Journal of Computer Science & Computational Mathematics, Volume 6, Issue 3, September 2016
DOI: 10.20967/jcscm.2016.03.004

Merging Two Steganography Techniques Adjusted to Improve Arabic Text Data Security

Safia Meteb Al-Nofaie[1], Manal Mohammed Fattani[2], Adnan Abdul-Aziz Gutub[1]

[1]Computer Engineering Department, Umm Al-Qura University, Makkah, Saudi Arabia
[2]Collage of Da'wa and Usul-ud-Din (Islamic Studies), Umm Al-Qura University, Makkah, Saudi Arabia
[*]Corresponding author email: aagutub@uqu.edu.sa

Abstract: Securing private texts to fully prevent any detection, is a technique called Steganography. This research work of Arabic text steganography focuses on improving the hiding secrets within Arabic language text utilizing the redundant extension “Kashida” letter as covering media. We propose modifying the “Kashida” stego-cover technique merging with a new way of embedding sensitive data within whitespaces. This modified merging was tested on hiding data within Arabic text of the last 30 Chapters of the Holy Quran (Sura Al-Buruj #85 to Sura An-Nas #114) comparing with both the normal method and the improved proposed one. The results demonstrated clear increase in the capacity as expected without degrading the security, which justifies this work for real promising research direction.

Keywords: Information Security, Arabic text steganography, Kashida Steganography, whitespace utilization, text hiding, Data Security.

1. Introduction

The use of insecure networks for exchanging information boosts the problems of breaking the privacy of secure data. One of the most serious sensitive data needing this type of privacy systems is found in health sectors. In fact, the percentage number of criminal attacks on healthcare systems has doubled since 2010 [1]. One good way to protect the confidentiality property of the exchanged data is steganography, 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" [2].

Steganography can hide secret information in redundant and un-used bits in any cover media such as image, audio and text [3]. This work is using a text as a preferred media over others because of its efficiency, less memory usage, and cost-saving in printing and networking [4] as well as its wide use in social media [5]. Hiding inside texts is considered more challenging than all other media types (Audio, Images and Videos) including the combination of all data visualization methods [6]. Note that atext does not have much of normal redundant bits, i.e. extra unneeded bits, to be used in hiding secrets. Any changes in the text may be easily visible to detect (visually or electronically) making secure text stego-systems really complex to be accepted [7]. Cyber-security professionals cannot smoothly agree to new text stego-systems due to this challenge compared to hiding within Audio, Images and Videos in the steganography techniques knowledge [8].

Text steganography methods depend on the language of cover media with its different characteristic and properties. In this work, we used Arabic language which has 28 different letters. Every word in Arabic contains more than one character connected with each other. This connectivity feature is useful to embed the well known extension character called "Kashida" between letters, i.e. the connected letters [7]. Kashida is originally found to decorate the Arabic text to either format it or justify the lines, which does not affect the text contents nor meaning [9].

This paper improves an approach of Kashida utilization within steganography known as ‘Arabic Text Steganography Using the Extension 'Kashida' Character’ [10] and compare with it. The idea merges two techniques to gain the capacity by adding whitespaces to Kashida as extra redundant cover bits to hide in. The "Kashida" is assumed to be imbedded wherever it is possible after any Arabic letter regardless of being dotted or not, i.e. different from both ‘Arabic Text Steganography Method Using Letter Points and Extensions’ [4] and ‘Improving Security and Capacity for Arabic Text Steganography Using 'Kashida' Extensions’ [11]. This point is added to the modification of merging Kashida with using the whitespaces between words, i.e. also to hide secret bits.

The rest of this paper is organized as follows. Section 2 presents a background and related study of Arabic language characters properties. Section 3 introduces different approaches related to Arabic text steganography and clarifies the testing methods of ‘Arabic Text Steganography Using the Extension 'Kashida' Character’ in [10] and the related results. After that, in Section 4, we describe the details of the proposed approach including the idea, algorithm and implementation. Section 5 afterwards, presents a comprehensive comparison between the proposed approach and the others in [10]. Finally, Section 6 summarizes the findings in a brief conclusion and suggests future work ideas to be considered related to this effort.

2. Background

Information Security has become a core requirement for software application, driven by the need to protect critical assets and the need to build and prevent widely trust in computing [12].

In 1996, the noted beginning of information hiding workshops took place in Cambridge. Then, series of workshops became the premium annual meeting place to present latest advancements in theory and applications of data hiding [13]. The most important requirement of any steganography system is that it should be impossible for an eavesdropper to distinguish between ordinary objects and stego-objects that contain secret data [14]. Therefore, Steganography ultimate objectives, which are un-detect-ability, robustness and capacity of the hidden data, are the main factors that are different from the related techniques such as watermarking and cryptography [15].

In Arabic letters and other similar languages characters, i.e. Persian and Urdu, there are special characteristics that can be useful in consideration for hiding data. In the rest of this section, we explore some Arabic language properties to use some of its attributes to possibly hide large amount of data inside carrier file [16].

2.1. Writing Direction

An Arabic text is written from right to left. It is an unidirectional language and numbers are read and written in the same direction.

2.2. Letter connectivity

Most of Arabic language letters in different words are connected with the previous letter and the next one. Therefore, the letter may have different shapes depending on its position in words as shown in Figure 1.

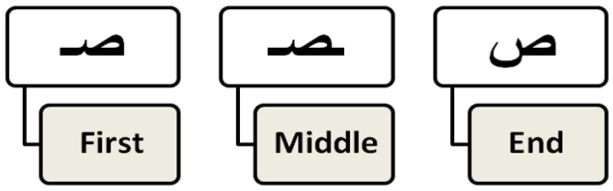


Figure 1. Letter (Saad) different shapes based on its position in words.

2.3. Dotted letters

Some Arabic letters have one, two and sometimes three points (dots). These points affect the letter's pronunciation [17], as shown in Table 1.

Table 1. Arabic letters

Number of points	Letter
0	ا,ح,د,ر,ص,ط,ع,ك,ل,ه,و
1	ب,ج,خ,ذ,ز,ض,ظ,غ,ف,ن
2	ت,ق,ي
3	ث,ش

2.4. Kashida letter

Kashida is one of Arabic language characteristic to extend the letter without any meaning effect. This is represented as (-) connecting the letters. As (احمد), can be written as (احـمـد) where we add two Kashida, i.e. around the middle letter. This means that we can embed 2 bits inside this word. Therefore, if we have any word which consists of *N* connected letters we can embed *N-1* bits inside it. If our text contains J words that contain connective letters, the number of embedded bits is = *(N-1)× J.*

3. Arabic Steganography Related Work

Text steganography has different methods used for hiding secrets, with advantages and disadvantage. In the following, we will present some of the prior works and provide some discussions on them.

The original proposal found in this field was presented by Shirali-Shaherza [3]. They depend on the points (dots) inherited in the Arabic, Urdu and Persian letters for hiding binary value. The location of the point is slightly shifted up if the hidden bit value is one as shown in Figure 2; otherwise, the location remains unchanged [18].

This method did not attract much attention although it can hide a large volume of information in Arabic text. It took high capacity in storing hidden bits due to the fact that Arabic language has 15 out of 28 letters having points, i.e. more than the letters of Arabic language points. But the drawback is in the robustness such that the output font cannot be standard. Thus, the receiver will not be able to extract the secret message if the output font is not installed on his machine. Also, the hidden information can be lost in any retyping or scanning process.

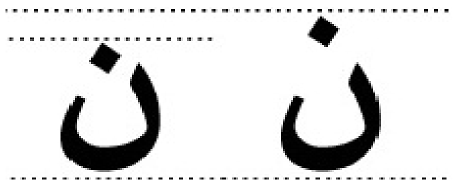
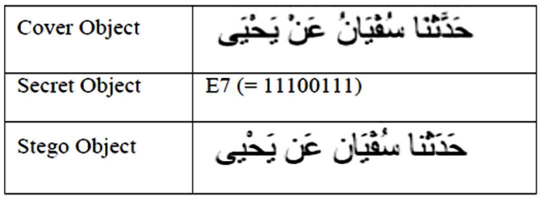


Figure 2. Example of vertical displacement of the point in an Arabic letter to hide a bit value

Another related research proposal [19] aims to use the advantages of the eight diacritics in Arabic scripts [20] to implement steganography. This diacritic approach, assigned the most available frequent well-known diacritics, i.e. Fat'ha, to hide the bit value equals "one", and all remaining other seven diacritics were assigned to hide "zero". The implementation of this diacritic stego-method starts by sensing the first secret bits, if it was found as "one", the system searches for the first Fat'ha diacritic in the cover media to hide in it [19]. However, if the diacritic is not Fat'ha and the bit value is "one", the diacritic is removed from the cover media and, in the same time, the index of the cover media is incremented only to read the next diacritic. The same process is proposed to hide bit value "zero", except that the "zero" will search for all the other seven diacritics instead of Fat'ha [20]. An example of this approach can be seen in Figure 3.

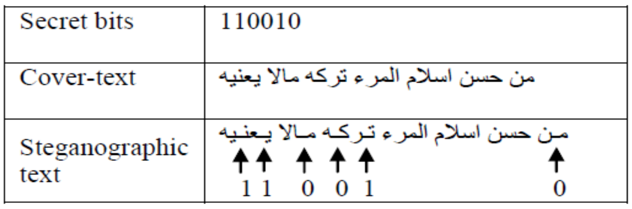


Cover Object	حَدَّثَنَا سُفْيَانُ عَنْ يَحْيَى
Secret Object	E7 (= 11100111)
Stego Object	حَدَثَنا سُفْيَان عَن يَحْيى

Figure 3. Hiding (E7) using diacritics approach

The main advantage of this method is found in its high capacity, since each Arabic letter is applicable for a diacritic and can possibly hide a secret. However, diacritics in standard Arabic language is optional, it is uncommon to be used nowadays. The security is degraded due to high probability of raising suspicions for an eavesdropper about the existence of secrets. Normally, the receiver has to have the original text so that the extracting algorithm can compare the diacritics in the stego object with the original cover object to extract the secrets.

Other proposed Arabic stego-systems use the redundant Arabic extension character "Kashida" for hiding secret bits using different methods such as high capacity stego-tool [17], e-text watermarking [21], utilization two diacritics for Steganography [22], and a stego-system for hiding text in images of personal computers [23]. The noted drawback of these ideas using the Kashida character is that it cannot be added at the beginning or ending of words. It can only be added between connected letters in the words. In the Kashida method presented in [4] and [7], the Kashida is linked to the pointed letters to hold secret bit one and the un-pointed letters to hold secret bit zero as shown in Figure 4, as an example. Using this link to pointed characters enhances the features of security and robustness. But, have drawbacks in capacity of the cover medium if the size of secret bits in the secret object is large.



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

Figure 4. Hiding secret bits using Kashida character

One Kashida representing secret bit zero and two consecutive Kashida letters when secret bit is one [10]. The Kashida is placed in all locations that can hold it. An interesting observation is that if the cover object is much longer than secret bits to be hidden, then, the stego object change is in the first few lines, which will be suspicious for an eavesdropper and might infer the existence of a hidden message in the text. This work presents merging an improvement over the study done by Adnan Gutub et al. [10] to utilize whitespaces for data hiding as well, making the system higher in capacity without suspicious degradation in the security as detailed next.

4. Proposed Idea and Result

The proposed idea for Arabic texts steganography is to improve the capacity of the Kashida extension stego-system by utilizing the whitespaces found between words in Arabic language for hiding. This work increases the full capacity without degrading the security. In our approach, hiding secret bits inside the Kashida and at the same time spaces between words are performed as follows. If the secret bit equals one, Kashida is added between the appropriate Arabic letters, i.e. as can be accepted, until all possible Kashida additions are considered. Then, before moving to the next word, whitespaces are to be used too; if the considered secret bit is found one, two consecutive whitespaces are added between words, otherwise (if secret bit is zero) normal single whitespace is kept as it is. Thus, representing secret bits equal ones is performed by inserting Kashida wherever possible and two whitespaces between the words, respectively. These two features are not put at the same time, i.e. if the word finishes from Kashida additions (as study of [17]) then whitespaces are to be added to hide secret bits before moving to next word, and so on. Note that the secret can be read from a text file then converted to binary bit representation to make it ready for insertion. The work assumes that the cover text is plain without any formatting or justifying. In fact, this work also assumes that the text does not have any "Kashida" nor extra whitespaces between words.

We tested our proposed Arabic text stego-system assuming the last 30 Surah (chapters) from Holy Quran [24] as our standard cover media benchmark. The normal statistics of this standard cover-media is listed in Table III. This proposed approach similar to the previous method presented on the study done by Adnan et al. [10] will be compared using the same benchmark. To illustrate the statistics of Table 2, observe Surah Al-Kawthar #108, which is the shortest Surah in the Holy Quran. This Surah (Al-Kawthar #108) has 10 words and 9 whitespace between words; which the Kashida possibilities in all words are in 20 locations and beside it 9 spaces of the total secret bits can that be embedded equal 29. The total secret bits that can be embedded in cover media equals the maximum number of Kashida in Surah plus the whitespace between words in Surah.

For comparison purposes, the percentage of secret bits that can be embedded is calculated for a certain cover (Surah) by dividing the total Secret bits by the total letters and space times hundred showing results listed in Table 3. This table values (Table 3) calculates the percentage of maximum number of secret bits that can be embedded in cover media using our proposed method. For example, consider Surah Al-Kawthar with total secret bits that can be embedded as 29 and total number of letters and whitespaces as 51 making the percentage that equals 56.86%. Similarly, the percentage of capacity using stego-system of the study done by Adnan et al. [10] is shown in Table 3, as using Kashida only. Surah Al-Kawthar, for example, is having total of possible Kashida additions as 20 and total number of letters as 42, the capacity percentage is 100*20/42 that equals 47.61%, as using Kashida only.

For a comparison example, assume a fixed secret text of one's (111…111) are to be hidden in Surah Al-Kawthar (Chapter # 108 of Holy Quran) using both methods, i.e. Kashida only (as previously proposed in [10]) and Kashida with whitespaces (our improvement). Figure 5 shows stego-output after implementing Kashida only approach as in [10].

Table 2. Statistics list of the last 30 Surah of the Holy Quran as testing benchmark

Name –number of Surah in the Holy Quran	**Number of letters**	**Number of words**	**Whitespace between words**	**Max number of Kashida in**	**Total secret bi can be embedd**	**Total of letters + whitespaces**
114. An-Nas	80	20	19	28	47	**99**
113. Al-Falaq	71	23	22	32	54	**93**
112. Al-'Ikhlas	47	15	14	23	37	**61**
111. Al-Masad	81	29	28	41	69	**109**
110. An-Nasr	79	19	18	35	53	**97**
109. Al-Kafirun	95	27	26	39	65	**121**
108. Al-Kawthar	42	10	9	20	29	**51**
107. Al-Ma`un	112	25	24	54	78	**136**
106. Quraysh	73	17	16	37	53	**89**
105. Al-Fil	96	23	22	56	78	**118**
104. Al-Humazah	133	33	32	60	92	**165**
103. Al-`Asr	70	14	13	31	44	**83**
102. At-Takathur	122	28	27	67	94	**149**
101. Al-Qari`ah	158	36	35	75	110	**193**
100. Al-`Adiyat	164	40	39	81	120	**203**
99. Az-Zalzalah	156	36	35	60	95	**191**
98. Al-Bayyinah	394	94	93	197	290	**487**
97. Al-Qadr	112	30	29	56	85	**141**
96. Al-`Alaq	281	72	71	125	196	**352**
95. At-Tin	156	34	33	79	112	**189**
94. Ash-Sharh	102	27	26	43	69	**128**
93. Ad-Duhaa	164	40	39	63	102	**203**
92. Al-Layl	312	71	70	148	218	**382**
91. Ash-Shams	249	54	53	122	175	**302**
90. Al-Balad	335	82	81	166	247	**416**
89. Al-Fajr	573	139	138	266	404	**711**
88. Al-Ghashiyah	378	92	91	193	284	**469**
87. Al-'A`la	293	72	71	137	208	**364**
86. At-Tariq	249	61	60	114	174	**309**
85. Al-Buruj	459	109	108	216	324	**567**

Table 3. Percentages of secret bits embedded by different methods

Name –number of Surah in the Holy Quran	**proposed technique (Kashida & whitespaces)**	**previous method [7] (Kashida ONLY)**	**Differenc e between the two methods**
114. An-Nas	47.47	35	12.47
113. Al-Falaq	58.06	45.07	12.99
112. Al-'Ikhlas	60.65	48.93	11.72
111. Al-Masad	63.30	50.61	12.69
110. An-Nasr	54.63	44.30	10.33
109. Al-Kafirun	53.71	41.05	12.66
108. Al-Kawthar	56.86	47.61	9.25
107. Al-Ma`un	57.35	48.21	9.14
106. Quraysh	59.55	50.68	8.87
105. Al-Fil	66.10	58.33	7.77
104. Al-Humazah	55.75	45.11	10.64
103. Al-`Asr	53.01	44.28	8.73
102. At-Takathur	63.08	54.91	8.17
101. Al-Qari`ah	56.99	47.46	9.53
100. Al-`Adiyat	59.11	49.39	9.72
99. Az-Zalzalah	49.73	38.46	11.27
98. Al-Bayyinah	59.54	50	9.54
97. Al-Qadr	60.28	50	10.28
96. Al-`Alaq	55.68	44.48	11.2
95. At-Tin	59.25	50.64	8.61
94. Ash-Sharh	53.90	42.15	11.75
93. Ad-Duhaa	50.24	38.41	11.83
92. Al-Layl	57.06	47.43	9.63
91. Ash-Shams	57.94	48.99	8.95
90. Al-Balad	59.37	49.55	9.82
89. Al-Fajr	56.82	46.42	10.4
88. Al-Ghashiyah	60.55	51.05	9.5
87. Al-'A`la	57.14	46.75	10.39
86. At-Tariq	56.31	45.78	10.53
85. Al-Buruj	57.14	47.05	10.09

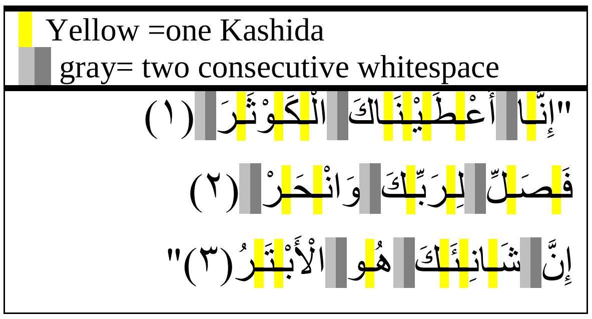

Yellow =one Kashida
gray= two consecutive whitespace
"إِنَّا أَعْطَيْنَاكَ الْكَوْثَرَ (١)
فَصَلِّ لِرَبِّكَ وَانْحَرْ (٢)
إِنَّ شَانِئَكَ هُوَ الْأَبْتَرُ (٣)"

Figure 6. Stego-output of hiding ones using Kashida and whitespaces as our modification proposal.

Figure 6 shows the results of applying our modification of using Kashida and whitespaces approach on the same cover media text. Observe that both methods, i.e. previous as in the study done by Adnan et al. [10] and our modification, will give same stego-output if all secret bits are zeros. Thus, the results become similar to the original media because the change occurs only if the bits are one. The comparison of the capacity, security, and robustness between the two methods are presented in the next section.

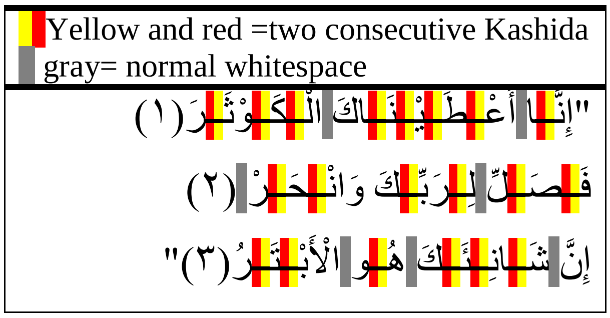

Yellow and red =two consecutive Kashida
gray= normal whitespace
"إِنَّا أَعْطَيْنَاكَ الْكَوْثَرَ (١)
فَصَلِّ لِرَبِّكَ وَانْحَرْ (٢)
إِنَّ شَانِئَكَ هُوَ الْأَبْتَرُ (٣)"

Figure 5. Stego-output of hiding ones using Kashida only as in [10].

5. Comparisons and Remarks

The merging improvement can be seen by increasing capacity as justified by the observation of number of ones and its percentage in secret bits. If the secret bits is having the number of ones much less than the number of zeroes, the cover media file size increase is showing small percentage and the stego-object is not changed much compared with the original cover text. Since the previous Kashida method in the study done by Adnan et al. [10] does not use whitespaces to hide secret bits, the capacity is always less. The previous method by Adnan et al. [10] is considered un-utilizing available capacity, as shown in Figure 7.

To demonstrate the comparison between the two methods, we investigated Surah Al-Falaq (Surah # 113 of Holy

Quran), which is showing interesting statistics (Table 2) and expressing percentages of secret bits that can be embedded using different methods with the highest percentage of difference (Table 3).

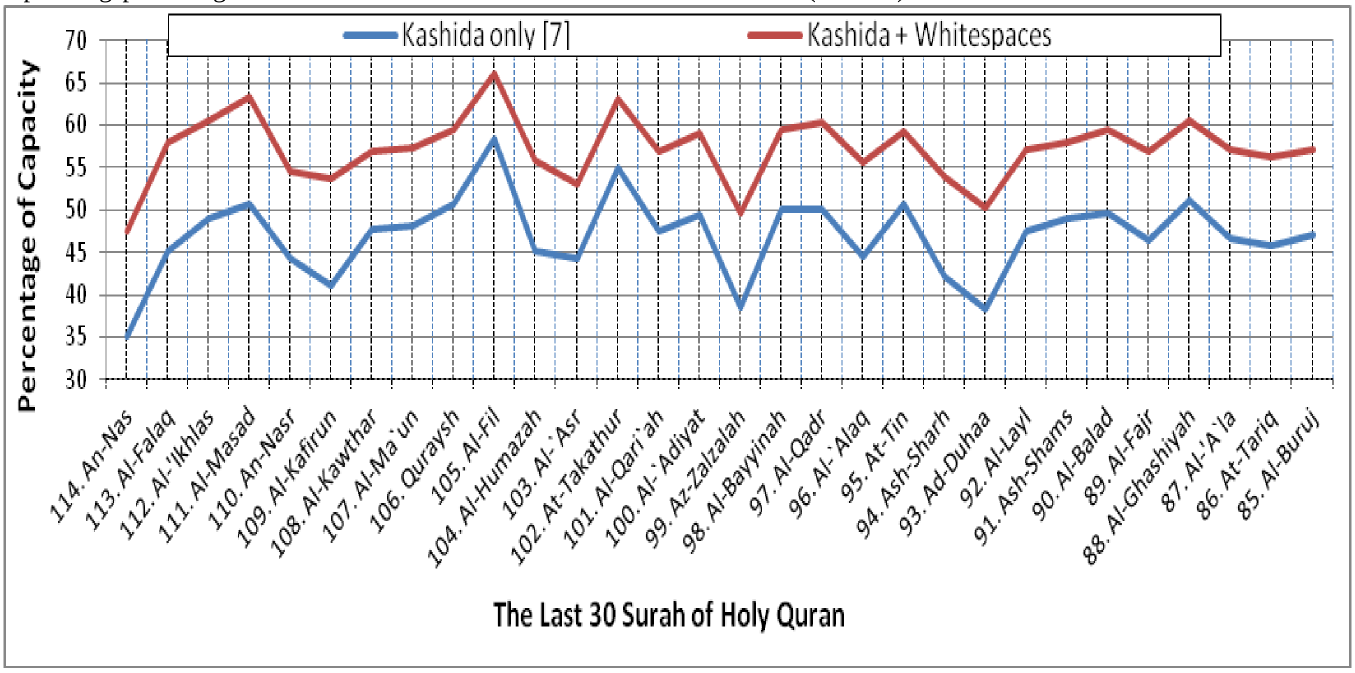
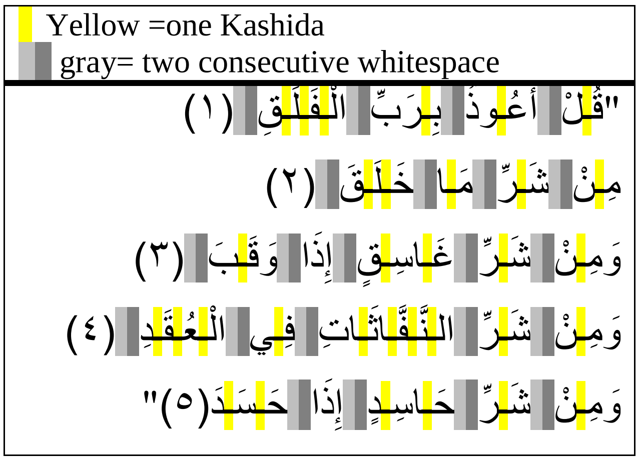


Figure 7. Percentage of capacity comparison between our merged proposal and Kashida only [10] approaches

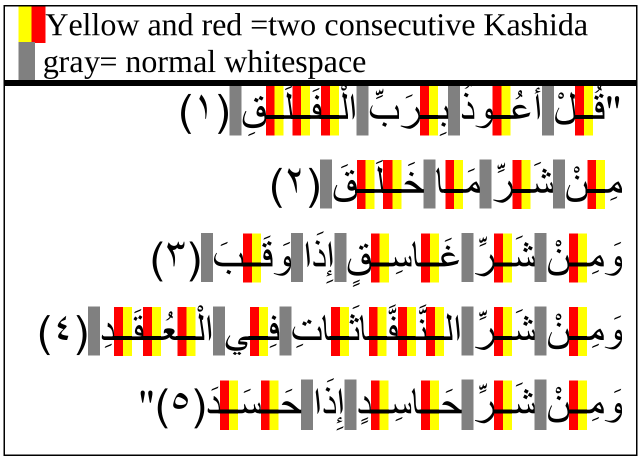


"قُلْ أعُوذُ بِرَبِّ الْفَلَقِ (١)
مِنْ شَرِّ مَا خَلَقَ (٢)
وَمِنْ شَرِّ غَاسِقٍ إِذَا وَقَبَ (٣)
وَمِنْ شَرِّ النَّفَّاثَاتِ فِي الْعُقَدِ (٤)
وَمِنْ شَرِّ حَاسِدٍ إِذَا حَسَدَ (٥)"

Figure 8. Applying our proposal using Kashida and whitespaces to hide secret bits ones

Using our proposed modified technique gave almost 13% increase in capacity over previous method by Adnan et al. in [10], such that we can embed 58% secret bits by using our improved stego-system of Kashida and whitespaces to hide bits of ones, as shown in Figure 8. This is compared to the same Surah Al-Falaq (Surah # 113 of Holy Quran), hiding secret in Kashida only (as previous method of Adnan et al. [10]) as shown in Figure 9, which kept normal whitespaces without any utilization resulting a capacity percentage of 45%.

Based on Table 2 the different percentage values for both methods as generated for the last 30 Surah of the Holy Quran and the difference is shown in Figure 10. The maximum difference in percentage between the two methods is almost 13% for Surah Al-Falaq # 113. This Surah value is found maximum because it has a lot of short words that can accept many Kashida letters as well as making more whitespaces between its words.



"قُلْ أعُوذُ بِرَبِّ الْفَلَقِ (١)
مِنْ شَرِّ مَا خَلَقَ (٢)
وَمِنْ شَرِّ غَاسِقٍ إِذَا وَقَبَ (٣)
وَمِنْ شَرِّ النَّفَّاثَاتِ فِي الْعُقَدِ (٤)
وَمِنْ شَرِّ حَاسِدٍ إِذَا حَسَدَ (٥)"

Figure 9. Applying proposal of hiding ones using Kashida only as in [10] to hide secret bits ones

On the other hand, the minimum capacity percentage of difference between both methods is found less than 8% for Surah Al-Fil # 105, as shown in Figure 10. Surah Al-Fil is showing the minimum difference because of its lowest number of whitespaces between words.

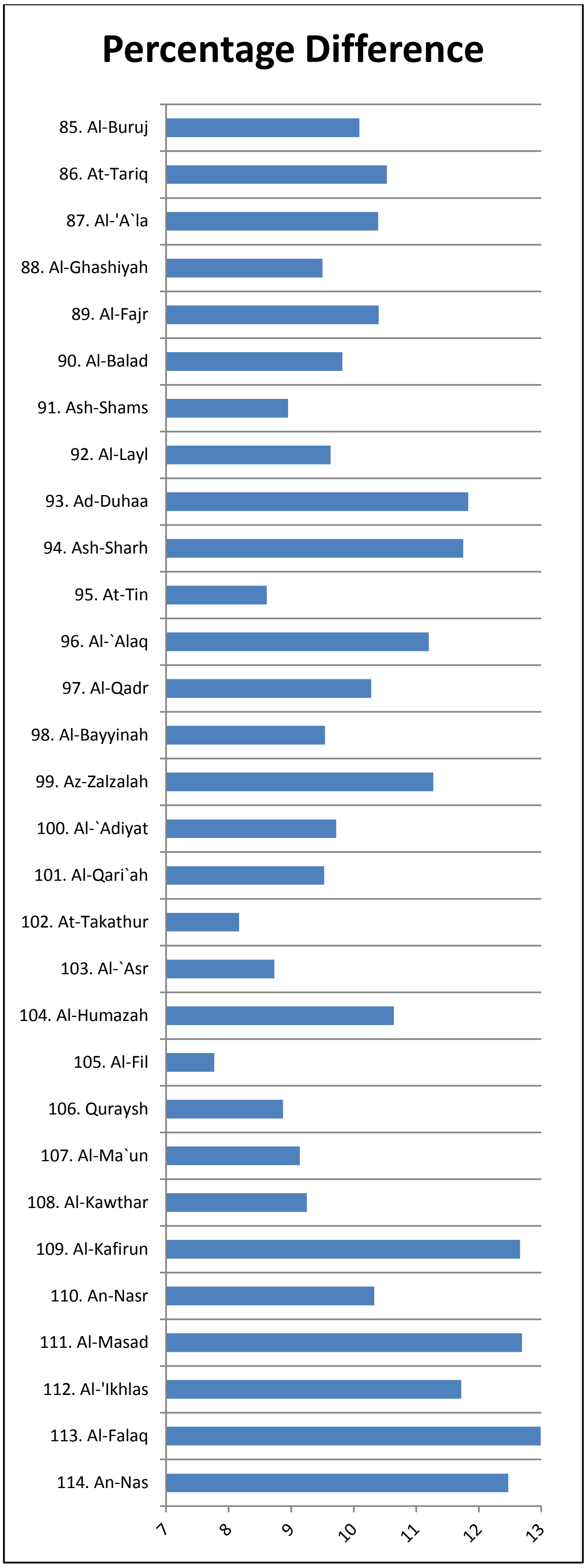


Figure 10. Difference in capacity percentage comparing improved merged method with previous single one from [10]

6. Conclusion

This paper presents a steganography method useful for Arabic text and other similar languages such as Persian and Urdu. The method improves the feature of hiding data within the Kashida character in Arabic script utilizing whitespaces between words. The method enhances the capacity and eliminate suspiciousness of an eavesdropper without inserting extension letter to represent zeros which is the most repeated in bits, but by using this new feature to hide secret bits within whitespaces between words, which will help to increase the capacity by 10% on average.

This approach enhanced and featured security, capacity, and robustness, which makes it useful to help users to exchange information through text documents and establish secure communication.

We would like to highlight some future works to be done in order to have a comprehensive steganography solution. We should use the extension character Kashida in the remaining un-used and applicable letters randomly to convert the attention of readers to have better security.

If somebody knows the algorithm, he 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. We plan to investigate using other formats for the secret. We look for utilizing other file types such as pdf and power points. For that, we need to convert those files to their binary representation.

We also need to add something to represent bits zeros that may be using some punctuation and any space between paragraphs to increase the capacity. The work is believed to be at the beginning stage opening new direction for more research in this security field.

Acknowledgment

We would like to thank Umm Al-Qura University (UQU) for hosting this research. Special thanks to the team within the course 1400519: Advanced Topics in Information Security, for motivating this research.

References

[1] D. Blumenthal, “Stimulating the Adoption of Health Information Technology”, *New England Journal of Medicine*, vol. 15, no. 360, pp. 1477-1479, April 2009.

[2] N. A. Al-Otaibi, A. Gutub, “2-Leyer Security System for Hiding Sensitive Text Data on Personal Computers”, *Lecture Notes on Information Theory*, vol. 2, no. 2, pp. 151-157, June 2014.

[3] M. H. Shirali-Shahreza, M. Shirali-Shahreza, “A New Approach to Persian/Arabic Text Steganography”, *IEEE/ACIS International Conference on Computer and Information Science (ICIS-COMSAR'06)*, pp. 310-315, 2006.

[4] A. Gutub, M. Fattani, “A Novel Arabic Text Steganography Method Using Letter Points and Extensions”, *WASET International Conference on*

Computer, Information and Systems Science and Engineering (ICCISSE), pp. 28-31, May 2007.

[5] A. Gutub, “Social Media & its Impact on e-governance”, *ME Smart Cities 2015-4th Middle East Smart Cities Summit*, December 2015.

[6] A. Gutub, “Exploratory Data Visualization for Smart Systems”, *ME Smart Cities 2015-4th Middle East Smart Cities Summit*, May 2015.

[7] A. A. A. Gutub, L. Ghouti, A. A. Amin, T. M. Alkharobi, M. K. Ibrahim, “Utilizing Extension Character ‘Kashida’ With Pointed Letters For Arabic Text Digital Watermarking”, *InSECRYPT*, July 2007.

[8] S. Al-Nofaie, M. Fattani, A. Gutub, “Capacity Improved Arabic Text Steganography Technique Utilizing ‘Kashida’ with Whitespaces”, *The 3rd International Conference on Mathematical Sciences and Computer Engineering (ICMSCE2016),* pp. 38-44, February 2016.

[9] A. Al-Nazer, A. Gutub, “Exploit Kashida Adding to Arabic e-Text for High Capacity Steganography”, *In Network and System Security, 2009. NSS'09. Third International Conference on*, pp. 447-451, IEEE, October 2009.

[10] A. A. A. Gutub, W. Al-Alwani, A. B. Mahfoodh, “Improved Method of Arabic Text Steganography Using the Extension 'Kashida' *Character”, Bahria University Journal of Information & Communication Technology (BUJICT),* vol. 3, no. 1, pp. 68-72, December 2010.

[11] F. Al-Haidari, A. Gutub, K. Al-Kahsah, J. Hamodi, “Improving Security and Capacity for Arabic Text Steganography Using 'Kashida' Extensions”, *In 2009 IEEE/ACS International Conference on Computer Systems and Applications*, pp. 396-399, May 2009.

[12] R. Anderson, “Security Engineering: A Guide to building Dependable Distributed Systems*”, Wiley Publishing*, 2001.

[13] J. Fridrich, “Steganography in digital media: principles, algorithms, and applications. Cambridge*”, Cambridge University Press*, 2009.

[14] T. Morkel, J.H. Eloff, M. S. Olivier, “An Overview of Image Steganography”, *Proceedings of the Fifth Annual Information Security South Africa Conference (ISSA2005),* June 2005.

[15] A. Cheddad, J. Condell, K. Curran, P. Mc Kevitt, “Digital image steganography: Survey and analysis of current methods”, *Journal of Signal Processing*, vol. 90, no. 3, pp. 727-752, March 2010.

[16] A. Odeh, K. Elleithy, M. Faezipour, “Steganography in Arabic text using Kashida variation algorithm (KVA)”, *IEEE Long Island Systems, Applications and Technology Conference (LISAT),* pp. 1-6, , May 2013.

[17] A. Abdul-Aziz Gutub, A. A. Al-Nazer, “High Capacity Steganography Tool for Arabic Text using 'Kashida'”, *The ISC Int'l Journal of Information Security (ISeCure)*, vol. 2, no. 2, pp. 109-120, July 2010.

[18] G. Abandah, F. Khundakjie, “Issues Concerning Code System for Arabic Letters”, *Dirasat Engineering Sciences Journal*, vol. 31, no. 1, pp. 165-177, 2004.

[19] M. A. Aabed, S. A. Awaideh, A. R. Elshafei, A. A. Gutub, “Arabic Diacritics Based Steganography”, *IEEE International Conference on Signal Processing and Communications (ICSPC 2007)*, pp. 756-759, November 2007.

[20] M. Al-Ghamdi, Z. Muzaffar, “*KACST Arabic Diacritizer”, Proceedings of the First International Symposium on Computers and Arabic Language*, pp. 25-28, 2007.

[21] A. A. A. Gutub, F. Al-Haidari, K. M. Kahsah, J. Hamodi, “e Text Watermarking: Utilizing 'Kashida' Extensions in Arabic Language Electronic Writing”, *Journal of Emerging Technologies in Web Intelligence (JETWI)*, vol. 2, no. 1, pp. 48-55, February 2010.

[22] E. M. Ahmadoh, A. A. A. Gutub, “Utilization of Two Diacritics for Arabic Text Steganography to Enhance Performance”, *Lecture Notes on Information Theory*, vol. 3, no. 1, pp. 42-47, June 2015.

[23] N. Alotaibi, A. Gutub, E. Khan, “Stego-System for Hiding Text in Images of Personal Computers”, *The 12th Learning and Technology Conference: Wearable Tech / Wearable Learning, Effat University,* April 2015.

[24] The Holy Quran: http://quran.ksu.edu.sa.