

A BENCHMARK FOR DESIGNING USABLE AND SECURE TEXT-BASED CAPTCHAS

Suliman A. Alsuhibany

Computer Science Department, College of Computer, Qassim University, Buridah, Saudi Arabia

ABSTRACT

An automated public Turing test to distinguish between computers and humans known as CAPTCHA is a widely used technique on many websites to protect their online services from malicious users. Two fundamental aspects of captcha considered in various studies in the literature are robustness and usability. A widely accepted standard benchmark, to guide the text-based captcha developers is not yet available. So this paper proposes a benchmark for designing usable-secure text-based captchas based on a community driven evaluation of the usability and security aspects. Based on this benchmark, we develop four new text-based captcha schemes, and conduct two separate experiments to evaluate both the security and usability perspectives of the developed schemes. The result of this evaluation indicates that the proposed benchmark provides a basis for designing usable-secure text-based captchas.

KEYWORDS

Text-Based CAPTCHA, security, usability, benchmark

1. INTRODUCTION

Distinguishing computers from humans is a key issue for website security. For example, eBay must stop bots from flooding its website with scams and Gmail must prevent abuse by automated spammers. A captcha¹, which is a Completely Automated Public Turing test to tell Computer and Human Apart [1], allows websites to make an automatic assessment.

There are different types of captchas [2, 3] such as text-based, video-based and image-based, to name but a few. Due to its many advantages [4], the text-based captcha is the most commonly deployed type currently used by websites. The text-based type prompts users to recognize text, which state-of-the-art text recognition programs cannot do. In this paper, the term captcha refers to text-based schemes only.

A well designed captcha should be robust and usable. Robustness refers to its strength in resisting adversarial attacks, while usability is the ease with which humans pass its challenges [5].

Several works have been proposed in the literature to evaluate these aspects. Robustness has attracted considerable attention in the research community (e.g. [6, 7, 8, 9, 10]) and usability has been studied from many different angles; for instance, from a functional level focusing on differences in expected accuracy and response time [1, 2, 3, 4]. There has also been systematic analysis of usability issues that should be considered in the design [5], discussion on designing an optimizer [11] that can optimize the generated text from the usability issues observed in [5], and the visual features used in captchas and how they interact in order to understand how they affect the complexity of the captcha [13].

¹ For readability purpose, we write this acronym in lower case.

Due to a trade-off between the security and usability aspects when designing a captcha, the proposed security features tend to cause most of the usability issues. Consequently, a widely accepted standard benchmark for the design of a usable-secure captcha, which developers can use for guidance, is not yet available. This paper proposes a benchmark for designing a usable-secure captcha based on a community driven evaluation of security and usability features.

The proposed benchmark depends on two main criteria identified as distinctive security features in most widely used captcha schemes [9, 12, 13]: core features and distortion. For each criterion, a recommended level is specified based on empirical results evaluated in the literature.

We develop four new captcha schemes, based on the proposed benchmark, with predictable results in terms of accuracy and saving time. These schemes are: Easy and Secure, Fast, Hard, and Annoying. Then, we designed and built a captcha generator and conducted experiments in order to evaluate the robustness and usability aspects of these schemes. The results of these experiments validated our benchmark in the analysis of features and our recommendations for a future captcha design.

The first contribution of this paper is the platform that can help to design secure-usable captchas. Our second contribution is the four new derived captcha schemes which systematically reflect the proposed benchmark through experimental evaluations.

The rest of this paper is organized as follows. Section 2 discusses the related work; Section 3 gives an overview of the proposed benchmark which is explained in Section 4. Section 5 describes the derived new captcha schemes. The evaluation of these schemes is presented in Section 6, the discussion is presented in Section 7 and Section 8 concludes the paper.

2. RELATED WORK

Evaluation of the robustness and usability of captchas attracted considerable attention in the research community. For instance, the study in [6] demonstrated that most captcha schemes are broken if they can be reliably segmented. Huang et al. [7] extended this study by developing an efficient segmentation algorithm for captchas with line cluttering and character warping. A systematic evaluation methodology in [9] applied to 15 current captcha schemes; found that 13 were vulnerable to automated attacks. Based on these results, the study divided the automated captcha-solving process into five generic steps: pre-processing, segmentation, post-segmentation, recognition, and post-processing. The study in [10] explored object recognition in clutter by testing their object recognition techniques on two examples of visual captchas Gimpy and EZ-Gimpy, where they identified the word in an EZ-Gimpy image with a success rate of 92%, and the requisite three words in a Gimpy image 33% of the time. Recently, studies in both [19] and [20] demonstrated a generic attack that breaks a wide range of text captchas, which has achieved a good success rate.

A number of studies have been conducted by researchers into captcha usability, for example, several captcha generators with a user-friendly design, along with an examination of the effect of different text distortion techniques on the readability of captchas, have been discussed in [4]. Yan and El Ahmad [5] proposed a three-dimensional framework for examining the usability of captchas. However, they did not discuss how to improve the usability issues related to character confusion. Hence, Alsuhibany introduced in [11] an optimization algorithm that can improve the usability issues related to character confusion without impairing the security level. A large scale study by Bursztein et al. [12] assessed how well captchas achieve the goal of making them more usable, and they concluded that captchas are often quite difficult for humans to read. A recent study in [13] described how they designed two new captcha schemes for Google that focus on maximizing the usability aspect, and they achieved a 95% human accuracy rate and a 6.7% improvement in new schemes.

For usable-secure captchas, the authors in [14] studied how to balance security and usability in video-based captchas. This topic is revisited in [15] and is extended to all sorts of moving captchas.

3. AN OVERVIEW OF THE PROPOSED BENCHMARK

As mentioned recently, a good captcha should be both usable and secure. The usability is defined, as quoted from Jakob Nielsen [16], by the following five quality components:

- 1) **“Learnability:** How easy is it for users to accomplish basic tasks the first time they encounter the design?
- 2) **Efficiency:** Once users have learned the design, how quickly can they perform tasks?
- 3) **Memorability:** When users return to the design after a period of not using it, how easily can they re-establish proficiency?
- 4) **Errors:** How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- 5) **Satisfaction:** How pleasant is it to use the design?”

A captcha requires users to intuitively and easily understand and remember, so requires components that can be learned and remembered. The usability criteria – accuracy, response time and satisfaction of using a scheme – are applied to address efficiency, errors and satisfaction respectively [5].

The captcha security should be segmentation-resistant [6] and recognition-resistant [9], where the segmentation process is the separation of two sequences of characters into individual characters, and the recognition process is the identification of those characters.

There are a number of security features that have been utilized in the literature [9, 13], for example, basic text features such as the font size and color, anti-segmentation techniques such as collapsing, and anti-recognition techniques such as character rotation. These features can cause many usability issues, such as readability which can be regulated by the use of distortion methods and by how much distortion is applied to text.

In this paper, we proposed the following two-dimensional benchmark for designing usable-secure captchas:

A. Core Features

This dimension demonstrates the content, such as the string length, and the presentation, such as the font size.

B. Distortion

This dimension demonstrates the form of distortion methods applied in a captcha, either anti-segmentation methods, such as character overlapping, or anti-recognition methods, such as rotation, or both.

Additionally, a recommended level for each single feature that has been evaluated empirically in the literature is determined in the proposed benchmark.

4. USABLE-SECURE CAPTCHAS BENCHMARK

This section discusses each dimension mentioned in Section 3 and the recommended parameter levels, as shown in Table 1.

Table 1. Usable-Secure Captchas Benchmark

Category	Parameter	Recommended level	
Core features	Content	Character set	- Pseudo-word - Digits - Upper-Lower case letters
		String length	3 >Random < 9 characters
	Presentation	Font type	Sans-serif
		Font size	Random
		Font families	Random
Foreground-Background colours	Random, but avoiding yellow for the foreground and white for the background, or vice-versa.		
Distortion	Anti-segmentation techniques	Character overlapping	Non-negative gap width (i.e. Gap ≥ 0)
		Random dot-size	$1 \leq \text{pixel dots} \leq 10$
		Random dot-counts	2000
		Line types	Straight lines
		Line number	$1 \leq \# \text{ of lines} \leq 3$
		Line width	Width ≤ 2 pixel wide
		Line positions	Random
		Collapsing	$0 \geq \text{Collapsing} \geq -5\text{px}$
	Anti-recognition techniques	Rotated character counts	$1 \leq \# \text{ of rotated Ch} \leq 6$ characters
		Rotated character degree	$30 \leq \text{rotation} \leq 90$ degrees
		Vertical shifting size	$1 \leq \text{shifting} \leq 50$ pixels
		Character size variation	Size difference ≤ 12 pixels
		Character distortion	2 random points

4.1. Core Features

Based on previous evaluation studies [5, 9, 13], the core features of a captcha can be divided into two categories: content and presentation. The content includes character set and string length. The recommended character sets are pseudo-word and digits, as evaluated in [13], as they are more usable without confusing characters such as *i*, and *j*. A combination of upper and lower case letters is less confusing and has been observed in [11] to be the main character set because they strengthen the captchas against attacks [5].

It is suggested that the string length is randomized from four to less than nine characters. That is, because of the possibility that three characters or less is easier to guess correctly at random 0.1% of the time, e.g. by force attacks, with a high success rate for security applications [6]. The length of captchas should be randomized to prevent attacks from knowing how many segments are

present [6, 9]. As confirmed in [13], there is a 1.5% reduction in accuracy for each additional character. Accordingly, we recommend that the length of captcha ranges from four to eight characters.

In the presentation category, the fonts are broadly divided into two types: serif and sans-serif. A number of serif fonts have unique features for certain characters that can cause performance issues [13]. In addition, serif fonts can be more easily recognized by computers [17]. Based on these, for usability and security, the recommended fonts are sans-serif. Attackers can be prevented from guessing how many characters there are in captchas by looking at the font width by using randomly varied character widths for font size [6]. It is, therefore, recommended to utilize several random types of sans-serif fonts. For foreground and background colors, the accuracy is minimal as these colors vary [12, 13]. However, yellow-on-white (or white-on-yellow) combinations should be avoided, as confirmed empirically in [13].

4.2. Distortion

The distortion is the alteration of the captcha shape in order to make it robust and resistant to attacks. The distortion has generally two essential techniques: anti-segmentation and anti-recognition. The anti-segmentation technique is used to prevent machines from segmenting captchas, while the anti-recognition technique is used to prevent machines from distinguishing one character from another. It is important to note that although Optical Character Recognition (OCR) engines might not play a part in the text-based captcha security as machine learning techniques can recognize distorted characters better than humans [6], using anti-recognition techniques can strengthen overall captcha security. Therefore, applying anti-recognition techniques, such as the scaling and rotating of some characters, will reduce the recognition efficiency of a classifier and increase the anti-segmentation security.

Researchers have explored a set of anti-segmentation techniques including character overlapping, where the amount of overlapping is suggested to be a non-negative gap width (i.e. $gab \geq 0$) where characters touch all other characters. The results in [13] show that, accuracy is dramatically reduced due to the amount of negative gap. Hence, the random dot-size technique is recommended in a range between one and 10 pixels. For the random-dot counts, it seems that captcha is usable up to 2000 dots, but with more dots, the readability of captcha is reduced. The use of lines is one anti-segmentation technique. As derived from [13], between one and three straight lines are recommended, with a width of between one and two pixels, in a random position. The collapsing technique is specifically designed to make captcha more difficult for machines to automatically determine where characters begin and end. However, the usability of captcha can be negatively affected [11], therefore, the recommended collapsing level is between 0 and -5 pixels.

Character rotation is one recommended technique used to avoid the possibility of recognizing either a segmented or non-segmented character [9]. Based on the empirical results in [13], the suggested number of rotated characters is between one and six for each captcha, and the rotation is between 30 and 90 degrees. The vertical shifting size is also recommended and the suggested range is between one and 50 pixels. The size of each character should vary to prevent the attacker from using size to discriminate. Thus, the difference in each character size variation is suggested to be less than 13 pixels and to ensure the deformation is not predictable, the character distortion should be from two random points on the image, as discussed in [6] and [9].

In this benchmark, we discussed the recommended levels for each of the features and techniques that can be added to a generated captcha of a usable-secure design. In the following section, we construct a set of captcha schemes derived from the proposed benchmark.

5. CONSTRUCTING DIFFERENT CAPTCHA SCHEMES

The proposed benchmark helps us to understand how to customize both the core features and the distortion techniques. Based on this, we were able to design and build a captcha generator that can produce various types of captchas. Four different schemes have been constructed: Easy and Secure, Fast, Hard, and Annoying schemes. *It is important to note that all features and techniques discussed in the previous section are not necessarily to be applied in one scheme.* The following discusses the selected features of constructed schemes.

5.1. Easy and Secure Scheme

This scheme has large random features that do not impair accuracy, consisting of five to seven uppercase characters to produce a pseudo-word in a black 40 point random sans-serif font on a random colored background (except yellow), with no pixels between characters. A sample of this scheme is shown in Figure 1.



Figure 1. A sample of the Easy and Secure Scheme.

5.2. Fast Scheme

The features of this scheme are the shortest answering time, with just four to six random uppercase characters or digits in black 40 point Arial and Verdana fonts on a white background, with five pixels between characters, and two of the four characters rotated 10 degrees. A sample of this scheme is shown in Figure 2.



Figure 2. A sample of the Fast Scheme.

5.3. Hard Scheme

The features of this scheme are low user accuracy, from seven to nine lower or uppercase letters and digits in a black 35point random sans-serif font on a random color background (including yellow), with no pixels between characters, and with characters shifted vertically by 40 pixels. A sample of this scheme is shown in Figure 3.



Figure 3. Samples of the Hard Scheme: (a) represents a lowercase sample and (b) represents an uppercase sample.

5.4. Annoying Scheme

The features of this scheme are ranked in the literature (e.g. [13]) as low on user preference and typically hard, and consists of between seven to nine lowercase letters and digits in a random color 32 point random sans-serif font on a random color background (including yellow), with no pixels between characters, and characters shifted vertically by 40 pixels. A sample of this scheme is shown in Figure 4.



Figure 4. A sample of the Annoying Scheme.

6. EVALUATION OF THE SCHEMES

We have designed the proposed schemes discussed in the previous section from the presented benchmark. In this section, we conducted two experiments to evaluate the security and usability of the proposed schemes.

6.1. Security Testing

Since the security of a captcha is typically determined by the strength of both its segmentation-resistance and recognition-resistance mechanisms [6, 9, 18, 19, 20], the proposed schemes have been evaluated based on these mechanisms.

6.1.1. Segmentation Attack:









The segmentation attack of a target challenge is mostly applied in a vertical direction. Thus, vertical segmentation is applied broadly in the research community, for example in [6, 7, 9, 18, 19], to segment a challenge vertically into several chunks that may contain one or more characters. The process of vertical segmentation is explained in detail in [18], and starts by a vertical slicing process that reverses pixels from top to bottom and from left to right. This process stops once a pixel with a non-background color is detected. The X co-ordinate of this pixel defines the first vertical segmentation line. Typically, this vertical segmentation method not only achieves partial segmentation, but also contributes to our security testing process.

Using our developed generator, we generated 1000 random samples from each scheme, and performed the following attacks on them.

“Color Filling Segmentation” (CFS) is an algorithm that uses a distinct color to flood each component, which could be used to segment any color [6]. In order to apply this algorithm, we converted all samples of each scheme into a black-and-white image. This binarising process is done using the standard thresholding method: all the pixels with a color value above a heuristically predetermined threshold are converted to white, and those below it to black.

This algorithm is applied sequentially to all samples of each scheme. Examples of applying this algorithm on each scheme’s sample are shown in Table 2.

Table 2. Examples of Applying the CFS Algorithm

CAPTCHA Scheme	Results of Applying the CFS algorithm	
	No Segmentation	Partially Segmented
Scheme 1		
Scheme 2		
Scheme 3		
Scheme 4		

By applying this CFS algorithm in all schemes, no sample was completely segmented. However, 17% of samples were partially segmented in scheme one, and 24% in scheme two, and 8% and 3% were partially segmented in schemes three and four, respectively.

Based on these results, we developed a sophisticated segmentation algorithm that segments challenges vertically by using a Python programming language. This algorithm follows the recent attacking stages proposed in [19]. We fed all schemes' samples into this segmentation algorithm, sequentially. Table 3 summarizes the results of segmenting all schemes.

Table 3. Results of applying the segmentation algorithm

CAPTCHA scheme	Number of Segmented Characters		
	Zero	Partially	All
Scheme 1	78%	22%	0%
Scheme 2	60%	38%	2%
Scheme 3	83%	17%	0%
Scheme 4	94%	6%	0%

As shown in Table 3, in schemes one and two, 78% and 60% of challenges respectively, have no segmented characters, while 22% and 38% were partially segmented. Examples of these challenges are shown in Figures 5 and 6. Note that in the following examples, the vertical line demonstrates the cuts in the vertical slicing process.



Figure 5. Examples of Segmenting Scheme 1: (a) represents non segmented sample and (b) represents partially segmented sample.



Figure 6. Examples of Segmenting Scheme 2: (a) represents non segmented sample and (b) represents partially segmented sample.

In scheme three, 83% of challenges have no segmented characters, whereas only 17% was partially segmented. Figure 7 shows examples of the segmenting samples.



Figure 7. Examples of Segmenting Scheme 3: (a) represents non segmented sample and (b) represents partially segmented sample.

In scheme four, likewise, only 6% of challenges were partially segmented, while 94% of challenges have no segmented characters. Figure 8 shows examples of the segmenting samples.



Figure 8. Examples of Segmenting Scheme 4: (a) represents non segmented sample and (b) represents partially segmented sample.

It can be observed that only 2% of scheme two's challenges have completely segmented, while none of the other schemes (i.e. schemes one, three and four) have completely segmented.

6.1.2. Recognition Attack:

In this phase, all segmented characters of a scheme, whether partial or all, are tested to benchmark how resistant the segmented characters are to some OCR engines. In particular, we tested all segmented characters with Asprise V.15, ABBYY Finereader V.12 and Tesseract V.3.03. The main reason for choosing these OCR engines is that the first two are considered the best commercial OCR products in the market, while Tesseract is considered one of the most accurate open source OCR engines currently available. We fed each scheme's characters into OCR engines sequentially for an automated recognition. Table 4 summarizes the results of these attacks.

Table 4. Test Results of OCRs Automatic recognition attacks

CAPTCHA scheme	OCR Engine	Number of Recognized Characters									
		Zero	Partially (no. of characters)								All
			1	2	3	4	5	6	7	8	
Scheme 1	<i>Asprise</i>	77	3	2	8	6	1	3	0	-*	0
	<i>ABBYY</i>	69	6	11	3	5	2	4	0	-	0
	<i>Tesseract</i>	71	4	13	9	2	1	0	0	-	0
Scheme 2	<i>Asprise</i>	73	7	8	6	6	0	0	0	-	0
	<i>ABBYY</i>	60	9	11	13	5	1	1	-	-	1
	<i>Tesseract</i>	66	11	8	2	1	0	0	-	-	1
Scheme 3	<i>Asprise</i>	82	13	2	1	0	0	0	0	0	0
	<i>ABBYY</i>	78	11	6	3	2	0	0	0	0	0
	<i>Tesseract</i>	80	8	10	3	0	0	0	0	0	0
Scheme 4	<i>Asprise</i>	93	5	2	0	0	0	0	0	0	0
	<i>ABBYY</i>	94	4	1	1	0	0	0	0	0	0
	<i>Tesseract</i>	92	7	1	0	0	0	0	0	0	0

*- means no. of characters is not applicable for this scheme.

Scheme One. No sample was completely recognized by all the OCR engines. In particular, 77, 69 and 71 challenges were not recognized correctly by *Asprise*, *ABBYY* and *Tesseract* respectively, whereas the remaining 23, 31 and 29 challenges were partially recognized between one and six characters.

Scheme Two. Two samples were completely recognized by *ABBYY* and *Tesseract*, whereas 40 and 34 challenges partially recognized between one and six characters respectively. No characters were successfully recognized in 60 and 66 challenges respectively. By using *Asprise*, only 23 challenges were partially recognized, while 73 challenges were not completely recognized, and no sample was completely recognized by this engine.

Scheme Three. None of the utilized OCR engines completely recognized all the samples. Specifically, 82, 78, and 80 challenges were not recognized by *Asprise*, *ABBYY* and *Tesseract* respectively and only a few challenges were partially recognized between one and four characters respectively.

Scheme Four. No sample was completely recognized by all the OCR engines, whereas seven, six and eight characters partially recognized between one and three characters by *Asprise*, *ABBYY* and *Tesseract* respectively.

It appears that the anti-recognition techniques applied in all schemes provide reasonable resistance to OCR software attacks.

6.2. Usability Testing

We conducted a controlled laboratory experiment to evaluate the usability of our proposed schemes. In this section, the experiment setup and procedure are explained.

6.2.1. Experiment Setup:

The experiment involves using a number of subjects to solve a set of captchas presented sequentially to each participant. In the following, the experiment design, participants, apparatus and environment are described.

Experiment Design: This usability testing used a between-subject, which means that each group of participants were asked to solve 37 captchas in one of the proposed schemes (i.e. four groups: group one is asked to solve the Easy and Secure scheme, group two is asked to solve the Fast

scheme, group three is asked to solve the Hard scheme and group four is asked to solve the Annoying scheme). This ensures that the same number of captchas is typed in each group, and that there are no confounding factors causing bias in the results.

Participants: 120 participants were randomly assigned to each of the four groups, and each of these groups had an equal number of participants (i.e. 30 participants).

Apparatus: A graphical user interface (GUI) was implemented by using a Java applet, as shown in Figure 9, together with all four captcha schemes. The system in charge of storing and presenting captcha stimuli and gathering various data was developed using Java programming language and MySQL.

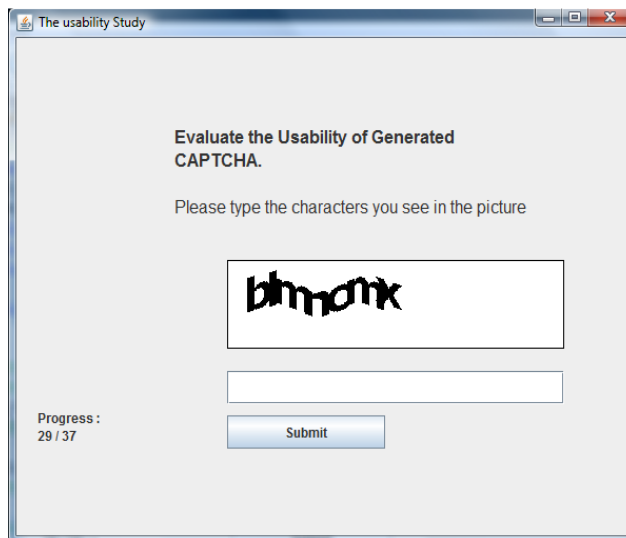


Figure 9. The developed GUI for conducting the experiment.

Environment: The experiment took place in a controlled laboratory environment to avoid any distribution issues. The participants performed the experiment task in a seated position; the tilt angle of the monitor and the chair height was adjusted to their preference.

6.2.2. Experiment Procedure:

In this section, we describe the way we conducted the experiment, i.e. instructions, procedures and data collected.

Instructions: Subjects were instructed to type characters once a captcha appeared on the screen, then to click a submit button to submit the answer. Subjects were able to monitor their progress by looking at a counter on the left of the screen which showed how many captchas had been successfully sent so far and how many remained.

Procedures: The experiment was conducted by one person in the environment mentioned previously. A consent form with a brief description of the study was given to each participant before the experiment began. Once the form was signed, the participants familiarized themselves with the procedure and the tasks involved. In each group, each participant had to complete 37 captcha samples.

Collected Data: The system records all typed characters and the time taken by each participant to type a captcha.

6.2.3. Results and Analysis:

In this study, all participants successfully completed their given task. The following outcomes were measured:

- Response time: The time (in seconds) that elapsed between the time when a captcha was shown in the screen and the time when the ‘Submit’ button was clicked (i.e. a participant spends on both recognizing and typing that captcha).
- Accuracy: The degree of conformity and correctness of typing a shown captcha in the GUI.

Table 5 shows the average response time and the accuracy with which users solved each type of proposed scheme. The Fast scheme was completed with at least 95% accuracy, while the Hard scheme proved more difficult with an accuracy level of 3%, which we had predicted since it combined some of the most difficult features. Encouragingly, the Easy and Secure schemes were completed with an average of 95% accuracy in 6.3 seconds. These results demonstrated the usability of the proposed schemes.

Table 5. Performance of Our Proposed Schemes

Scheme	Avg. Response Time	Accuracy
<i>Easy and Secure</i>	6.3	0.95
<i>Fast</i>	5.0	0.95
<i>Hard</i>	33.0	0.03
<i>Annoying</i>	14.8	0.00

7. DISCUSSION

The evaluation results of the derived schemes in terms of the characteristics of security and usability validate the proposed benchmark for designing usable-secure captchas. In particular, based on the recommended level of features in the design, there is a trade-off between security and usability. For example, some samples in scheme two were correctly segmented and recognized, and the accuracy of solving them was 95%. In contrast, no sample was completely segmented or even recognized in scheme four.

Scheme one seems to offer a balance between security and usability by demonstrating the ability to defeat any attacks as no sample was completely recognized or segmented, and the solving accuracy rate was 95%, demonstrating the usability of this scheme.

It is worth mentioning that although a set of features are applied in the proposed schemes in Section 4, other transformations, such as Lines and Arcs, can be applied according to specific levels as shown in Table 1. It would be useful in a future investigation, to examine how each particular technique affects both usability and security.

Regarding security evaluation, we notice a new attack has evolved in [20] which could be applied to test to what extent the proposed schemes can resist such new attacks. Although the results achieved might not be encouraging, testing the proposed schemes should be considered. Moreover, such heuristics can be obtained by the proposed benchmark for designing usable-secure captchas.

It is important to note that this paper analyzes the results solely based on descriptive statistics without applying any inferential statistics that could potentially reveal significant differences between the various schemes. While the current statistical analysis provides some interesting insights and the effectiveness of the benchmark, nonetheless, we are going to add inferential statistics as one of our future works.

8. CONCLUSION

This paper proposes a benchmark for designing usable-secure text-based captchas based on a community driven evaluation of security and usability. Based on this benchmark, four new captcha schemes were proposed, implemented and evaluated. The result of this evaluation showed that although the Hard and Annoying schemes demonstrated resistance to attacks, they appeared hard to read compared to other schemes. This result indicates that the proposed benchmark provides a set of efficient features and empirical heuristics for designing usable-secure captchas, such as the proposed Easy and Secure scheme. The proposed benchmark is applicable when a new distortion, a core feature or both are developed. Our ongoing work is in checking how each distortion and mechanism mentioned in the benchmark affects usability and security.

ACKNOWLEDGEMENTS

We would like to thank all the participants in our experiments. Comments from anonymous reviewers helped to improve this paper.

REFERENCES

- [1] Von Ahn, L., Blum, M. and Langford, J., 2004. Telling humans and computers apart automatically. *Communications of the ACM*, 47(2), pp.56-60.
- [2] ur Rizwan, R., 2012. Survey on captcha systems. *Journal of Global Research in Computer Science*, 3(6), pp.54-58.
- [3] Roshanbin, N. and Miller, J., 2013. A survey and analysis of current CAPTCHA approaches. *Journal of Web Engineering*, 12(1-2), pp.1-40.
- [4] Chellapilla, K., Larson, K., Simard, P. and Czerwinski, M., 2005, April. Designing human friendly human interaction proofs (HIPs). In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 711-720). ACM.
- [5] Yan, J. and El Ahmad, A.S., 2008, July. Usability of CAPTCHAs or usability issues in CAPTCHA design. In *Proceedings of the 4th symposium on Usable privacy and security* (pp. 44-52). ACM.
- [6] Yan, J. and El Ahmad, A.S., 2008, October. A Low-cost Attack on a Microsoft CAPTCHA. In *Proceedings of the 15th ACM conference on Computer and communications security* (pp. 543-554). ACM.
- [7] Huang, S.Y., Lee, Y.K., Bell, G. and Ou, Z.H., 2010. An efficient segmentation algorithm for CAPTCHAs with line cluttering and character warping. *Multimedia Tools and Applications*, 48(2), pp.267-289.
- [8] El Ahmad, A.S., Yan, J. and Marshall, L., 2010, April. The robustness of a new CAPTCHA. In *Proceedings of the Third European Workshop on System Security* (pp. 36-41). ACM.
- [9] Bursztein, E., Martin, M. and Mitchell, J., 2011, October. Text-based CAPTCHA strengths and weaknesses. In *Proceedings of the 18th ACM conference on Computer and communications security* (pp. 125-138). ACM.
- [10] Mori, G. and Malik, J., 2003, June. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on* (Vol. 1, pp. I-134). IEEE.
- [11] Alsuhbany, S.A., 2011, August. Optimising Captcha Generation. In *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on* (pp. 740-745). IEEE.

- [12] Bursztein, E., Bethard, S., Fabry, C., Mitchell, J.C. and Jurafsky, D., 2010, May. How Good Are Humans at Solving CAPTCHAs? A Large Scale Evaluation. In IEEE Symposium on Security and Privacy (pp. 399-413).
- [13] Bursztein, E., Moscicki, A., Fabry, C., Bethard, S., Mitchell, J.C. and Jurafsky, D., 2014, April. Easy does it: more usable CAPTCHAs. In Proceedings of the 32nd annual ACM conference on Human factors in computing systems (pp. 2637-2646). ACM.
- [14] Kluever, K.A. and Zanibbi, R., 2009, July. Balancing usability and security in a video CAPTCHA. In Proceedings of the 5th Symposium on Usable Privacy and Security (p. 14). ACM.
- [15] Xu, Y., Reynaga, G., Chiasson, S., Frahm, J.M., Monrose, F. and Van Oorschot, P., 2012. Security and usability challenges of moving-object CAPTCHAs: decoding codewords in motion. In Presented as part of the 21st USENIX Security Symposium (USENIX Security 12) (pp. 49-64).
- [16] Nielsen, J., 2003. Usability 101: Introduction to usability.
- [17] Wilkins, J., 2009. Strong captcha guidelines v1. 2. Retrieved Nov, 10(2010), p.8.
- [18] Yan, J. and El Ahmad, A.S., 2007, December. Breaking visual captchas with naive pattern recognition algorithms. In Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual (pp. 279-291). IEEE.
- [19] Bursztein, E., Aigrain, J., Moscicki, A. and Mitchell, J.C., 2014. The end is nigh: generic solving of text-based CAPTCHAs. In 8th USENIX Workshop on Offensive Technologies (WOOT 14).
- [20] Gao, H., Yan, J., Cao, F., Zhang, Z., Lei, L., Tang, M., Zhang, P., Zhou, X., Wang, X. and Li, J., 2016. A Simple Generic Attack on Text Captchas. In Proc. Network and Distributed System Security Symposium (NDSS). San Diego, USA.

AUTHOR

Suliman Alsubibany, PhD, is an assistant professor in the Computer Science department and the head of the department at Qassim University, Saudi Arabia. He received his PhD in information security from Newcastle University, UK, and MSc in computer security and resilience from Newcastle University, UK.

