted that we have considered two different cover media for the image-based approach. In the first instance, a softcopy of the document image is used as a cover media while a printed version of the document is used in the second one. Although the textual approach is not, generally, robust to printing, it

Table 3: Comparison of the proposed approaches in terms of capacity, robustness and security.

Proposed Approach	Capacity	Robustness	Security
Text + Softcopy	High, up to infinity using the first variant	Not robust to printing	Invisible, in code
Image + Softcopy	Very low, due to image overhead	Not robust to printing	Slightly visible
Image + Hardcopy	Moderate in first variant (Block size of 2)	Robust to printing	Slightly visible

is capable of achieving arbitrarily high embedding/hiding capacities. However, the security of the embedded information can be jeopardized when the cover media is excessively used for embedding and,

therefore, the file size gets increasingly larger. On the other hand, the image approach is, to some extent, robust to printing. For comparison purposes, we have considered, in our work, the softcopy version. It has a very low embedding capacity. Also, its security is vulnerable since text isn't usually sent in images. The hardcopy version of the image approach aims to achieve robustness with good security.

It should be noted that the redundant nature of using such diacritics has served hiding information by the selective omission of some instances. Again, this proposal suffered from the *wasting property*. To improve the capacity, however, a multi-instance approach, with *non-wasting* scenarios, was proposed by Aabed *et al.* (2007). We will explore the wasting property in the next Section.

EXPLOITING WASTING PROPERTY TO IMPROVE STEGANOGRAPHY

In this Section, we will first define the *wasting property* noticed in some steganographic algorithms. The *wasting property* manifests itself in algorithms that use different kinds of cover media for different types of secret messages to be hidden. For example, the original Kashida method can hide a one-bit after a dotted Arabic character (after pointed letter); but needs an un-dotted character to hide a zero-bit. This would mean an excess waste of the cover media whenever the latter is not appropriate for the secret bit to be embedded. Figure 12 shows a regular Kashida-based steganographic process. An improved version that exploits the wasting property is shown in Figure 13. In this case, the potential recipient segments (cover blocks) of the Arabic cover are elongated and highlighted (marked underlined). Then, the secret binary sequence *11001* will be embedded in the shown cover text in Figure 12. The wasted cover text blocks are marked with double underlined (highlighted in red) while the ones being used are marked with single underlined (highlighted in green). It is worth noting that this sentence is only able to bare 3 (out of 5) secret bits.

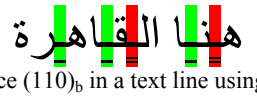


Figure 12: Embedding the secret sequence $(110)_b$ in a text line using the Kashida method (wasting property).

A non-wasting version of the Kashida method would remove the dotted letter consideration from the procedure. We simply insert a Kashida after an extendible character to represent a one, and omit a (otherwise possible) Kashida for a secret zero, i.e. regardless of the character's possession of dots. All 5 extendible characters in the text line are now capable of carrying an arbitrary bit: zero or one. The result of embedding the same 5-bit binary sequence is depicted in Figure 13 with the "to-remain" Kashida's as double underlined (highlighted in yellow) and the "to-omit" ones as single underlined (highlighted in blue). Usually, the maximum achievable capacity is obtained when using a *non-wasting* algorithm.

Figure 13: Embedding the secret sequence 11001 in a text line using the *non-wasting* Kashida method.

In this way, we principally classify *non-wasting* steganographic scheme for any scheme whose embedding capacity is simply a function of the *number* of secret-bit carriers and not of their actual sequence. With this in mind, the proposed diacritics algorithm, detailed by Aabed *et al.* (2007), is a non-wasting enhancement to that outlined by Gutub *et al.* (2008).

MAPPING THE HIDDEN MESSAGE

The diacritics algorithm presented by Aabed *et al.* (2007) embeds secret messages in multiple instances of invisible Arabic cover text with diacritics. Because multiple-instances are used in the hiding process, secret binary bits must be decoded into the (integer) number of repetitions of a diacritic mark. We will provide a comparison between two possible decoding scenarios: the *fixed-size* decoding scenario and the *content-based* decoding scenario.

The former decoding scheme parses a stream of binary bits into blocks of fixed-size. The size of the block is system-dependent. Decoding a block of bits into an integer is straightforward given that a coding system is defined. Here, the basic Binary Code System is assumed. The latter scheme, variable-size content-based, parses a stream binary data into an integer number regardless of the number of bits they might have. The basic RLE idea is used to perform the mapping. It maps all consecutive *ones* and all consecutive *zeros* into the length of their corresponding run.

Table 4 shows the encodings of the binary value *11000101* according to the two scenarios. Column 2 shows how the original sequence (i.e. the binary value *11000101* upper row in the mapping Column – typed in red) is mapped into integers (beneath the corresponding sets of bits). The total number of blocks that are used in the sequence mapping is shown in Column 3. It defines the minimum number of diacritic marks that need to be found in the cover message to carry the secret message. The number of extra diacritics to be added to the cover message to enable secret message embedding is given in the last column.

Table 4: The encodings of the binary value 11000101 according to the two scenarios.

Scenario	Mapping	Needed Diacritics	Extra Diacritics
Fixed-size = 1	1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1	8	1 + 1 + 0 + 0 + 0 + 1 + 0 + 1 = 4
Fixed-size = 2	1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 3 | 0 | 1 | 1	4	3 + 0 + 1 +1 = 5
Fixed-size = 4	1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 12 | 5	2	12 + 5 =13
Content-based	1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 2 | 3 | 1 | 1 | 1	5	(2-1)+(3-1)+(1-1)+(1-1)+(1-1) = 8 - 5 =3

Comparing our two decoding schemes, we notice that the number of needed diacritics in the fixed-size

scenario is determined solely by the size of the secret message regardless of the actual data it carries. The size of the combined message, however, cannot be predicted until decoding of the actual data is performed. On the other hand, the number of required diacritics in the content-based scheme cannot be predicted without examining the content of the secret message to find out the number of runs. Moreover, it is interesting to notice that the size of the final combined output can be determined solely by inspecting the size of the secret message. These interesting findings are summarized in Table 5.

Finally, for an optimized use of both decoding schemes, the fixed-size method should be considered when the selection of the cover message is done on a quick basis. However, content-based decoding should be considered when the size of the combined message matters most. Therefore, whenever the size is the only determining factor, the related property is easier to compute in advance.

Table 5: Factors determining the sizes of the cover and the combined messages for the two decoding schemes.

Scenario	Needed Diacritics	Extra Diacritics
Fixed-size	only size of secret data	complete secret data sequence
Content-based	complete secret data sequence	only size of secret data

PERFORMANCE RESULTS

A prototype steganographic system is implemented to carry out the performance evaluation of the proposed algorithms. In the first stage of the evaluation, a dataset to be used as a covert text messages must be appropriately selected. For this purpose, an Arabic text corpus, from University of Leeds (Al-Sulaiti 2004), was used. Then, a set of plain text content was selected to be used as a cover message. One of the largest e-books available in Arabic literature, namely Musnad Al-Imam Ahmed (available online), was used for such purposes. Because of its large size, only its half was considered sufficient for our testing purposes. The selected content contains 7,305,490 characters and 1,037,265 words. Once the data sets were selected, hidden messages were encrypted and hidden in the cover text. Then, a performance analysis was carried out. It is worth mentioning that the encryption stage is not an integral of the proposed steganographic algorithms but it aims at increasing the security level of the covert messages only. The encryption was carried out using a publicly available library in Crypto++ Reference Manual (available online). The encode/decode library supports most of the basic schemes in modern cryptography literature such as the Elliptic Curve Cryptography (ECC) defined over GF(p) (Koblitz 1987) with private and public keys randomly generated. At the output of the ECC block, we noticed that the RLE encoding is giving comparable efficiency to the 2-bits block. Further investigation on the resulting data revealed that the ECC block was generating balanced data.

Tables 6 and 7 give a summary of the utterances of the different patterns used. Both tables clearly indicate that the data is well balanced in this regard. For the extraction stage, the hidden message is first extracted from each output file by counting diacritics and translating them into bytes. The RLE block requires an extra step for flattening each run into its corresponding bits. Then, the hidden messages are decoded using the Crypto++ library, and the output is compared with the original hidden files.

For a realistic testing procedure, excerpts of corpus (i.e. a representative sample of the naturally used language) are selected as the original secret messages. The corpus used is the Corpus of Contemporary Arabic (CCA) from Al-Sulaiti (2004) PhD thesis, which is reported to have 842,684 words from 415 diverse texts, mainly from websites. Furthermore, we encrypt these messages before hiding them as mentioned previously. Table 8 shows sizes of the plain text and its corresponding encrypted message. Note in Table 8, the ECC encryption slightly increases the size of the message. Our cover messages are also intended to be natural by choosing them from a well-known Arabic diacritized reference of Musnad Al-Imam Ahmed (available online, Abandah & Khundakjie 2004). The book is reported to have a diacritic ratio of about 50% (Aabed *et al.* 2007).

Table 6: Utterances of patterns 01 to 11 in the 10 files used.

01	**10**	**11**
1059	1098	1090
2138	2112	2055
3066	3112	3187
4128	4216	4160
5343	5208	5171
6197	6138	6266
7244	7185	7303
8065	8283	8336
9050	9213	9508
10459	10303	10349

Table 7: Utterances of patterns 0001 to 1111 in the 10 files used.

0001	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
139	152	161	137	126	152	133	116	129	123	151	132	117	152	122
298	257	251	271	267	284	237	253	276	262	255	267	238	263	272
406	408	403	377	378	362	376	410	396	363	406	401	393	404	402
508	508	527	510	530	471	530	536	534	549	515	475	515	554	522
640	637	642	642	701	662	684	647	657	652	640	626	656	661	631
764	737	776	795	775	780	772	786	755	756	780	789	781	788	790
897	854	920	927	882	913	919	915	902	874	909	851	922	944	919
1031	1096	1080	1062	986	988	984	992	1012	1044	1101	1039	1016	1006	1055
1149	1130	1283	1182	1104	1123	1082	1169	1150	1156	1166	1240	1156	1163	1209
1323	1269	1319	1252	1341	1314	1267	1283	1316	1256	1276	1275	1305	1333	1287

Based on our findings and results, we can qualitatively conclude that the choice of the encoding algorithm affects capacity in two measures:

1. The amount of embedded bits in a predefined cover message (secret bits/cover byte) or the amount of cover message needed for a given secret message (cover bytes/secret bit).
2. The size of the combined message needed for a hidden message; either compared to the hidden message (combined bytes/secret bit), or recorded as a percent increase to the size of the cover message (bytes/byte).

The effects of each scheme on the first and second capacity measures are summarized in Figures 14 and 15, respectively. Further, to see the effect of the block-size parameter within the fixed-size scenario, we plot the results of three variations of block sizes, namely, 1, 2, and 4. Finally, we show the average percent increase in the size of the cover message in Table 9, for this ratio results to be independent from the input message size.

Table 8: Sizes of the indirect plain and the direct cipher inputs to the system.

Original CCA-Text		ECC Encrypted Text
(KB)	**(bits)**	**(bits)**
1	8,192	8,736
2	16,384	16,912
3	24,576	25,128
4	32,768	33,296
5	40,960	41,488
6	49,152	49,680
7	57,344	57,872
8	65,536	66,064
9	73,728	74,256
10	81,920	82,448

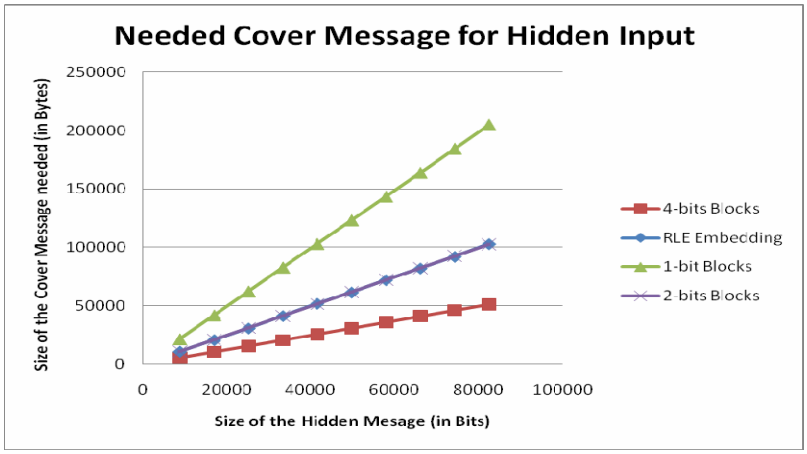


Figure 14: Needed bytes of cover message per bits of secret message.

It is to be observed as expected that the results of both figures, i.e. Figures 14 and 15, are linear. Also as anticipated, the slope of the 4-bit blocks scenario in Figure 14 is half of that of the 2-bit blocks scenario, which in turn is half of the 1-bit blocks scenario slope. Notice the very close values of the content-based (RLE) curve and the 2-bits block curve (They appear superimposed in the graph). Recall that the relation between the needed cover and the secret messages can be determined solely by knowing the size of the secret message in case of the block scenarios.

As for Figure 15, the content-based scenario slightly out performs the 2-bits block in terms of the final size. Both of them are better than the 4-bits scenario, which might suffer from high overhead if the block

has leftmost *ones*. The 1-bit block scenario barriers the worst overhead for each diacritic needs a minimum of 2 bytes of overhead (the diacritic and the character that bares it).

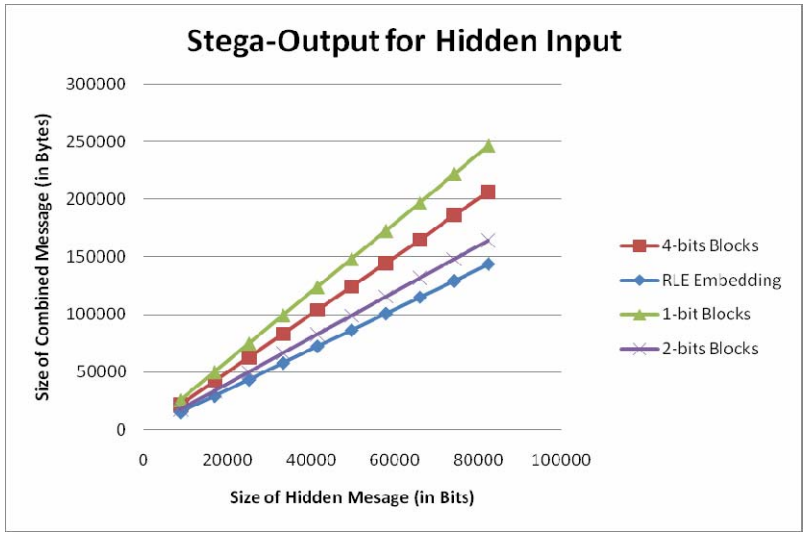


Figure 15: Bytes of combined message per bits of secret message.

Finally, if the focus of the steganographic implementation is to have minimal variation from the original text, then the 1-block scenario would function best, as depicted by Table 9. Here the 4-bit size deteriorates dramatically.

Table 9: Average percent increase from the size of the cover message to the size of the combined message

Scenario	Average Percent Increase in Size
Content-based	40.36 %
Fixed-size = 1	20.14 %
Fixed-size = 2	60.41 %
Fixed-size = 4	300.69 %

CONCLUSION

Two steganographic algorithms for Arabic text were proposed. The algorithms were developed based on a new concept of wasting/non-wasting property of the Arabic diacritics. The first algorithm, the fixed-size block parsing, parses a stream of binary bits into cover blocks of fixed size. In the second algorithm, the variable-size content-based, binary data is parsed into an integer number of blocks regardless of the number of bits they carry. The proposed algorithms are characterized by different properties and, thus, are suited for different application types and steganography requirements (i.e. robustness, capacity and file size). Unlike the content-based algorithm, the fixed-size one allows straightforward computation of the needed amount of cover text but cannot directly predict the output file size. Reported results indicate that a 4-bit block size is preferred for a minimal size of the resulting message file. On the other hand, for minimal overhead, the content-based algorithm is preferred. In terms of percent increase in cover size, 1-bit block acts most secretly.

ACKNOWLEDGEMENT

Thanks to King Fahd University of Petroleum and Minerals (KFUPM) for supporting this research.

REFERENCES

Aabed, M., Awaideh, S., Elshafei, A., & Gutub, A., 2007. Arabic Diacritics Based Steganography. Proceedings of the IEEE International Conference on Signal Processing and Communications (ICSPC 2007), Dubai, UAE, 24-27 Nov. 2007, Pp. 756-759.

Abandah, G., & Khundakjie, F., 2004. Issues Concerning Code System for Arabic Letters. Dirasat Engineering Sciences Journal, **31(1):**165-177.

Al-Ghamdi, M., & Zeeshan, M., 2007. KACST Arabic Diacritizer. Proceedings of the First International Symposium on Computers and Arabic Language, 25-28 Mar. 2007.

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

Amano, T., & Misaki, D., 1999. A feature calibration method for watermarking of document images. Proceedings of the Fifth International Conference on Document Analysis and Recognition (ICDAR 1999), **2:**91-94, Bangalore, India.

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

Crypto++ Reference Manual. Available online: http://cryptopp.sourceforge.net/docs/ref/.

Dmitri, V., 2007. Digital Security and Privacy for Human Rights Defenders. The International Foundation for Human Right Defenders, Feb. 2007 Manual.

El-Imam, Y., 2004. Phonetization of Arabic: Rules and Algorithms. Computer Speech and Language, **18(4):**339-373.

Gal, Y., 2002. An HMM Approach to Vowel Restoration in Arabic and Hebrew. ACL-02 Workshop on Computational Approaches to Semitic Languages.

Gutub, A., Elarian, Y., Awaideh, S., & Alvi, A., 2008. Arabic Text Steganography Using Multiple Diacritics. Proceedings of the 5th IEEE International Workshop on Signal Processing and its Applications (WoSPA 2008), Sharjah, UAE. 18-20 Mar. 2008.

Gutub, A., & Fattani, M., 2007. A Novel Arabic Text Steganography Method Using Letter Points and Extensions. Proceedings of the International Conference on Computer, Information and Systems Science and Engineering (ICCISSE), Vienna, Austria, 25-27 May 2007, Pp. 28-31.

Gutub, A., Ghouti, L., Amin, A., Alkharobi, T., & Ibrahim, M.K., 2007. Utilizing Extension Character 'Kashida' With Pointed Letters For Arabic Text Digital Watermarking. Proceedings of the International Conference on Security and Cryptography (SECRYPT), Barcelona, Spain, 28-31 July 2007.

Kim, Y.-W., & Oh, I.-S., 2004. Watermarking Text Document Images using Edge Direction Histograms. Pattern Recognition Letters, **25:**1243-1251.

Koblitz, N., 1987. Elliptic curve cryptosystems. Mathematics of Computation **48:**203–209.

Musnad Al-Imam Ahmad. available on-line: http://www.islamport.com/b/3/alhadeeth/motoon/%df%ca%c8%20%c7%e1%e3%ca%e6%e4/%e3%d3%e4%cf%20%c3%cd%e3%cf%20%c8%e4%20%cd%e4%c8%e1/

Sakhr Software Co., Challenges in Arabic NLP. Arabic Language Resources (LR) and Evaluation: Status and Prospects, Pp. 1126-1130.

Samphaiboon, N., & Dailey, M., 2008. Steganography in Thai Text. Proceedings of the fifth annual international conference organized by Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI) Association (ECTI-CON 2008), Thailand.

Shirali-Shahreza, M.H., & Shirali-Shahreza, M., 2006. 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, 10-12 July 2006, Pp. 310-315.

Sukuun & Shadda. Available on-line: http://en.wikibooks.org/wiki/Arabic/LearnRW/Sukuun_and_Shadda.

Tanween. Available on-line: http://en.wikibooks.org/wiki/Arabic/LearnRW/Tanween.

Zitouni, I., Sorensen, J.S., & Sarikaya, R., 2006. Maximum entropy based restoration of Arabic diacritics. Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL, Sydney, Australia, Pp. 577-584.