Exploit Kashida Adding to Arabic e-Text for High Capacity Steganography

Ahmed Al-Nazer
Saudi Aramco, Dhahran
Saudi Arabia
ahmed.nazer@aramco.com

Adnan Gutub
King Fahd University of Petroleum & Minerals,
Dhahran, Saudi Arabia
gutub@kfupm.edu.sa

***Abstract*—Steganography is the ability to hide information in a cover media such as text, and pictures. An improved approach is proposed to embed secret into Arabic text cover media using Kashida, an Arabic extension character. The proposed approach is maximizing the use of Kashida to hide more information, represented in binary bits, in Arabic text cover media. A stego system has been developed based on this approach. After sufficient testing and evaluation, our system shows promising performance in terms of capacity.**

Keywords—text steganography, text watermarking, text hiding, Arabic text, maximizing capacity, Kashida

I. INTRODUCTION

Steganography is defined as "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" [1]. The advantage of steganography is that the messages do not attract attention to themselves, to senders and to receivers [1].

Steganography works as we hide information in un-used and redundant bits in any cover media such as pictures and sound files. Hiding such bits in text is more challenging because we have less un-used bits but the result is more appreciated since it has less size and it is easier to transfer over the network [4,5].

Different human being languages have different characteristics and properties. Text steganography as it is hiding a secret inside text has dependencies on the language used as cover media. 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 purposes of using the extension character (Kashida) in Arabic text. One is to decorate the Arabic text format so that it looks better and more convenient. Second is to justify the Arabic writings within lines. It is similar to English where spaces are used for justifying the text in lines. The advantage of using Kashida in Arabic text formatting and justifying the lines does not affect the writing contents [2].

In this paper, a new approach is proposed to maximize the use of Kashida between joined Arabic characters. This approach is based on putting Kashida wherever possible in the join between different Arabic characters. This work enhances previous work done using Kashida for Steganography or Watermarking [6, 7, 8]. This method has been developed as a stego system for Arabic e-text called MSCUKAT (Maximizing Steganography Capacity Using Kashida in Arabic Text).

The rest of this paper is organized as follows. First, brief background information about Arabic characters properties is presented. Then, a review of work related to utilizing Kashida in Arabic text steganography. Next, the proposed approach is described in details. Afterwards, evaluation and testing results are reported and comparison to the previous work is analyzed. Finally, a conclusion will summarize the findings.

II. BACKGROUND

In [2], the paper proposes a new watermarking technique to hide a secret by utilizing the extension character in Arabic language (Kashida) with the pointed Arabic letters to hide a secret. 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'. This method is under secrecy feature coding methods.

The results of applying this technique give a good performance in capacity while security is still under low precision. A comparison between the result of this technique and previous work done by Shirazi will give a clear idea about the increased capacity. A recommended future work is to use Kashida alone and this is our part in the project. Our approach is extending this approach in [2, 6] by extending the Arabic text using Kashida whenever possible in the text as performed differently in [7,8]. For

that, we have defined the rules where Kashida can fit in the text.

III. PROPOSED APPROACH

The objective of the project is to build a steganography schema and tool that utilize the extension character (Kashida) in Arabic language to hide a secret. As part of the project, a study is done to know which Arabic letters can be extended and to define the rules for MSCUKAT to embed Kashida in Arabic text. In this study, we make sure that whenever we add Kashida it does not affect the look of the text.

The idea and motivation of this project is to maximize the capacity by utilizing all possible location for Kashida in the Arabic letters.

There are 28 letters in Arabic language. Some letters have more than one format. For example, the letter { ا } has 6 formats {آ, أ, ؤ, إ, ئ, ا}.

Kashida can come before certain letter formats and 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 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 Kashidas in 4 letters word. We could not put after the last letter {ل} and after {أ}.

We have studied Arabic letters to see their applicability to add Kashida. Based on our study, we define the letters that can come after Kashida which are the following table. Also we have studied the letters that can come before Kashida as part of our MSCUKAT technique. The following table shows also those letters.

The following table shows the Arabic letters and their applicability to be extended. It shows the 28 Arabic letters followed by 35 letter formats. Then, it shows if Kashida comes before the letter with an example. Finally, it shows if Kashida comes after the letter with an example again.

Although the letter (ل) can accept Kashida after the letter, there are four exceptions. They happen when the letter (ل) is followed directly by one format of the letter (أ). Those letter formats are (آ, أ, إ, ا). In Arabic language, those two letters () and () when followed by each other, they are written in well known look: (لا، لأ، لإ، لآ). Arabic reader does not see it convenient if Kashida came between. Hence, we exclude Kashida to come between those letters.

Based on the above study, we put Kashida where it is applicable and the bit representation of the secret has value of 1. A programming language (C#) with Dot Net Framework 2.0 is used for encoding and decoding the secret message.

Table 1: Arabic Letters and Their Applicability for Kashida before and after the letter

Arabic Letter	Letter Format	Num. Rep.	Applicable for Kashida Before letter		Applicable for Kashida After letter	
أ	آ	1570	ـآ	Yes	آـ	No
	أ	1571	ـأ	Yes	أـ	No
	ؤ	1572	ـؤ	Yes	ؤـ	No
	إ	1573	ـإ	Yes	إـ	No
	ئ	1574	ـئ	Yes	ئـ	Yes
	ا	1575	ـا	Yes	اـ	No
ب	ب	1576	ـبـ	Yes	بـ	Yes
ت	ة	1577	ـة	Yes	ةـ	No
	ت	1578	ـتـ	Yes	تـ	Yes
ث	ث	1579	ـثـ	Yes	ثـ	Yes
ج	ج	1580	ـجـ	Yes	جـ	Yes
ح	ح	1581	ـحـ	Yes	حـ	Yes
خ	خ	1582	ـخـ	Yes	خـ	Yes
د	د	1583	ـد	Yes	دـ	No
ذ	ذ	1584	ـذ	Yes	ذـ	No
ر	ر	1585	ـر	Yes	رـ	No
ز	ز	1586	ـز	Yes	زـ	No
س	س	1587	ـسـ	Yes	سـ	Yes
ش	ش	1588	ـشـ	Yes	شـ	Yes
ص	ص	1589	ـصـ	Yes	صـ	Yes
ض	ض	1590	ـضـ	Yes	ضـ	Yes
ط	ط	1591	ـطـ	Yes	طـ	Yes
ظ	ظ	1592	ـظـ	Yes	ظـ	Yes
ع	ع	1593	ـعـ	Yes	عـ	Yes
غ	غ	1594	ـغـ	Yes	غـ	Yes
ف	ف	1601	ـفـ	Yes	فـ	Yes
ق	ق	1602	ـقـ	Yes	قـ	Yes
ك	ك	1603	ـكـ	Yes	كـ	Yes
ل	ل	1604	ـلـ	Yes	لـ	Yes
م	م	1605	ـمـ	Yes	مـ	Yes
ن	ن	1606	ـنـ	Yes	نـ	Yes
ه	ه	1607	ـهـ	Yes	هـ	Yes
و	و	1608	ـو	Yes	وـ	No
ي	ى	1609	ـىـ	Yes	ىـ	Yes
	ي	1610	ـيـ	Yes	يـ	Yes

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 representation. The program is able to extract the secret from the cover media that has an embedded secret. The following image is a snapshot of the MSCUKAT application.

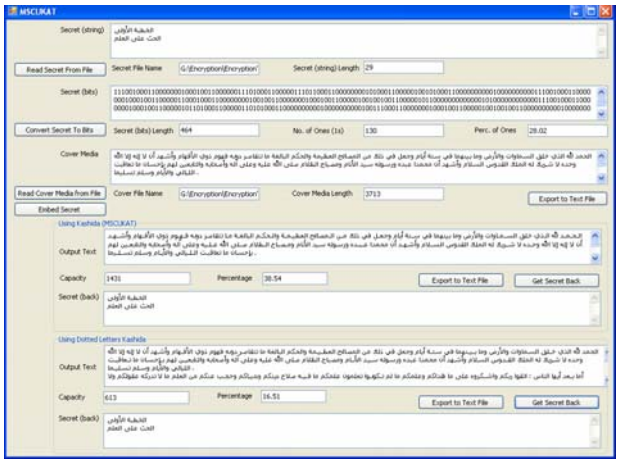

Figure 1: Snapshot of the MSCUKAT application

IV. EXPERIMENTAL RESULTS

In the evaluation phase of MSCUKAT, we compare the previous approach done in [2] which is focusing on adding Kashida after dotted letters in Arabic text. The main factor we compare is the capacity. We define the capacity as number of applicable places that we can add Kashida in the cover media to hide a secret.

For example, if we have cover with size 100 letters and we have a secret of 16 bits. We observe that we can hide the 16 bits using Kashida where it is applicable in the first 80 letters of the cover media. Hence the capacity of the cover media is 80 to hide that secret. The percentage of the capacity is 16/80 = 20%.

For dotted letters, we implemented the algorithm in [2] and we have selected to put Kashida after dotted letter if we want to hide 1 and put Kashida after non-dotted letter if we want to hide 0.

The data used in the experiment is taken from 15 Khotbas, Friday's speeches, of Ibn Othaimeen, Islamic scholar, with different length for each [3].

Based on the experiments we did, we observe that using MSCUKAT is giving much more capacity than using Kashida with dotted letters.

We have two experiments. The first one, we make the secret constant (352 bits) and we change the cover media. The following table shows the results.

Table 2: Comparison between MSCUKAT and Kashida with Dotted Letters with fixed secret

#	Cover Media Length	MSCUKAT		Dotted Letters	
		Capacity	Per %	Capacity	Per %
1	2,357	861	40.88	1,653	21.29
2	2,503	845	41.66	1,785	19.72
3	2,905	977	36.03	1,649	21.35
4	2,990	909	38.72	1,741	20.22
5	3,137	962	36.59	1,681	20.94
6	3,337	997	35.31	1,883	18.69
7	3,591	924	38.10	1,677	20.99
8	3,656	933	37.73	1,622	21.70
9	3,689	873	40.32	1,639	21.48
10	3,713	930	37.85	1,751	20.10
11	3,747	894	39.37	1,784	19.73
12	3,794	921	38.22	1,606	21.92
13	3,893	855	41.17	1,603	21.96
14	4,040	932	37.77	1,728	20.37
15	5,567	880	40.00	1,623	21.69
Average			39.00		21.00

Using MSCUKAT gives an average of 39% capacity, that is mean we can utilize 39% of the cover media to hide a certain secret of length 352 bits. On the other hand, using Kashida with dotted letters gives an average of 21% capacity to hide the same secret. Clearly, using MSCUKAT is giving 186% better than using Kashida with dotted letters. The following chart visualizes the above table.

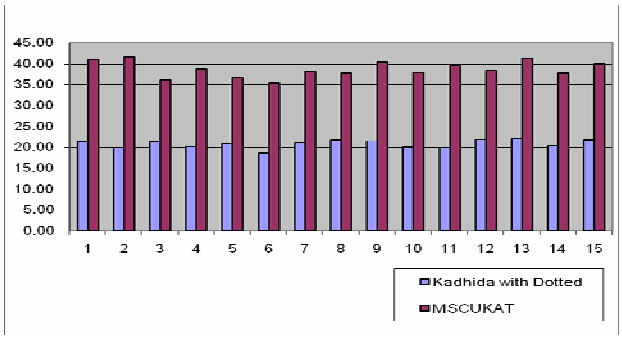

Figure 2: Comparison between MSCUKAT and Kashida with Dotted Letters with fixed secret

In the second experiment, we fix the cover-media with length 5567 characters (Last Khotba) and we change the secret. We have selected 8 different secrets; each one of them represents part of Surat AlFatiha, the first Sura in Holy Quran. Then, we covert each part to its bit-representation to be hided inside the fixed cover-media. The following table shows the result of this experiment.

Table 3: Comparison between MSCUKAT and Kashida with Dotted Letters with fixed cover-media

#	Secret Length	MSCUKAT		Dotted Letters	
		Capacity	Per %	Capacity	Per %
1	208	540	38.52	996	20.88
2	224	580	38.62	1,090	20.55
3	352	880	40.00	1,623	21.69
4	336	846	39.72	1,623	20.07
5	336	846	39.72	1,647	20.40
6	352	880	40.00	1,680	20.95
7	352	880	40.00	1,647	21.37
8	464	1,168	39.73	2,178	21.30
Average			39.50		20.90

We observe that in the two experiments we have similar average capacity for both MSCUKAT and dotted-letters. 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.

In the second experiment, we observe that having same-length secrets with different bits affect the capacity. This effect is taking place event if we have the same cover-media.

The following graph represents the second experiment results which are located in Table-3.

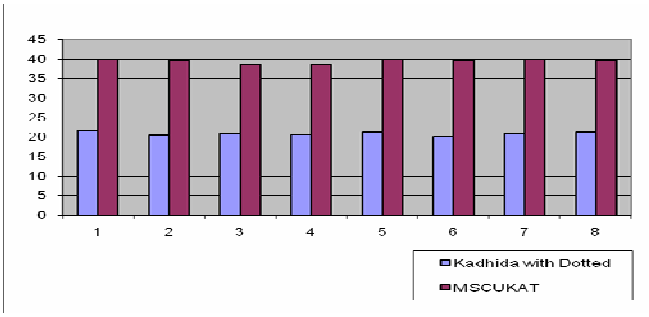

Figure 3: Comparison between MSCUKAT and Kashida with Dotted Letters with fixed cover media

Although the size of embedded secret in the cover media will not change, the observed limitation of the capacity of using Kashida with dotted letters affect the ability to hide a long secret in a limited size cover media. It means if we have a long secret, there is high probability that Kashida with dotted letters will not survive. This implies an advantage of using MSCUKAT that it gives us more possibility to hide longer secrets.

Moreover, we have studied the input secret that we can embed in a cover media. We want to analyze the number of 1s in the secret and its percentage compared to the size of the secret. This will open a future work to better utilize the cover media to have more capacity. The following table shows our findings.

Table 4: Percentage of 1s in a secret

#	Length (string)	Length (bits)	Number of 1s	Per %
1	2,357	38,000	10,304	27.12
2	2,503	40,048	10,595	26.46
3	3,337	53,392	14,104	26.42
4	3,747	59,952	15,773	26.31
5	3,893	62,288	16,718	26.84
Average				27.00

Based on the above table, we have only 27% average number of ones in the secret. It means that the other 73% are zeros and hence we only utilize 27% of the capacity to add Kashida. This also gives us a hint that the file size will not be increasing much, it will increase 27% of the capacity to add Kashida if we utilize the full capacity. The following bar chart visualizes the table.

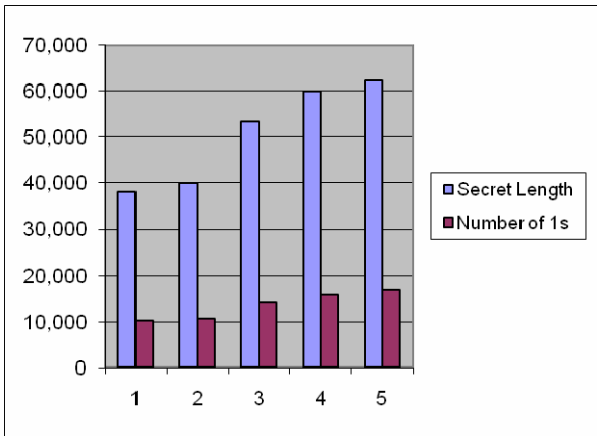

Figure 4: Percentage of 1s in a secret

V. CONCLUSION & FUTURE WORK

A study was done on characteristics of Arabic letters and how Kashida will affect its look. The results of the study help us to answer some questions like: “Is it proper to use Kashida whenever possible?” “How many places in an Arabic text I can embed Kashida (percentage)?”

Based on the proposed approach, a new system has been developed. It enables us to embed a chosen secret after it is converted to bit presentation and gives the result as cover media of text and a secret inside it. The system is also able to read the cover media and extract the secret which is in bit representation. Then, conversion of those bits will return back the secret in its original format.

Based on the evaluation of the new system MSCUKAT, we observe a huge advancement in terms of capacity comparing to the pervious approach in [2] which utilizes Kashida with dotted letters. Our experiments show 186% increase in capacity using MSCUKAT approach compared to the previous approach, Kashida with dotted letters. Moreover, the size of embedded secret in the cover media will not change.

Future work can be carried out from our experiment to enhance the way we embed Kashida in the text. Based on our study, we conclude that we can have more capacity by utilizing the places of adding Kashida.

ACKNOWLEDGMENT

We would like to thank King Fahd University of Petroleum & Minerals (KFUPM) for supporting this work.

REFERENCES

[1] Steganography, from Online website of Wikipedia, http://en.wikipedia.org/wiki/Steganography, last visited March 28, 2008

[2] Adnan Gutub, Lahouari Ghouti, Alaaeldin Amin, Talal Alkharobi and Mohammad Ibrahim, “Utilizing Extension Character ‘Kashida’ With Pointed Letters for Arabic Text Digital Watermarking”, International Conference on Security and Cryptography - SECRYPT, Barcelona, Spain, July 28 - 31, 2007

[3] Online, http://www.ibnothaimeen.com/all/Khotab.shtml e-Library of Shaikh binothaimeen Charity Organaization,

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

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

[6] Adnan Gutub and Manal Fattani, “A Novel Arabic Text Steganography Method Using Letter Points and Extensions”, WASET International Conference on Computer, Information and Systems Science and Engineering (ICCISSE), Pages: 28-31, Vienna, Austria, May 25-27, 2007.

[7] Adnan Gutub, Fahd Al-Haidari, Khalid Al-Kahsah, and Jameel Hamodi, "e-Text Watermarking: Utilizing 'Kashida' Extensions in Arabic Language Electronic Writing", *Journal of Emerging Technologies in Web Intelligence (JETWI)*, September 2009.

[8] Fahd Al-Haidari, Adnan Gutub, Khalid Al-Kahsah, and Jameel Hamodi, "Improving Security and Capacity for Arabic Text Steganography Using 'Kashida' Extensions", AICCSA-2009 - The 7th ACS/IEEE International Conference on Computer Systems and Applications, Pages: 396-399, Rabat, Morocco, 10-13 May 2009.

Ahmed Ali Al-Nazer is currently a student at King Fahd University of Petroleum & Minerals (KFUPM) and is pursuing his Doctor of Philosophy (PhD) from College of Computer Sci & Engineering. He received the Bachelor degree (BS) and Masters degree (MS) in Computer Science also from KFUPM, Saudi Arabia. Beside him as a researcher in the computing and IT field, he is an experainace employee at Saudi Aramco, Dhahran. Ahmad's late research activities and interests include computer security and its relation to Arabic language.

Adnan Abdul-Aziz Gutub is an associate professor in Computer Engineering at 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 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 in KFUPM; currently, he is chairing the computer engineering department (COE) at KFUPM in Dhahran, Saudi Arabia.