The ISC Int'l Journal of Information Security

ISeCure

July 2010, Volume 2, Number 2 (pp. 109–120)

http://www.isecure-journal.org

# High Capacity Steganography Tool for Arabic Text Using 'Kashida'

Adnan Abdul-Aziz Gutub [a,*] and Ahmed Ali Al-Nazer [b]

[a] *College of Computer, Umm Al-Qura University, Makkah, Saudi Arabia.*

[b] *Saudi Aramco, Dhahran, Saudi Arabia.*

ARTICLE INFO.

*Article history:*

**Received:** 29 November 2009

**Revised:** 9 June 2010

**Accepted:** 16 June 2010

**Published Online:** 13 July 2010

*Keywords:*

Arabic E-Text, Text Steganography, Text Watermarking, Text Hiding, Kashida, Feature Coding

ABSTRACT

Steganography is the ability to hide secret information in a cover-media such as sound, pictures and text. A new approach is proposed to hide a secret into Arabic text cover media using "Kashida", an Arabic extension character. The proposed approach is an attempt to maximize the use of "Kashida" to hide more information in Arabic text cover-media. To approach this, some algorithms have been designed and implemented in a system, called MSCUKAT (Maximizing Steganography Capacity Using "Kashida" in Arabic Text). The improvements of this attempt include increasing the capacity of cover media to hide more secret information, reducing the file size increase after hiding the secret and enhancing the security of the encoded cover media. This proposed work has been tested outperforming previous work showing promising results.



## 1 Introduction

Steganography is defined as in [1] "the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, even realizes there is a hidden message". Steganography works as we hide information in un-used and redundant bits in any cover media such as pictures, sound and text.

Hiding secret information in text is more challenging. First, text documents have relatively little redundant information. Second, the structure of text documents is almost identical to their look and hence any change may be visible. Nevertheless, using text is preferred over other media because it needs less memory to save, is easier to transfer over the network and more efficient and cost-saving in printing [2, 3].

Text steganography as it is hiding a secret inside text has dependencies on the language used as cover media. Different human being languages have different characteristics and properties. In Arabic language, there are 28 different characters. Arabic characters are joined when writing words contain more than one character. Depending on the joined characters, an extension character "Kashida" may be embedded between two Arabic characters.

There are two uses of the extension character "Kashida" in Arabic text. One is to decorate the Arabic text format so that it looks better and more convenient. This use is important especially in the titles of the documents. The second use is to justify the Arabic writings within lines, similar to English where spaces are used for justifying the text in lines. The advantages of using "Kashida" in Arabic text to either format it or justify the lines will not affect the text contents and meaning [4, 5].

In this paper, an improved approach is proposed to maximize the use of the Arabic extension character, Kashida, between joined characters in Arabic text cover media. The idea of this approach is to embed "Kashida" wherever possible after any Arabic letter regardless of it being dotted or not dotted; as Arabic

* Corresponding author.

Email addresses: aagutub@uqu.edu.sa (A. A. Gutub), ahmed.nazer@aramco.com (A. A. Al-Nazer).

ISSN: 2008-2045 

letters are categorized to two groups: dotted and non-dotted letters. The approach initiative is originally presented by us in [5]; where it is improved here and compared to the earlier work presented in [4].

The rest of this paper is organized as follows. Section 2 presents different approaches related to Arabic text steganography. Section 3 starts by presenting a background and study of Arabic characters properties. After that, it describes the details of the proposed approach including the idea, algorithms and implementation. In Section 4, we highlight the improvements of this work over other approaches. Section 5 afterwards, presents a comprehensive comparison between the proposed approach and other approaches including the proposed approach testing results. Then, a new secured MSCUKAT approach is detailed in Section 6. Section 7 suggests ten items to be future work ideas to be considered related to this effort. Finally, Section 8 summarizes the findings in a brief conclusion.

## 2 Related Work

In [2], the paper proposes a new approach to Text Steganography in Persian and Arabic texts. This approach uses one of the characteristics of Persian and Arabic languages which are the rich existence of points in their phrases. More than half of the Arabic and the Persian characters have points. To hide a secret, the authors propose the vertical displacement of those points. Before hiding a secret, the authors propose to compress the secret information first. Then, they locate the first pointed letter in the cover text. The size of hidden information is also hidden in the beginning of the text. After that, the compressed secret bits are read. If the bit has value of zero, the pointed letter remains unchanged. In case the bit has value of one; the point of the pointed letter is shifted a little upward. This procedure is repeated for the next pointed letters in the cover text and the next bits of compressed secret information. Then, points of the remaining pointed letters are vertically displaced randomly to divert the attention of readers to have better security. To recover the bits, they identify all hidden bits in the letters based on the place of points on the character. After that, the decompression is done to get the original hidden secret. This approach has a fair capacity and reasonable robustness in printing and resizing. On the other hand, it requires having a new font and it works only with that font. Retyping and scanning the text can cause loss of hidden information. This approach is tested using several Iranian newspapers to prove the capacity of the approach. As explained above, the results give a good performance in capacity while security is still questionable. Figure 1 shows an example of an Arabic letter before and after the vertical displacement.

**Figure 1**. An example of a vertical displacement of the point in an Arabic letter

In [4], the authors propose a new watermarking technique to hide a secret by utilizing the extension character in Arabic language "Kashida" with the pointed Arabic letters. To hide the secret bits, the authors proposed using "Kashida" with pointed letters to represent 'one' while "Kashida" with un-pointed letters to represent 'zero'. The authors propose two ways to implement it: "Kashida" before and Kashida-after. "Kashida" before adding the extension letter before, while Kashida-after adds the extension letter after the current letter. The results of applying those techniques give a good performance in capacity, as compared to [2], while security is still unconvincing. However, the authors in [4] propose a secured method that mix Kashida-Before and Kashida-After by having odd lines encoded with one method and even lines encoded with the other one. A comparison between the results of this technique and previous work done by Shirazi [2] gave a clear idea about the increased capacity.

In [6], the authors propose a new steganography method to hide secret information into Arabic text cover media. The proposed approach utilizes diacritics in Arabic language which are used for vowel sounds and found in many religious documents. There are eight different diacritical symbols used in Arabic. They found that one diacritical symbol, "Fatha", is used in Arabic text as much the other seven diacritical symbols. So, they used "Fatha" symbol to represent 1 and the other symbols to represent 0. To hide bit of value 1, they search for the first applicable location for "Fatha" and then remove it. And to hide 0 they search for the first applicable location for other diacritical symbols and remove it. The advantage of this method is the high capacity since each Arabic letter is applicable for a diacritic. The disadvantage is that hiding some diacritics will get the reader's attention. Figure 2 shows text with diacritics and text without them.

In [7], the authors extend the use of diacritics to hide more information in the cover text. The main idea of their proposed approach is to put multiple diacritics on top of each other so that they will look invisible. Two approaches were proposed: one is based on text and one is based on images. The bit-representation is converted to decimal number. In the text approach, they put multiple diacritics that mapped to the decimal number. Then, they need to have a digital copy of the document and a program to extract the number

| Text with diacritics | Text without diacritics |
|---|---|
| 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 | حدثنا سفيان عن يحيى عن محمد بن إبراهيم التيمي عن علقمة بن وقاص قال سمعت عمر رضي الله عنه يقول سمعت رسول الله صلى الله عليه وسلم يقول إنما الأعمال بالنية ولكل امرئ ما نوى فمن كانت هجرته إلى الله عز وجل فهجرته إلى ما هاجر إليه ومن كانت هجرته لدنيا يصيبها أو امرأة ينكحها فهجرته إلى ما هاجر إليه |

**Figure 2**. An example text with and without diacritics

of hidden diacritics. In the image, the text containing multiple diacritics is converted into image and then analyzed to get the secret back. The advantage of multiple diacritics approach is the huge capacity since they can hide big secrets in one diacritic. However, the disadvantage is that putting diacritics in specific places in the documents gets the reader's attention.

In [8] and [9], the authors proposed two approaches based on the UNICODE encoding of the pseudo space and pseudo connection characters. Arabic letters are written in a connected way so that the letters of a word are connected. However, some Arabic letters can't be connected. After each connected letter, we can put pseudo connection character which is not visible. The pseudo connection character is known as zero width joiner (ZWJ). If the letter is not connected, we can put pseudo space character which is not visible as well. The pseudo space character is called zero width non joiner (ZWNJ). So, a pseudo character is put where applicable to hide 1 and skipped to hide 0. The advantages of this approach are the invisibility of the pseudo characters and the huge capacity since we can hide a secret bit after each Arabic letter. In the printing format, this approach is not helping since hidden information is invisible.

In [10], a new approach is proposed to hide information in the Arabic and Persian cover text based on the UNICODE codes for the letters. Based on the fact that the writing is connected, Arabic and Persian letters have four formats each based on their location in the word. Each format has a code in UNICODE and the unique format representing the letter has another code. The text is saved using the unique representation code. Any text-program which reads the text makes contextual analysis to show the correct format in the program. This approach proposes using the unique representation codes for a word to hide 0 and using the location-based format code for a word to hide 1. It has the advantage of good capacity since each one word in the cover text will be used to hide one bit. However, it will not help in the printing format since hidden information is invisible.

In [11], the authors propose using reverse "Fatha" to hide information in the cover text instead of the regular "Fatha". "Fatha" among other diacritics is the most used diacritic in Arabic, Persian and Urdu

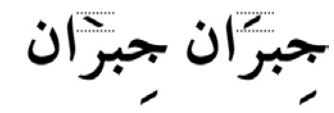

**Figure 3**. An example of regular "Fatha" and inverse "Fatha"

languages. We put an inverse format of "Fatha" where applicable on the same letters we want to hide. No one will notice this inverse "Fatha" easily which is an advantage. The disadvantage of this approach is the need for a new font to use to put the inverse "Fatha" since it is not a standard diacritic. Figure 3 shows both regular "Fatha" and inverse "Fatha".

In [12], a new approach to Steganography in Persian and Arabic texts is proposed. It uses special form of "La" word to hide information. "La" ("لا") is created when the letter "Lam" ("ل") is followed by the letter "Alef" ("ا"). To hide 0, we insert Arabic extension letter between "Lam" and "Alef" letters and use the normal form of "La". To hide 1, a special form of the word "La" with a unique code in Unicode is used. This method is used in limited format of the text and hence has less capacity than the others.

## 3 Proposed Approach

The idea is originally described briefly in [5]. There are 28 letters in Arabic language, where some letters have more than one format. For example, the letter {أ} has 6 formats {آ ، أ ، ﺃ ، ﺇ ، ﺈ ، ﺂ}. The Arabic keyboard contains a total of 35 different formats for the 28 letters. The Arabic extension letter, Kashida, can come before or after certain letter formats. In both cases, "Kashida" can't start a word and can't end a word, i.e. "Kashida" can't come in the beginning of a word and can't come in the end of a word. We can put "Kashida" after all Arabic letters if it is not the last letter and it is not from the letters {أ ، و ، د ، ذ ، ر ، ز}, in addition to the {ة} format of the letter {ت}. For example, let's take the word "كمال". We saw here we could put two Kashida(s) in 4-letter word. We could not put Kashida after the last letter {ل} and after {أ}.

We have studied Arabic letters to see their applicability to add "Kashida", as shown in Table 1. Table 1 shows the 28 Arabic letters followed by 35 letter formats. Then, it shows if "Kashida" comes before the letter with an example. Finally, the table shows if "Kashida" comes after the letter with an example again. Although the letter Lam (ل) can accept "Kashida" after itself, there are four exceptions. They happen when the letter Lam (ل) is followed directly by one format of the letter Alef (أ). Those letter formats are (ا ، ﺇ ، ﺃ ، ﺂ). In Arabic language, those two letters, Lam and Alef, when followed by each other are normally written dif-

**Table 1**. Arabic letters and their applicability for "Kashida"

| Arabic Letter | Letter Format | Num. Rep. | Applicable for "Kashida" Before Letter | | Applicable for "Kashida" After Letter | |
|---|---|---|---|---|---|---|
| | آ | 1570 | آـ | Yes | ـآ | No |
| | أ | 1571 | أـ | Yes | ـأ | No |
| أ | ؤ | 1572 | ؤـ | Yes | ـؤ | No |
| | إ | 1573 | إـ | Yes | ـإ | No |
| | ئ | 1574 | ئـ | Yes | ـئ | Yes |
| | ا | 1575 | اـ | Yes | ـا | No |
| ب | ب | 1576 | بـ | Yes | ـب | Yes |
| | ة | 1577 | ةـ | Yes | ـة | No |
| ت | ت | 1578 | تـ | Yes | ـت | Yes |
| ث | ث | 1579 | ثـ | Yes | ـث | Yes |
| ج | ج | 1580 | جـ | Yes | ـج | Yes |
| ح | ح | 1581 | حـ | Yes | ـح | Yes |
| خ | خ | 1582 | خـ | Yes | ـخ | Yes |
| د | د | 1583 | دـ | Yes | ـد | No |
| ذ | ذ | 1584 | ذـ | Yes | ـذ | No |
| ر | ر | 1585 | رـ | Yes | ـر | No |
| ز | ز | 1586 | زـ | Yes | ـز | No |
| س | س | 1587 | سـ | Yes | ـس | Yes |
| ش | ش | 1588 | شـ | Yes | ـش | Yes |
| ص | ص | 1589 | صـ | Yes | ـص | Yes |
| ض | ض | 1590 | ضـ | Yes | ـض | Yes |
| ط | ط | 1591 | طـ | Yes | ـط | Yes |
| ظ | ظ | 1592 | ظـ | Yes | ـظ | Yes |
| ع | ع | 1593 | عـ | Yes | ـع | Yes |
| غ | غ | 1594 | غـ | Yes | ـغ | Yes |
| ف | ف | 1601 | فـ | Yes | ـف | Yes |
| ق | ق | 1602 | قـ | Yes | ـق | Yes |
| ك | ك | 1603 | كـ | Yes | ـك | Yes |
| ل | ل | 1604 | لـ | Yes | ـل | Yes |
| م | م | 1605 | مـ | Yes | ـم | Yes |
| ن | ن | 1606 | نـ | Yes | ـن | Yes |
| ه | ه | 1607 | هـ | Yes | ـه | Yes |
| و | و | 1608 | وـ | Yes | ـو | No |
| | ى | 1609 | يـ | Yes | ـى | Yes |
| ي | ي | 1610 | يـ | Yes | ـي | Yes |

ferently as: (لآ، لإ، لأ، لا). Arabic readers see it is not convenient if "Kashida" comes between. Hence, we exclude "Kashida" to come between those letters.

The idea is to build a steganography schema and tool that utilizes the extension character "Kashida" in Arabic language to hide a secret. The motivation of this work is to maximize the capacity by utilizing all possible locations for "Kashida" in the Arabic letters. To achieve this, we have done a study to know which Arabic letters can be extended and we defined the rules for MSCUKAT to embed "Kashida" in Arabic text, as in the previous section. Based on the above study, we put "Kashida" where applicable and the bit representation of the secret has value of 1 while we skip it if the secret has value of 0. The algorithm is based on Binary Coded Decimal (BCD) representations as other papers do in this field. An important assumption is that the cover text is plain text without any formatting or justifying and it is without "Kashida".

A programming language (C#) with Dot Net Framework 2.0 is used for encoding and decoding the secret message. The cover media which is represented in an Arabic text is taken from text files so the program will take the Arabic text and embed a secret message in it using MSCUKAT technique. Moreover, the secret can be read from a text file and then converted to binary bit representation. The program is able to extract the secret from the cover media that has a secret.

The program has two parts: one for encoding secret in a cover media and the second for decoding the secret. The first part which is fully implemented has four steps:

(1) entering or uploading the secret,
(2) converting the secret to bit representation,
(3) entering or uploading the cover media, and finally,
(4) embedding (or encoding)the secret in the cover media.

Once you click on the fourth step (embedding the secret), the secret will be embedded using "Kashida" and MSCUKAT approaches with useful statistics information. Not only that, but it will export the statistics into text file to use it in the excel sheet.

Figures 4 and 5 were taken as snapshots of the MSCUKAT program. Figure 4 shows the full picture of the program. Figure 5 shows, in focus, the steps of encoding a secret. Figure 6 shows how the output encoded messages will be.

We have implemented the two "Kashida" approaches in [4]: Kashida-after and Kashida-before as explained below. Moreover, we have implemented the third approach suggested by the authors to have better security where we apply Kashida-after for odd lines and Kashida-before for even lines.

For Kashida-after, we put "Kashida" if we have 0 bit and we have applicable non-dotted character for putting "Kashida" after. We defined the applicable characters as shown in Table 1. Note that the letter (Lam) is applicable if it is not followed by the letter (Alef). We have considered that and have tested each (Lam). We have also tested to see if the next letter is space or enter so that we have excluded it since adding "Kashida" is not applicable in this case. On the other hand, if we have 1 bit and we have applicable dotted character for putting "Kashida" after. We defined the applicable characters as shown in Table 1. Also, we have tested if the next letter is space or enter so that we have excluded it since adding "Kashida" is not applicable in this case.

For Kashida-before, we put "Kashida" if we have 0 bit and we have applicable non-dotted character for

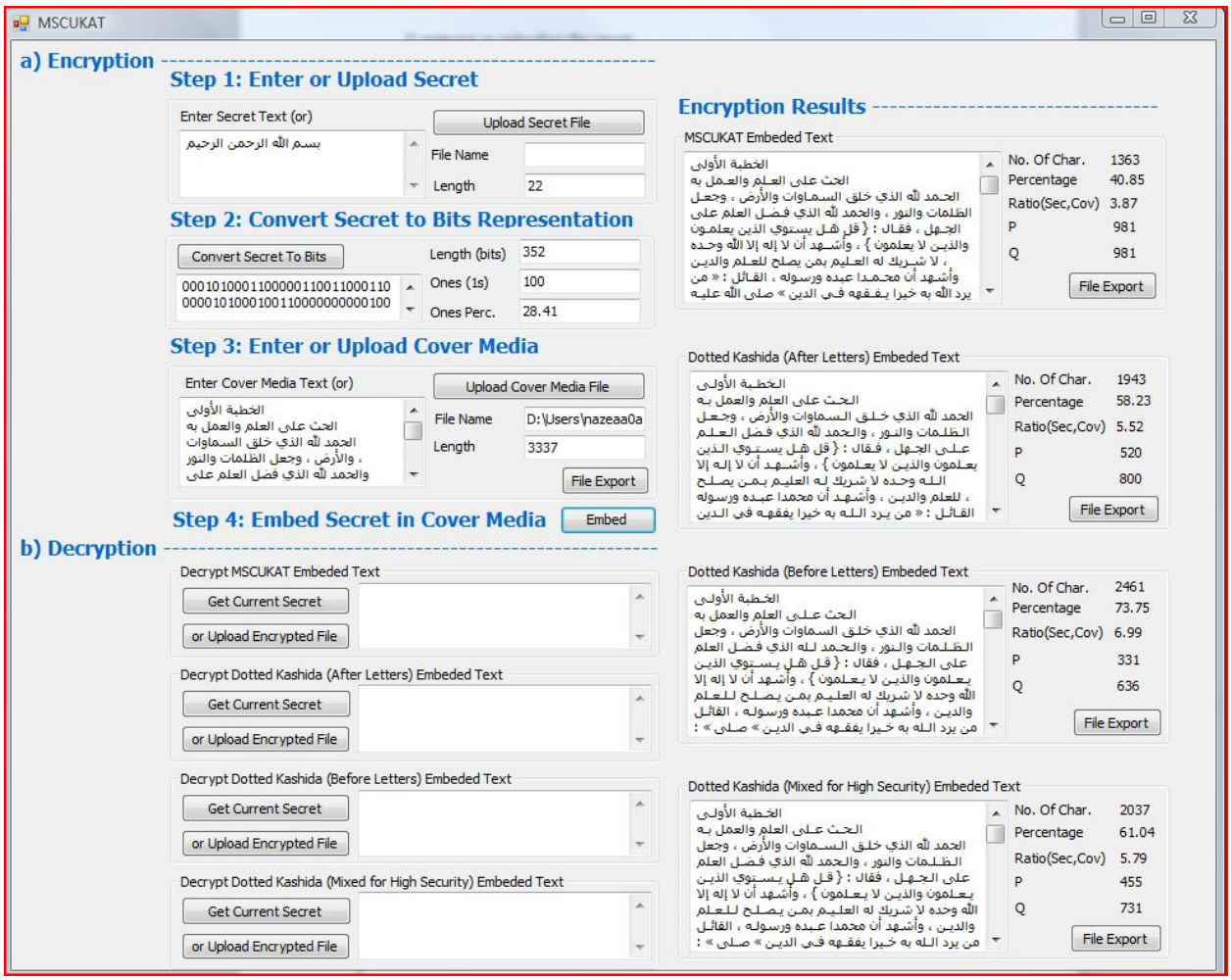


**Figure 4**. Snapshot of the MSCUKAT program - the full picture


**Figure 5**. Snapshot of the MSCUKAT program - in focus the steps of encoding a secret

putting "Kashida" before. We defined the applicable characters as shown in Table 1. Also, we have tested if the previous letter is space or enter so that we have excluded it, i.e. adding "Kashida" is considered not applicable in this case. We further studied if the previous character of the character we want to add "Kashida" before, if it is applicable to have "Kashida" after; we generally defined the applicable characters as shown in Table 1.

The following example is built based on the example shown in [4]. It clearly shows how we have implemented "Kashida" approaches. Moreover, it shows the result of applying the proposed MSCUKAT approach on the same cover media text. We suppose that we want to hide the secret "110010"in the following text:

"من حسن اسلام المرء تركه مالا يعنيه"

We count the number of characters in the cover media text to be 34. Then, we encode the secret and we count how many characters are needed to hide the secret as shown in Table 2.


**Figure 6**. Snapshot of the MSCUKAT program - output encoded messages

## 4 Improvements

The improvements of applying MSKUKAT can be measured when we compare it to "Kashida" approaches in [4]. We observe three improvements. First, the capacity of cover media is increased to hide more secret information. Second, the file size increase after hiding the secret is reduced since we have less addition to the original cover media. Third, the security of the encoded cover media is enhanced.

First, the main motivation of MSCUKAT is to increase the capacity of the cover media to hide longer secrets; secrets are represented in bits. As a result of increasing the capacity, MSCUKAT can hide more information according to our experiments in the next section; the proposed approach is giving capacity of 55% more "Kashida" approaches as compared to [4].

Second, another major improvement is that although we have increased the capacity of the cover media, the size of the cover media is not increased much. We have reduced the increase of the file size by 70% compared to "Kashida" approaches of [3, 4]. So, we are only increasing the file size with 30% of what it should increase compared to "Kashida" approaches of [3, 4]. This improvement comes from our observation to the secrets' size with the number of ones and its percentage. Our finding, from the sample secrets we have, is that the number of 1's in the secret is much less than the number of 0's. This is based on our experiment which assumes that the information we are encoding is simple text not encrypted or compressed. It is the same assumption of the other paper which we compared to, i.e. papers [3–5]. We found that on average the percentage of 1's in a secret is 28.4%. We set up MSCUKAT approach so that we put "Kashida" if we have the bit equals to 1 and this has a great impact on the increase of the cover media file size. The cover media file size is increased by small percentage. Finally, as a result of putting less number of extension letters, Kashida, in our approach, the Arabic readers see it more convenient and comfortable to read compared to "Kashida" approaches where we add Kashida(s) as much as the size of the secret.

**Table 2**. Example of applying MSCUKAT compared to "Kashida" approaches

| Secret bits | Cover media length | Cover media text |
|---|---|---|
| 110010 | 34 | من حسن اسلام المرء تركه مالا يعنيه |
| **Approach** | **Needed letters to hide secret** | **Output text** |
| Kashida-After | 32 | 1 1 0 0 1 0<br>من حسن اسلام المرء تركه مالا يعنيه |
| Kashida-Before | 32 | 1 1 0 0 1 0<br>من حسن اسلام المرء تركه مالا يعنيه |
| MSCUKAT | 17 | 1 1 0 0 1 0`<br>من حسن اسلام المرء تركه مالا يعنيه |

Third improvement is the enchantment of the security. From security point of view, one could count the number of the extension letters in "Kashida" encoded text to know the size of the secret. We think that our approach is more secure than [3] and [4], since the number of extension letters in MSCUKAT encoded text does not reflect the size of the secret.

## 5 Experiments and Comparisons

The cover media used in the experiments is taken from 15 Khotbas, Friday's speeches, of Ibn Othaimeen, Is-

**Table 3**. Comparison between MSCUKAT with "Kashida" approaches in capacity

| Approach | P | Q | (P+Q)/2 |
|---|---|---|---|
| Kashida-After | 0.163 | 0.224 | 0.194 |
| Kashida-Before | 0.109 | 0.167 | 0.138 |
| Kashida-Mixed | 0.136 | 0.196 | 0.169 |
| MSCUKAT | 0.300 | 0.300 | 0.300 |

lamic scholar, of different lengths [1] . Also, the eight secrets used in the experiments are the parts of Sorat Al-Fatiha from Holy Quran, the holy book of Muslims. We have compared the proposed approach, MSCUKAT, with previous "Kashida" approaches in [4] in different ways. In [4], there are three approaches when deciding to put "Kashida" to hide a secret. They are: (1) "Kashida-After" where we put "Kashida" after the applicable letter; (1) "Kashida-Before" where we put "Kashida" before the applicable letter; and (3) "Kashida-Mixed" where we put "Kashida" after the applicable letter in odd lines and put "Kashida" before the applicable letter in the even lines of the cover text.

First, we need to know that our approach is similar to "Kashida" approaches in the use of the extension letter "Kashida" to hide a secret bit. However, "Kashida" approaches use the extension letter to hide any secret bit whereas MSCUKAT approach uses the extension letter to hide only the secret bits with value 1. For the other secret bits which contain 0, we skip the applicable location and move to the next. In "Kashida" approaches, they use dotted letters to hide 1's and un-dotted letters to hide 0's whereas we did not distinguish between dotted and un-dotted letters in our approach.

To numerically compare the two approaches, we count the number of applicable locations to put the extension letter "Kashida" in the cover media in both approaches independent of the secret message. We have used the 15 speeches in [**?** ] and then taken the average as shown in Table 3 below. Similar to what given in [4], we use p to represent the ratio of the applicable locations to hide 1's to the cover media length. Also, we use q for ratio of the applicable locations to hide 0's to the cover media length. Finally, we average p and q by adding them up and dividing by 2. Table 3 shows the results.

We observe that MSCUKAT is performing better that the other "Kashida" approaches. It gives at least 55% (0.194:0.300) more capacity than that with the best "Kashida" approach in our experiment, namely Kashida-After.

[1] It is available online at http://www.ibnothaimeen.com/all/Khotab.shtml

**Table 4**. Comparison between MSCUKAT with "Kashida" Approaches in Coverage Percentage and Secret Occupation Ratio

| Approach | Coverage Percentage | Secret Ratio |
|---|---|---|
| Kashida-After | 49.2 | 5.1 |
| Kashida-Before | 67.3 | 7.2 |
| Kashida-Mixed | 56.4 | 5.9 |
| MSCUKAT | 32.8 | 3.4 |

Then, we calculated the ratio between the secret and the needed characters in the cover media to hide the secret. Also, we compared the percentage of the needed characters to hide the secret and the cover media size to see how much it would consume in order to hide this secret. We hid the 8 secrets, one by one, in the 15 cover media, again one by one. So, we had a total of $8 \times 5$ runs. Then, we averaged them as shown in Table 4 below.

To read Table 4 correctly, we need to know the following. *Number of char* is the number of first characters in the cover media that can hide the secret. *Cover percentage* is ((*Number of char*)/(Cover media length))×100; this gives the percentage of cover media that has been used to hide the secret. *Secret ratio* is the ratio between secret and cover media which equals to (secret length)/(*Number of char*). It helps to know the ratio between number of bits to be hidden and number of characters in the cover media that are enough to hide such secrets.

We observe that MSCUKAT is outperforming the other approaches by utilizing less percentage of the cover media to hide a secret. MSCUKAT is saving at least 33% (32.8:49.2) of the cover media compared to "Kashida" approaches. The ratio between the secret and the needed cover media size to hide is better in MSCUKAT with at least 33% (3.4:5.1).

The following bar charts (Figure 7 and Figure 8) illustrate this study. Figure 7 shows the comparison between "Kashida" approaches and MSCUKAT approach in the needed cover media percentage that can be used to hide a secret. Figure 8 presents a similar comparison but in the ratio between the secret and the needed cover media, i.e. how much we need (in ratio) to hide a certain secret with a defined length.

Based on the experiments we did, we observe that using MSCUKAT is giving much more capacity than using "Kashida" approaches. This study implies the limitation of the capacity when using "Kashida" previous approaches. On the other hand, using MSCUKAT

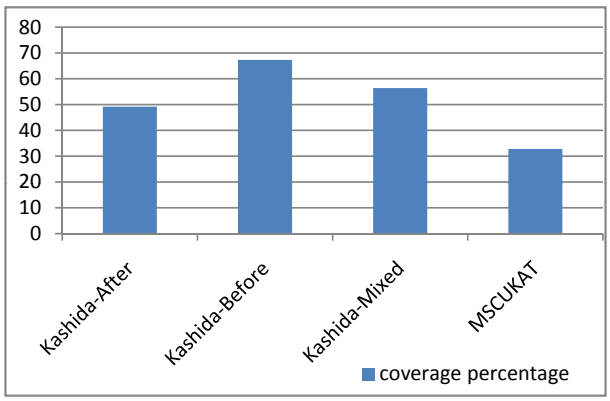


**Figure 7**. Comparison between MSCUKAT and "Kashida" approaches in the coverage percentage

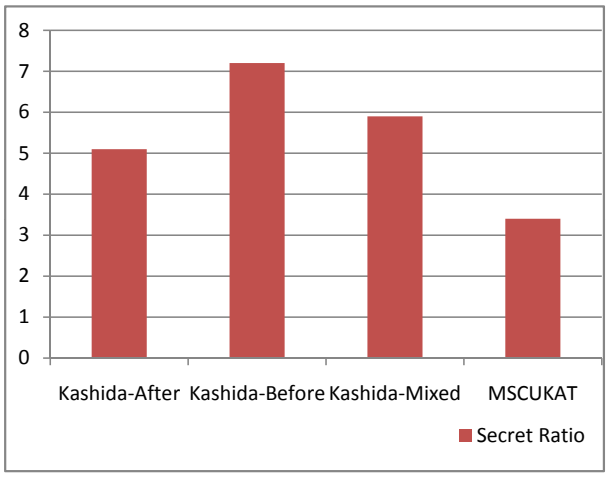


**Figure 8**. Comparison between MSCUKAT and "Kashida" approaches of ration between the secret and the needed cover media size

approach gives more possibility to hide longer secrets.

One important note we observe is the secrets' size with the number of ones and its percentage. We have studied the input secret that we can embed in a cover media. We want to analyze the number of 1's in the secret and its percentage compared to the size of the secret. Table shows our findings.

We observe that we have on average 29.3% of the secret are 1's and the other 70.7% are 0's. One important difference between our approach MSCUKAT and "Kashida" old approach [3, 4] is that we put extension letter for ones only while we skip the applicable location for "Kashida" if we want to hide 0. However, the extension letter is required for 0's and 1's in "Kashida" approaches described in [3, 4]. This is based on our experiment which also assumes that the information we are encoding is simple text not encrypted or compressed such as in the diacritics stego approach detailed in [6, 7]. The comparison is using the same assumption of the other paper, i.e. [4], which we compared to.

**Table 5**. Secrets Statistics

| Secret | Length | Number of 1's | Percentage |
|---|---|---|---|
| 1 | 208 | 63 | 30.3 |
| 2 | 224 | 63 | 28.1 |
| 3 | 336 | 95 | 28.3 |
| 4 | 336 | 106 | 31.6 |
| 5 | 352 | 100 | 28.4 |
| 6 | 352 | 104 | 29.6 |
| 7 | 352 | 105 | 29.8 |
| 8 | 464 | 131 | 28.2 |
| Average | 328 | 95.9 | 29.3 |

To encode the cover media with a secret, we need to add extension letters which will increase the size of the file. The proposed MSCUKAT approach is more efficient in encoding the message compared to "Kashida" approaches. Encoding the message in the proposed approach results in smaller file sizes as compared to "Kashida" approaches. On average, the file sizes are reduced by 70.7% by using the proposed approach. Moreover, we have two more experiments. The first one, we make the secret constant and we change the cover media. We have taken secret number 1, which is

"بسم الله الرحمن الرحيم"

and hid it in the fifteen-speech cover media. Once we converted the secret to bit-representation, it had 352 bits: 100 ones (28.4%) and the others were zeros. Table 6 shows the results.

Using the proposed approach, MSCUKAT, gives an average of 35.1% capacity, which means we can hide the specified secret by using 35.1% of the cover media. On the other hand, using "Kashida" approaches has an average of 62.4% capacity to hide the same secret. Clearly, using MSCUKAT approach is giving 78% better than using "Kashida" approaches.

In the second experiment, we fix the cover-media of length 5,567 characters (Last Khotba) and we change the secret. Table 7 shows the result of this experiment. Moreover, we observe that MSCUKAT approach is giving better capacity than "Kashida" approaches by at least 53% more.

Overall, MSCUKAT is giving better capacity. Whether we fix the secret and change the cover media or we fix the cover media and change the secret, we have similar results. From security point of view, one could count the number of the extension letters in "Kashida" encoded text to know the size of the secret. We think that our approach is more secure since the number of extension letters in MSCUKAT encoded text does not reflect the size of the secret.

**Table 6**. Comparison between MSCUKAT and "Kashida" approaches in the percentage of cover media occupations with fixed secret

| Cover media | Kashida-After | Kashida-Before | Kashida-Mixed | MSCUKAT |
|---|---|---|---|---|
| 1 | 71.33 | N/A | 83.87 | 48 |
| 2 | 70.99 | 97.32 | 83.22 | 46.82 |
| 3 | 63.92 | 98.28 | 77.25 | 44.61 |
| 4 | 59.1 | 78.33 | 67.69 | 39.2 |
| 5 | 60.09 | 90.63 | 69.49 | 38.83 |
| 6 | 58.23 | 73.75 | 61.04 | 40.85 |
| 7 | 48.43 | 67.56 | 54.39 | 33.31 |
| 8 | 48.66 | 75.27 | 59.79 | 32.58 |
| 9 | 47.79 | 66.52 | 56.28 | 32.18 |
| 10 | 47.64 | 65.55 | 56.45 | 31.75 |
| 11 | 47.29 | 66.35 | 54.58 | 30.13 |
| 12 | 46.18 | 68.11 | 51.48 | 30.07 |
| 13 | 46.49 | 64.29 | 55.3 | 28.33 |
| 14 | 47.13 | 63.09 | 53.89 | 29.26 |
| 15 | 33.68 | 44.8 | 35.23 | 20.8 |
| Average | 53.1 | 72.8 | 61.3 | 35.1 |

**Table 7**. Comparison between MSCUKAT and "Kashida" approaches in the percentage of secret occupations with fixed cover media

| Secret | Kashida-After | Kashida-Before | Kashida-Mixed | MSCUKAT |
|---|---|---|---|---|
| 1 | 31.31 | 43.27 | 33.91 | 20.8 |
| 2 | 19.94 | 25.4 | 19.94 | 12.47 |
| 3 | 33.68 | 44.8 | 35.23 | 20.8 |
| 4 | 29.28 | 41.64 | 33.91 | 19.94 |
| 5 | 30.11 | 43.27 | 35.85 | 20.8 |
| 6 | 19.81 | 26.93 | 21.43 | 13.24 |
| 7 | 30.84 | 40.4 | 33.91 | 19.94 |
| 8 | 41.82 | 56.3 | 49.15 | 26.55 |
| Average | 29.6 | 40.25 | 32.92 | 19.32 |

Other factors in the comparison between the two approaches are considered such as complexity, security and robustness. First, regarding complexity, MSCUKAT is less complex than the approaches in [4]. In [4], their approach is mixing the dotted letter with the extension character as well as the odd and even lines. In MSCUKAT, it is a straightforward approach to find applicable Kashida to hide the secret regardless of dotted letters and order of lines. Second, regarding security, both approaches are using "Kashida" to hide the secret bits but the approach in [4] is using "Kashida" to hide all bits (0's and 1's) where MSCUKAT is using "Kashida" to hide only 1's. This indicates that our approach has less appearance in the cover text and hence is more secure. The approach in [4] will get reader's attention because of the extensive use of "Kashida" in the cover text. Finally, regarding robustness, both approaches share the same level since they use explicit extension letter to hide information. Retyping the text or scanning it may cause loss of the hidden information.

## 6 Secured MSCUKAT

We would like to go further and make secured version of MSCUKAT to have better security and make it difficult to crack. It starts with the time we think of how we will save the number of bits of the secret. We

**Table 8**. Numbers from 0-9 and their bit representation

| Number | Bit Representation |
|---|---|
| 1 | 1000110000000000 |
| 2 | 0100110000000000 |
| 3 | 1100110000000000 |
| 4 | 0010110000000000 |
| 5 | 1010110000000000 |
| 6 | 0110110000000000 |
| 7 | 1110110000000000 |
| 8 | 0001110000000000 |
| 9 | 1001110000000000 |
| 0 | 0000110000000000 |

decided to save it at the beginning of the encoded cover media followed by the encoded secret bits using the extension letter, Kashida, as shown in the previous sections.

The size of the secret bits is taken as numerical representation (e.g. 8 means secret with 8 bits) and then convert it to its bit representation (e.g. 8 is converted to be 1000). Then, we put it at the beginning of the cover media by putting "Kashida for 1 and skipping the applicable location for 0 as we did previously. After that, we put a mask then we encode the secret bits.

Table 8 shows the numbers from 0 to 9 and their bit representation. The mask is "1111" since we have studied the bit representation of all numbers from 0 to 9 (which will compose the secret size) and found that there is no four ones followed by each other. The algorithm is based on the BCD representations as other papers do in this field.

Next, we looked at the secured MSCUKAT. We want to make the encoding process that hides the secret more securely by taking a *skip-number* between 0 and 4. This number is calculated by the following equation:

*skip-number=modular(cover-media-size, 5)*

Then, we skip none, one, two, three, or four characters if we have skip-number values 0, 1, 2, 3 or 4, respectively. We mean, by skipping, that we should not apply MSCUKAT in the desired location. Once we go through the cover media, we make a pointer of the *current-location* that is applicable to put "Kashida" in and we then skip if the following formula equals zero.

*modular(current-location, skip-number)*

For example, if *skip-number* is 3, it means that it will hide the secret in all possible location for "Kashida" but it will skip the third possible from each five possible locations.

Encoding the cover media with a secret using secured MSCUKAT requires more steps than normal MSCUKAT approach. In addition to skipping letter extension being relatively random (based on the length of the cover media), we go through the remaining unencoded text in the cover media and we do random encoding. This will assure higher security. After encoding the message, we need to know how to decode it. Here, we need the format of the encoded message explained at the beginning of this section. This is suggested as future work to be evaluated if it can help securing the proposed algorithm

## 7 Future Work

Although we got good results out of the experiments we conducted, we would like to highlight some future work to be done in order to have a comprehensive steganography solution. First, we need to implement the secured MSCUKAT approach, test it and compare it to both approaches in [4] and [2]. Second, we need to hide the length of the secret inside the cover-media to easily retrieve the secret information from the cover media. For this, we need to formulate the length and assign starting and ending bits to identify it easily. Third, we should use the extension character, Kashida, in the remaining un-used and applicable letters randomly to divert the attention of readers to have better security. This method works fine in [4] and gives an advantage in its security. Yet, the security of this system is based on an algorithm. If somebody knows the algorithm, he/she will be able to extract the secret information. In future, this can be enhanced by encrypting the secret to make it more challenging to decrypt.

Forth, we should enable MSCUKAT program to read cover-media that has hidden secret and then decode it automatically. It should recognize the size of the secret from reading the first part of cover media. Fifth, we should look at the possibility of encrypting the secret to have better security. The security of our system is based on an algorithm. If somebody knows the algorithm, he/she will be able to extract the secret information. This can be enhanced by encrypting the secret to make it more challenging to decrypt. Sixth, we should think of compressing the secret as in [2] to encode the cover media with smaller size secrets. This will increase the capacity much more but it might add computational overhead to compress and decompress the secret.

Seventh, we should use other formats for the secret. So far, we have used only text files. We look for using other file types like pdf and power points. For that, we need to convert those files to their bit binary repre-

sentation which is not an easy task. Eighth, we should make a web version of MSCUKAT to expose the use of it and make it publically used. This will help us with getting feedback from the users in order to improve it. Ninth, we should utilize richer data in the experiment to have better evaluation of MSCUKAT and compare it with "Kashida" approaches in [4]. Tenth, we need to implement Shahreza's approach [2] and compare it to our approach and "Kashida" approaches to have better evaluation in both capacity and security.

## 8 Conclusion

A study was done on characteristics of Arabic letters and how the extension letter, Kashida, can be embedded in between Arabic letters. Based on the results of the study, a new approach is proposed to hide a secret into Arabic text cover media using Kashida. The proposed approach is maximizing the use of "Kashida" to hide more information in Arabic text cover media. Based on this approach, sufficient algorithms have been designed and implemented in a system. The developed system, called MSCUKAT (Maximizing Steganography Capacity Using "Kashida" in Arabic Text) has been tested and shown promising results that outperform previous work in [4].

MSCUKAT gives 55% more capacity than best "Kashida" approach in [4] when we have counted the applicable locations for "Kashida" in the cover media independent of the secret. Moreover, MSCUKAT saves 33% of the cover media size that is used to hide a secret when compared to the best "Kashida" approach. The ratio between the secret and the needed characters in the cover media is better 33% in MSCUKAT. Once we have experimented constant secret with changing cover media, we have found that using MSCUKAT approach is giving 78% better than using "Kashida" approaches. Also, testing constant cover media with changing secret tells that MSCUKAT approach is giving better capacity than "Kashida" approaches by at least 53% more. On the other hand, the number of 1's in a secret is 29.3% of the secret size based on our study. This is based on our experiment which assumes that the information we are encoding is simple text not encrypted or compressed. It is the same assumption of the other paper which we compared to. The decrease of number of 1's implies reducing the file size (which will be increased after putting Kashida) by 70.7% using MSCUKAT compared to "Kashida" approaches. Based on our study, we conclude that we can have more capacity by utilizing the places of adding Kashida.

Cleary, the improvements include increasing the capacity of cover media to hide more secret information, reducing the file size increase after hiding the secret and enhancing the security of the encoded cover media. Moreover, we have proposed a new secured MSCUKAT approach that can improve the security of MSCUKAT. Future work can be carried out from our experiment to enhance the way we embed "Kashida" in the text.

## Acknowledgements

The authors would like to thank Umm Al-Qura University, Saudi Aramco and King Fahd University of Petroleum & Minerals (KFUPM) for supporting this work.

## References

[1] Steganography, 2009. Available at http://en.wikipedia.org/wiki/Steganography.

[2] M.H. Shirali-Shahreza and M. Shirali-Shahreza. A New Approach to Persian/Arabic Text Steganography. In *Proceedings of the IEEE/ACIS International Conference on Computer and Information Science (ICIS-COMSAR'06)*, pages 310–315, 2006.

[3] Adnan Gutub and Manal Fattani. A Novel Arabic Text Steganography Method Using Letter Points and Extensions. In *Proceedings of the WASET International Conference on Computer, Information and Systems Science and Engineering (ICCISSE'07)*, pages 28–31, Vienna, Austria, 2007.

[4] Adnan Gutub, Lahouari Ghouti, Alaaeldin Amin, Talal Alkharobi, and Mohammad K. Ibrahim. Utilizing Extension Character 'Kashida' With Pointed Letters For Arabic Text Digital Watermarking. In *Proceedings of the International Conference on Security and Cryptography (SECRYPT'07)*, Barcelona, Spain, 2007.

[5] Ahmed Al-Nazer and Adnan Gutub. Exploit Kashida Adding to Arabic e-Text for High Capacity Steganography. In *Proceedings of the International Workshop on Frontiers of Information Assurance & Security (FIAS'09) in conjunction with the IEEE 3rd International Conference on Network & System Security (NSS'09)*, Gold Coast, Queensland, AUSTRALIA, 2009.

[6] Mohammed Aabed, Sameh Awaideh, Abdul-Rahman Elshafei, and Adnan Gutub. Arabic Diacritics Based Steganography. In *Proceedings of the IEEE International Conference on Signal Processing and Communications (ICSPC'07 )*, pages 756–759, Dubai, UAE, 2007.

[7] Adnan Gutub, Yousef Elarian, Sameh Awaideh, and Aleem Alvi. Arabic Text Steganography Using Multiple Diacritics. In *Proceedings of the 5th IEEE International Workshop on Signal Process-*

*ing and its Applications (WoSPA'08)*, University of Sharjah, Sharjah, UAE, 2008.

[8] M.H. Shirali-Shahreza and M. Shirali-Shahreza. Steganography in Persian and Arabic Unicode Texts Using Pseudo-Space and Pseudo-Connection Characters. *Journal of Theoretical and Applied Information Technology (JATIT)*, 4 (8):682–687, 2008.

[9] M. Shirali-Shahreza. Pseudo-space Persian/Arabic text steganography. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC'08)*, pages 864–868, 2008.

[10] Mohammad Shirali-Shahreza and Sajad Shirali-Shahreza. Persian/Arabic Unicode Text Steganography. In *Proceedings of the Fourth International Conference on Information Assurance and Security (ISIAS'08)*, pages 62–66. IEEE Computer Society, 2008.

[11] Jibran Ahmed Memon, Kamran Khowaja, and Hameedullah Kazi. Evaluation of Steganography for Urdu /Arabic Text. *Journal of Theoretical and Applied Information Technology (JATIT)*, 4 (3):232–237, 2008.

[12] Mohammad Shirali-Shahreza. A New Persian/Arabic Text Steganography Using "La" Word. In *Proceedings of the International Joint Conference on Computer, Information, and Systems Sciences, and Engineering (CISSE'07)*, pages 339–342, Bridgeport, CT, USA, 2007. Springer Verlag.

**Adnan Abdul-Aziz Gutub** is currently working as Chairman of the Information Systems Department at the College of Computer & Information Systems within 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 a researcher at the Center of Research Excellence in Hajj and Omrah (HajjCoRE) at Umm Al Qura University.

Adnan 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 many administrative academic positions 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.

**Ahmed Ali Al-Nazer** is a PhD candidate in Computer Science and Engineering at King Fahd University of Petroleum and Minerals (KFUPM) in Saudi Arabia. He has completed all the PhD course requirements and working on the dissertation proposal. Since 2001, Ahmed is working in the IT of the largest oil producer company in the world, Saudi Aramco where he got exposed to real world information technology deployments.

Ahmed received his BSc degree in Computer Science in 2001and MSc degree in Computer Science in 2006 both from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. His thesis title was: "Collaborative Autonomous Interface Agent for Personalized Web Search". Ahmed's research areas are in semantic web, data mining, steganography, security applications, software engineering, machine learning, personalization, search engines technologies and enterprise search. He worked on Arabic stegnoagrpahy on new powerful techniques to hide information in the Arabic text.

Ahmed has published several technical papers and conducted technical researches and participated in many scientific conferences. He delivered couple of seminars & public lectures in IEEE and local conferences. In addition, he participated in several funded research projects.