# Evaluating Robustness of Arabic CAPTCHAs

Suliman A. Alsuhibany[1],
[1]Computer Science Department, College of Computer,
Qassim University, Qassim 51477, Saudi Arabia
salsuhibany@qu.edu.sa

Mohammad Tanvir Parvez[2],
[2]Computer Engineering Department,
College of Computer,
Qassim University, Qassim 51477, Saudi Arabia
m.parvez@qu.edu.sa

Naseem Alrobah[3], Fatimah Almohaimeed[3],
Samar Alduayji[3]
[3]Information Technology Department,
College of Computer,
Qassim University, Qassim 51477, Saudi Arabia
331204315@std.qu.edu.sa, 331202328@std.qu.edu.sa,
daiejy@qu.edu.sa

***Abstract*—CAPTCHAs are applied on websites to differentiate between human users and automated programs, which indulge in spamming and other fraudulent activities. Since there are many websites that provide services in Arabic language, Arabic CAPTCHAs have been proposed by a number of studies. These CAPTCHAs rely on the distortion of text images rendering them unrecognizable to essentially pattern recognition techniques. However, the segmentation resistance evaluation against the proposed Arabic CAPTCHA schemes has not yet been done. Thus, this paper evaluates the robustness of the state of the art Arabic CAPTCHAs. Specifically, we showed that a number of proposed Arabic CAPTCHAs could be broken with an acceptable success rate. Moreover, a set of recommendation is derived in order to guide the design of robust Arabic CAPTCHAs.**

***Keywords— CAPTCHA; Arabic Script; Cyber security; Spam; Robustness; Internet security; Segmentation***

## I. INTRODUCTION

With the rapid increase of using Internet and E-services, such as e-government and e-health services, information security becomes more valuable than before. Users of these services are asked to register and fill forms. Authentication, which is the way of verifying the identity of a user, is one of the main aspects of securing information over the Internet. Therefore, online services are recommended to include an authentication method to verify if the user is a human or a robot.

A robot program can pose as a human and gain illegitimate access to resources and abuse online services. For example, bots can record thousands of votes automatically in online polls or sign up for thousands of accounts with different types of service providers. This, therefore, wastes a large volume of website resources and reduces the system performance [1].

HIPs, or Human Interactive Proofs, are challenges meant to be easily solved by humans while remaining too hard to be solved by automated attacks. CAPTCHA *(Completely Automatic Public Turing test to tell Computer and Human Apart)* is a HIP challenge that widely used in World Wide Web (WWW) to protect websites from the malicious programs [2].

Since bots are constantly improved to overcome common HIP challenges, the research community has developed many types of CAPTCHAs [1]. Nevertheless, most of the websites only adopt text-based CAPTCHAs that based on Latin script due to many advantages [4]. Indeed, CAPTCHAs based on Latin script have much vulnerability as stated in (e.g. [11, 12, 13]). Thus, they are threatened by malicious attacks.

A good-designed CAPTCHA should focus on two aspects: robustness and usability [4].The robustness of CAPTCHA refers to its resistance to adversarial attacks, while the usability is the ease to solve by the human. Balancing between the robustness and usability seems an important point to be considered. Indeed, accomplishing these aspects is very difficult [1, 3, 4].

Despite the fact that most of Arabic countries are moving towards implementing e-services and supporting of Arabic language in most of their online websites, they are still adapting Latin text-based CAPTCHAs. Therefore, using Arabic CAPTCHAs can be seen as a more usable alternative.

Arabic script is used in a wide variety of languages besides Arabic, including Persian, Malay and Urdu. In addition, Arabic script inherits many characteristics that have been proven to be effectively used in the CAPTCHA security [6]. According to a study conducted by Fidas et al. [5], a CAPTCHA composed of native language characters is considered as a viable and probably usable alternative to a Latin-based one. For this, Arab users, especially who merely use Arabic letters in all online activities, might find Latin CAPTCHAs inconvenient to recognize.

There have been few studies done to investigate the typed-text Arabic CAPTCHAs. The first study employing Arabic script in the CAPTCHA field was conducted in [6]. Khan et al. [7] improved the previous work by using different fonts and distortions. Moreover, a set of websites provide Arabic CAPTCHAs as a service that can be integrated into the user's website [8, 9, 18]. Although a security evaluation has been

**978-1-5090-5814-3/17/$31.00 ©2017 IEEE**

conducted to a number of these studies, evaluating the robustness level against the segmentation attack, which is an important potentially attack as stated in [11, 13], has not yet been accomplished.

Therefore, this paper evaluates the robustness level of the state of the art Arabic CAPTCHAs by using developed segmentation algorithms. We run each segmentation algorithm on five different schemes by conducting five independent experiments. The results showed that the existing proposed Arabic CAPTCHAs appear vulnerable to the segmentation attack. Hence, we suggest a set of recommendations that can be utilized as a guideline for designing a robust Arabic CAPTCHA.

This paper is organized as follows. Section 2 discusses related works. Section 3 provides a short overview of Arabic script. Section 4 presents the evaluation of the state of the art Arabic CAPTCHAs. Section 5 discusses the results. Section 6 presents some recommendations. Finally, Section 7 concludes the paper.

## II. Related Works

Generally speaking, most of studies on Arabic CAPTCHAs have been done in the typed text schemes, while very little works were reported for handwritten schemes. Therefore, this section highlights studies of both types of schemes.

### A. Typed Text-based Arabic CAPTCHAs

The first work employing Arabic script in the CAPTCHA field is presented in [6] that generates images of random meaningless Arabic words as CAPTCHA. In particular, the work reported in [6] presents an application of Persian/Arabic CAPTCHA, while the work in [14] applies Arabic CAPTCHAs for verifying spam SMS. Khan et al. [7] improved the previous work of typed-text Arabic CAPTCHA. Specifically, they exploited the limitations of Arabic OCRs in reading Arabic text by adding some background noise and using specific Arabic font types in CAPTCHA generation. Besides, the study in [17] proposed advanced Nastaliq CAPTCHA that provides essentially random meaningless Persian words which are close to Arabic words in terms of script.

The work presented in [8] is an open source application of Typed-text Arabic CAPTCHA. This application consists of fixed separated letters with some random noises. Additionally, BotDetect CAPTCHA generator [18] provides challenges with many different styles and languages in which Arabic language is one of these languages. Furthermore, an application of Typed-text Arabic CAPTCHA is presented in [9], where the CAPTCHA challenge has two dictionary words with two-crossed lines as distortions.

### B. Handwritten Text-based Arabic CAPTCHAs

The only contribution of implementing the handwritten Arabic CAPTCHA is in [10]. This study proposed a novel approach by exploiting the PAWs (Part of Arabic Words) and breaking Arabic connection rules by distortion-connected characters, and displacing them from baselines. However, since our work is focusing on evaluating the robustness of typed text-based Arabic CAPTCHAs, the evaluation of the handwritten Arabic CAPTCHA is out of the scope of this paper.

## III. Arabic Script: An Overview

This section explains the characteristics of Arabic language in terms of writing direction, shapes and recognition.

Arabic script has 28 letters [16]. Despite Arabic numbers are written from left to right, Arabic letters are written from right to left. A word includes a set of different letters. Moreover, the letters are connected during writing both in printed and handwritten texts. Besides, some letters are written overlapped, one letter is on top of another, as shown in Figure 1(a). Also, sometimes when a letter is written in the middle of the word, it partially disappears under its predecessor, as shown in Figure 1 (b).


(a) (b)

Fig. 1. Illustrations of (a) overlapping case and (b) partial disappearance of a letter in Arabic script.

There are a maximum of four different forms of writing a letter in Arabic script depending on its position in the word. For instance, the form of the letter *Mem* can be either "م", "ـم", "ـمـ", or "مـ", where it can be an isolated letter, at the end of the word, between two letters or in the beginning of the word, respectively. Moreover, several Arabic characters have similar shapes, e.g. ب ت ث ن, ج ح خ, ص ض, ع غ, ط ظ, د ذ ر ز و, ف ق. As stated in [16], this similarity confuses OCR to recognize characters correctly. These features, accordingly, lead to apply the Arabic language in the text of CAPTCHA. However, the vulnerability to such attacks likes the segmentation attack has not yet been tested.

## IV. Evaluation Of the State of The Art

This section evaluates the state of the art Arabic CAPTCHA schemes. In particular, many studies have been published [6, 7, 17], in addition to these websites [8, 9, 18] where Arabic script is suggested to be utilized in CAPTCHA tools. Even though the segmentation attack was discussed extensively in the text-based CAPTCHAs field as a key threat on CAPTCHAs, current Arabic CAPTCHAs have not been evaluated against this attack.

Moreover, as stated in [13] "*if breaking a CAPTCHA challenge can be reduced to the problem of recognizing its characters, then this CAPTCHA is effectively broken.*" Based on this, we developed a segmentation process in order to evaluate the existing Arabic CAPTCHAs. Segmentation process in general was used in a number of previous works (like in [11, 12, 13]) for Latin CAPTCHAs, with the goal of extracting characters from CAPTCHA images. In the following, we discuss the methodology for evaluating the CAPTCHA schemes.

### A. Methodology

The methodology that we follow can be divided into two consecutive steps: Pre-processing and Segmentation. These steps are described as follows.

*1) Pre-processing:* This step is important in order to remove the added distortion. In particular, the image is binarized to separate the text from the background noise. To do this, we used a commercial software called CAPTCHA Solving Software (GSA Captcha Breaker) [15] to accomplish the pre-processing step. It seems empirically that this software is a powerful method for removing the distortion of targeted schemes, as will be demonstrated in the following section.

*2) Segmentation:* Once the distortions are removed, the segmentation process is applied. Specifically, it is the process where the location of each character in a target object is found correctly and in the right order [11]. For this, we developed two Arabic script segmentation algorithms: vertical segmentation algorithm and joint point segmentation algorithm [10]. For the first algorithm (Algorithm 1), after segmenting targeted scheme's samples, we classify the results into four groups:

- **Not segmented** word: It means that all characters are not segmented and the algorithm could not find the joint points between the letters.
- **Incorrectly segmented** word: It means that either one letter is broken into two segments, or it is not in the right location (i.e. right order).
- **Partially segmented** word: It means that not all character-blocks are segmented (i.e. there are two or more letters are still connected).
- **Completely segmented** word: It means that all letters are segmented correctly and in the right order.

For the second algorithm (Algorithm 2)**,** the results are classified into three groups:

- **Un-segmented character**s: It means that the character is not segmented.
- **Segmented character**s: It means that the character is fully segmented.
- **Over-segmented characters**: It means that the character is segmented into parts. For example, the character NOON (ن) segmented in the middle.

## B. *Targeted Schemes (Using Algorithm 1)*

This section presents the schemes that are evaluated in this paper using *vertical segmentation algorithm*. The targeted schemes are as follows.

*1) Scheme 1 [6]:* This scheme is the first work in utilizing Arabic script as the CAPTCHA text. The observed features of this scheme are: fixed background color, a single font type that has the same style, size and color, constant location of the character string, and simple distortions such as constant location and number of straight lines. Figure 2 shows a sample from this scheme.

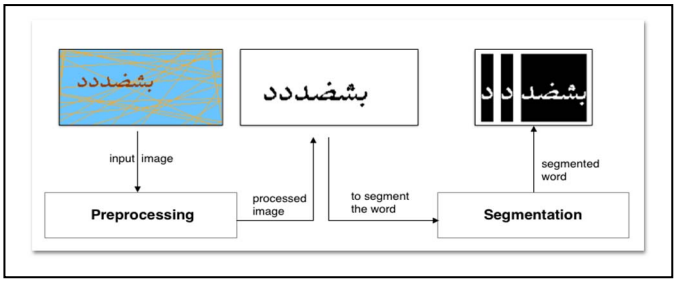



Fig. 2. A sample of scheme 1 [6] and its evaluation steps.

Unfortunately, we found only one sample from this scheme; although of this, we did evaluate the sample and could partially segment it. Accordingly, we observed many security flaws in this scheme. For example, it uses fixed and predictable patterns in the presentation of CAPTCHA images, which makes it vulnerable to attacks.

*2) Scheme 2 [7]:* This scheme extends the work in [6] by adding more features. For example, 52 font types and different font sizes are chosen randomly. Also, more than one distortion types are added, as well as the random variation of the position of CAPTCHA text.

We collected 10 samples which were available so far for the evaluation step. A sample of this scheme and its evaluation steps are demonstrated in Figure 3.

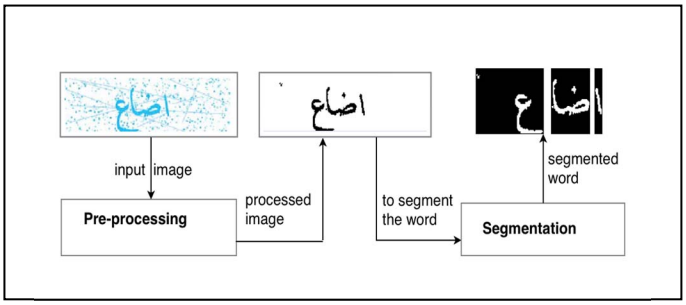



Fig. 3. A sample of scheme 2 [7] and its evaluation steps.

We fed the samples from [7] to the pre-processing step with the following parameters: the threshold value of binarization was 75%, while connected components that contain less than 10 pixels were removed. After that, we fed all samples to our segmentation algorithm, sequentially. We found that 10% of the samples were incorrectly segmented, whilst 90% were partially segmented.

Based on these results, we observed the following security weaknesses. The scheme was applying a fixed background color and foreground text color. Besides, the distortion color was different from text color. Also, there were only two types of distortion lines that were in form of straight lines or/and arcs. Finally, the number of lines was fixed in each level (10 in easy level, 20 in medium level and 30 in hard level).

*3) Scheme 3 [17]:* This scheme mainly consists of random meaningless Persian/Arabic words. A sample of this scheme and its evaluation steps are demonstrated in Figure 4.

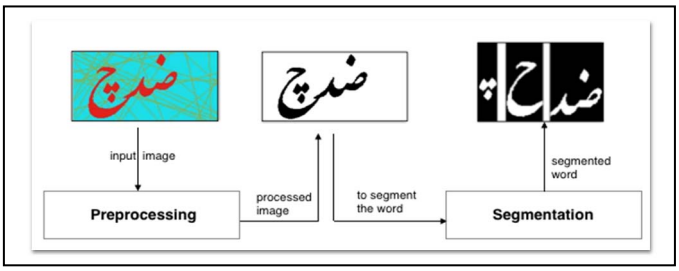



Fig. 4. A sample of scheme 3 [17] and its evaluation steps.

Unfortunately, we found only one sample from this scheme; although we did evaluate the sample and could partially segment it. Consequently, we observed that the main weakness in this scheme might be that some special Persian characters are applied which do not exist in Arabic characters, e.g. "چ" with 3 dots as shown in Figure 4.

*4) Scheme 4 [8]:* This scheme basically consists of a fixed number of characters with some random noises. We collected 51 samples from this scheme for the evaluation purpose. A sample of this scheme and the evaluation steps are demonstrated in Figure 5.

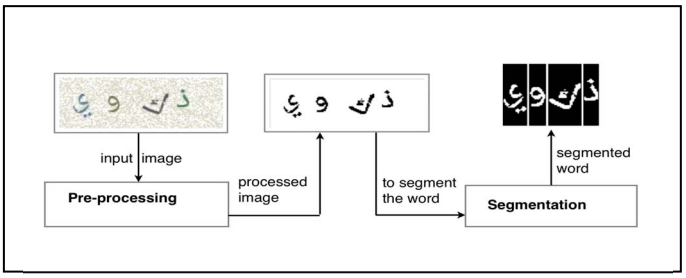



Fig. 5. A sample of scheme 4 [8] and its evaluation steps.

We then fed these samples to the pre-processing step with the following parameters: the threshold value of binarization was 80%. After that, we fed all samples to our segmentation algorithm, sequentially. The results of the segmentation step are that 49.02% of samples were incorrectly segmented, 3.92% of samples were partially segmented, and 47.06% of samples were completely segmented.

Based on these results, we observed the following security weaknesses. The scheme applied a fixed number of letters. Besides, it used different colors for the content and noise. Also, it used a fixed font type. Finally, the applied salt and pepper noise as a background distortion can be easily removed in our experiment.

*5) Scheme 5 [18]:* This scheme is generated by *BotDetect generator* that provides challenges with many different styles and languages, in which Arabic language is one of them.

We collected 30 different samples that are generated based on Arabic script for the evaluation step. A sample of this scheme and its evaluation steps are demonstrated in Figure 6.

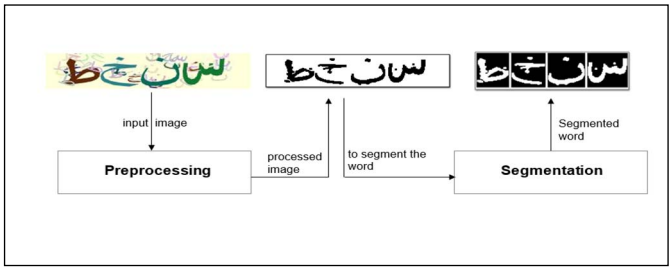



Fig. 6. A sample of scheme 5 [18] and its evaluation steps.

We then fed these samples to the pre-processing step with the following parameters: the threshold value of binarization was between 30% and 80%, while removing objects was depending on the applied distortion. After that, we fed all samples to our segmentation algorithm, sequentially. The results of the segmentation step are that 53% of samples were incorrectly segmented, 23% of samples were partially segmented, 10% of samples were completely segmented, and 13% of samples were not segmented at all.

Table I summarizes the results of applying vertical segmentation algorithm on all schemes.

TABLE I. RESULTS OF ALGORITHM 1.

| Segmentation Category | Schemes | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| **Not segmented** | NA | 0% | NA | 0% | 13% |
| **Incorrectly segmented** | NA | 10% | NA | 49% | 53% |
| **Partially segmented** | NA | 90% | NA | 3.9% | 23% |
| **Completely segmented** | NA | 0% | NA | 47% | 10% |

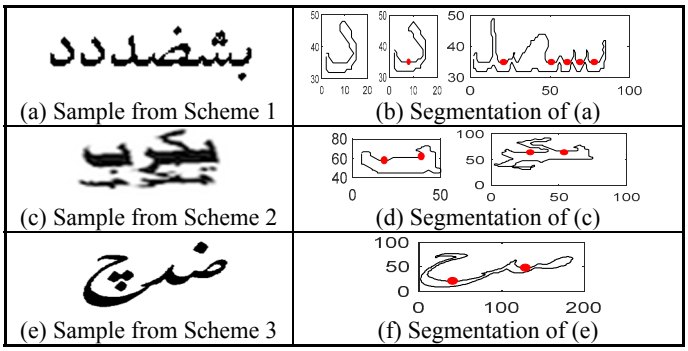
## C. Targeted Schemes (Using Algorithm 2)

This section presents the results for schemes that are evaluated using *joint point segmentation algorithm*. Since we aim to segment Arabic words into characters, schemes 4 and 5 will not be evaluated as these schemes contain isolated characters only (not words). The results of applying Algorithm 2 on targeted schemes are as follows.

*1) Scheme 1 [6]:* As mentioned in the previous section, we found only one sample from this scheme. Thus, the result of applying Algorithm 2 on this sample is shown in Figure 7 (a-b).

*2) Scheme 2 [7]:* We fed the same samples that were used in Algorithm 1 to Algorithm 2. Figure 7 (c-d) shows the result of applying Algorithm 2 on a sample of this scheme.

*3) Scheme 3 [17]:* As mentioned in the previous section, we found only one sample from this scheme. Thus, the result of applying Algorithm 2 on this sample is shown in Figure 7 (e-f).



(a) Sample from Scheme 1 (b) Segmentation of (a)

(c) Sample from Scheme 2 (d) Segmentation of (c)

(e) Sample from Scheme 3 (f) Segmentation of (e)

Fig. 7. Illustrations of samples from different schemes segmented using Algorithm 2 (only character main-bodies are shown). The filled circles denote the segmentation points.

Table II summarizes the results of applying *joint point segmentation algorithm*, where the results are very encouraging, especially for Scheme 2 (which has the most number of samples). For Scheme 2 samples, 83% of the characters are fully segmented, which shows the vulnerability of the scheme. For schemes 1 and 2, we could not test on many samples, although the results are also excellent.

TABLE II. SUMMARIZED RESULTS OF SEGMENTATION BY ALGORITHM 2.

| Segmentation Category | Schemes | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| **Un- segmented** | 0% | 0% | 0% | NA | NA |
| **Completely segmented** | 67% | 83% | 100% | NA | NA |
| **Over-segmented** | 33% | 17% | 0% | NA | NA |

## V. DISCUSSION

Due to the fact that most of studies that proposed Arabic CAPTCHAs [6, 7, 17] did not make their generator publically available, we could not collect much samples in favor of evaluating the robustness of the proposed schemes. Nevertheless, investigating the design principles of the proposed schemes, in addition to the presented samples in the literature, assists us to draw some recommendations (discussed in Section VI) for designing CAPTCHA schemes resistant to the segmentation attack.

On contrast, the schemes proposed by the websites in [8, 18] allowed us to collect many samples, where the generated samples can be controlled in terms of different settings. For example, in [18], the user can choose the background color, text color, size of the text, etc. Based on the achieved results using Algorithm 1, the proposed scheme in [18] appears to be more robust (i.e. 10% of samples were completely segmented) against the segmentation attack, followed by the scheme in [8] (i.e. 47.06% of samples were completely segmented). The possible reason behind this may be that the generated samples in [8] seem not applying the collapsing approach (i.e. removing the space between characters) on characters, while samples in [18] were applying the collapsing approach to some extent.

Furthermore, using Algorithm 2 for evaluating schemes 1, 2, and 3 yields interesting results in which the segmentation attack on Arabic scripts can be further improved. Besides, we test the collected samples on the "Color Filling Segmentation" (CFS) algorithm [11] that is effectively like using a distinct color to flood each component. Unfortunately, this algorithm was not effective for schemes 1, 2, and 3 due to the fact that the letters are connected during writing both in printed and handwritten texts [16], as shown in Figure 8. However, this algorithm can intuitively be used in schemes 4 and 5 as they contain a set of separate characters.

Fig. 8. A sample CAPTCHA image after applying CFS algorithm.

Furthermore, the usability aspect is one of the fundamental features in generating a CAPTCHA [19]. For this, most of the state of the art Arabic CAPTCHAs did not take into account this important aspect. For example, studies in [6, 7, 17] did not conduct experimental studies to evaluate the usability of the proposed schemes. Nonetheless, the work in [7] did such survey to evaluate the satisfaction of the users, but it did not investigate the impact of the implemented distortions on the usability. Therefore, although this aspect is out of the scope of this paper, we will extend our work to cover this aspect.

## VI. RECOMMENDATIONS

In this section, a number of recommendations have been derived empirically which can be used as a guideline to improve the robustness of typed-text Arabic CAPTCHAs. These recommendations are related to color, distortion, character set, number of letters, string style, string position, font type, font size, and image quality.

### A. Color

Using a wide range of colors for both the background and foreground text, which have low contrast and similar darkness and lightness, can complicate the process of binarization step. By following this, an image can be either completely black or white. Accordingly, it is recommended to use a wide range of colors for both background and text with low contrast and similar darkness and lightness.

### B. Distortion (curves & dots)

Background distortion can increase the robustness of CAPTCHA when the pre-processing algorithms cannot distinguish them from the text. This can be achieved by using distortion that have same color and similar shape and size of (target object) the foreground text. In this case, removing the distortion may lead to the removal of the original letters' dots and/or distorting the letters' strokes. Thus, it is recommended to apply dots as noises similar in sizes and shapes to the letters' dots. In addition, curved lines having random shapes and lengths can be used. These lines should intersect CAPTCHA words, where the curve's stroke width should be similar to that of letter strokes.

### C. Character set

As stated in [12], using a large complex character sets that includes upper and lower case letters and digits for English

CAPTCHAs is not effective in increasing the robustness, although the scheme may confuse the human reader. Since Arabic is cursively written, using words that consist of letters with digits in-between the letters may create gaps between the characters; thus facilitating the segmentation process. Consequently, using only Arabic letters is recommended, whereas using a large character set that may have, for instance, letters and numbers might reduce the usability.

### *D. Number of characters*

When a CAPTCHA word has a fixed number of characters, character positions can be easily predicted. However, using a random number of characters can make this prediction complicated for the segmentation process.

### *E. String style: cursive or separate*

Arabic text string in cursive style is more resistant to the segmentation attacks than using isolated letters. When the word has a larger number of connected letters, the difficulty in segmenting and detecting the correct joint points between the letters can be increased.

### *F. Text position and rotation of baseline*

The rotation degree of baseline can increase the resistance against the segmentation attacks that depends on the projection (i.e. vertical segmentation). Therefore, it is recommended to use a random text position in the image and rotation of the baseline with random degrees.

### *G. Font types*

There are different fonts for Arabic script, each of them differs in the design of the letters' strokes and Ligatures. A letter stroke is drawn differently from font type to another, for instance, the letter "غـ" is drawn as filled or hollow depends on the font type. After removing the distortion this may confuse the machine if it is "فـ" or "غـ". Accordingly, when using different font types and different font sizes, defining a threshold value as a distance (between letters) to segment the characters will be more difficult. The distance between two letters or the letter and its dot(s) varies from one font to another.

### *H. Font size*

It is recommended to apply different font sizes in order to avoid any prediction to the letter's pixel counts, as explained in [11].

## VII. CONCLUSION

Several Latin scripts based CAPTCHAs have been broken, which motivated researchers to propose Arabic script based CAPTCHAs. Although most of these proposed CAPTCHAs might be evaluated well against such attacks, they were not evaluated against the segmentation attack. As a result, this paper evaluated the robustness of reported Arabic CAPTCHAs against the segmentation attack. Using Arabic character segmentation algorithms, we could evaluate the available samples in the literature. The results showed that current schemes can be vulnerable to the segmentation attack. Moreover, a number of recommendations have been discussed that new CAPTCHA generators may follow as a designing guide.

The authors are working to build a robust Arabic CAPTCHA generator based on the recommendations discussed in this paper.


## ACKNOWLEDGMENT

The authors would like to thank Qassim University for supporting this research.


## REFERENCES


[1] N. Roshanbin, , "Interweaving Unicode, Color, and Human Interactions to Enhance CAPTCHA Security ," Ph.D. dissertation, Dept. Elect. and Comput. Eng, Univ. of Alberta, Edmonton, AB, 2014.

[2] L. von Ahn, M. Blum and J. Langford, "Telling humans and computers apart automatically", *Communications of the ACM*, vol. 47, no. 2, pp. 56-60, 2004.

[3] S. A. Alsuhibany, "A Benchmark for Designing Usable and Secure Text-Based Captchas, " *International Journal of Network Security & Its Applications*, vol. 8, no. 4, pp. 41–54, Jul. 2016.

[4] J. Yan and A. El Ahmad, "Usability of CAPTCHAs or usability issues in CAPTCHA design, " in *Proceedings of the 4th symposium on Usable privacy and security - SOUPS '08*, 2008, pp. 44–52.

[5] C. A. Fidas, A. G. Voyiatzis, and N. M. Avouris, "On the necessity of user-friendly CAPTCHA," in *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*, 2011, pp. 2623–2626.

[6] M. H. Shirali-Shahreza and M. Shirali-Shahreza, "Persian/Arabic Baffletext CAPTCHA," *Journal of Universal Computer Science*, vol. 12, no. 12, pp. 1783-1796, 2006.

[7] B. Khan, K. Alghathbar, M. K.Khan, A. M.AlKelabi, and A. Alajaji, "Cyber security using arabic captcha scheme, " *Int. Arab J. Inf. Technol., vol.* 10, no. 1, pp. 76-84, 2013.

[8] M. Anini, "Arabic CAPTCHA | عركابتشا - عربية كابتشا," ArCaptcha - Arabic CAPTCHA | عركابتشا - عربية كابتشا. [Online]. Available: http://arcaptcha.anini.me.

[9] "*العربية حسوب كابتشا,* " . *[Online]. Available: http://captcha.hsoub.com.*

[10] S. A. Alsuhibany and M. T. Parvez, "Secure Arabic Handwritten CAPTCHA Generation Using OCR Operations, " *15th International Conference on Frontiers in Handwriting Recognition (ICFHR-2016)*.

[11] J. Yan and A. El Ahmad, "A Low-cost Attack on a Microsoft CAPTCHA," in *Proceedings of the 15th ACM conference on Computer and communications security* (CCS '08)*,* 2008*,* pp. 543-554.

[12] E. Bursztein, M. Martin and J. Mitchell, "Text-based CAPTCHA strengths and weaknesses," in *Proceedings of the 18th ACM conference on Computer and communications security - CCS '11*, pp. 125-138.

[13] A. El Ahmad, J. Yan and L. Marshall, "The robustness of a new CAPTCHA, " in *Proceedings of the 3rd European Workshop on System Security - EUROSEC '10*, 2010, pp. 36-41.

[14] M. S. Shahreza, "Verifying Spam SMS by Arabic CAPTCHA," in *2nd IEEE International Conference on Information and Communication Technologies (ICTTA'06)*, 2006, pp. 78-83.

[15] "GSA Captcha Breaker," *GSA - Softwareentwicklung und Analytik GmbH*, 2016. [Online] Available: https://captcha-breaker.gsa-online.de/

[16] S.A. Sattar, S. Haque, M. K. Pathan and Q. Gee, "Implementation challenges for nastaliq character recognition," *Wireless networks, information processing and systems,* 2008*,* pp. 279-285.

[17] M. H. Shirali-Shahreza and M. Shirali-Shahreza, "Advanced Nastaliq CAPTCHA," *7th IEEE International Conference on Cybernetic Intelligent Systems*, London, 2008, pp. 1-3.

[18] "BotDetect CAPTCHA Demo – Features," *Captcha.com*, 2016. [Online]. Available: https://captcha.com/demos/features/captcha-demo.aspx.

[19] S. A. Alsuhibany, "Optimising CAPTCHA Generation," *2011 Sixth International Conference on Availability, Reliability and Security*, Vienna, 2011, pp. 740-745.