Article can be accessed online at http://www.publishingindia.com

Scratch CAPTCHA – A Novel Touch Scratch Approach to Resolve CAPTCHA Issues

Asif Hasnain*, Waqas Malik, Mohsin Yaseen***, Syed Jawad Hussain Shahzad******

Abstract

CAPTCHA [1; 2] (Reverse Turing Test) is an authorization test to ensure that response is coming from human. Over the years, its design has been vulnerable to different attacks that utilize the visual appearance of challenge. Therefore, designing a highly robust and usable CAPTCHA that can resist such attacks is highly desirable. The study proposes a novel scheme, named Scratch CAPTCHA. It is based on a new paradigm which emphasizes the notion that strength of the CAPTCHA must be formulated by the secrecy of challenge. We have tested the robustness of our idea against a simple bot attack that successfully emulates human behavior (or actions) but not their intelligence.

Keywords: CAPTCHA, Bot Attack.

1. Introduction

Historically paper has been major source of information gathering. But ever since the advent of computing storage systems; most of the information is moved on these devices. The prime reason of this change is the benefits of these devices i.e. durability and scalability. With that came the need of its security and authorization. The purpose of online resources is to serve humans in different ways. But for some, may be for adventure or financial benefit, those resources are mishandled. Bot is the term used for any program that mimics to be human, to take major share of those resources. This resulted in the idea of having human validation test (reverse Turing Test) named Completely Automated Turing Test to Tell Computers and Humans Apart (CAPTCHA), purposed by Luis von Ahn & Co. [1; 2]. It can be passed by most humans but not bots. In CAPTCHA, machine authorizes test to ensure that the response is from a human. Scope of CAPTCHA covers human test for authorization and malware detection. It is used in applications like online polls, free email service, spam's, search engine bots, dictionary attacks and solving hard AI problems [1]. Logic problems are good example of hard AI problems. Bongo [2] was based on this problem set. AI hard problems create a win-win situation if they are used in the construction of CAPTCHA; solution to the problem will either solve AI hard problems or a unique mechanism will remain intact that will separate humans from computers. A hard AI problem [1] is the one that is unsolved despite several years of work. Hardness is just a term used by the community whereas there isn't a proof that AI problem is hard. Although they still believe that any program passing the CAPTCHA test will solve AI problems. The thought is: if intelligent people aren't able to solve AI problems for years then it isn't possible to write one for the challenge.

The study proposes a novel approach in protecting computer systems against the bot attacks. A design that differs from other proposed CAPTCHAs by providing security through obscurity. It is scalable because of the flexibility in underlying text layer. And implementation that is both robust and usable in real-life threats and application, respectively.

The remainder of paper is organized into chapters. Section 2 is about related work. Section 3 takes on the design

* Comsats Institute of Information Technology Islamabad, Pakistan, asifhasnain85@gmail.com
** Askari Bank Islamabad, Pakistan, malekwaqas@gmail.com
*** Comsats Institute of Information Technology Islamabad, Pakistan, myasinr@gmail.com
**** Comsats Institute of Information Technology Islamabad, Pakistan, jawad.kazmi5@gmail.com

principles for a secure CAPTCHA and section 4 devises a novel Scratch CAPTCHA solution that is robust as well as usable. We also explain its building block, strengths and weaknesses according to security test in section 5. Section 6 has in-depth details on the implementation of Scratch CAPTCHA prototype. Section 7 concludes the paper and sets potential directions for future research.

2. Related Work

CAPTCHAs can be classified on content type of the challenge like text, audio, or image. Each category has number of CAPTCHAs but text-based challenges have been utilized substantially. There is reCAPTCHA [2], Assira [3], Scene Tagging [6], and DevaCAPTCHA [8] but, on the advent of touch-screen devices, we see Clickable [4], Drag n Drop CAPTCHAs [5], and Multilayered [9] CAPTCHA with interest. Our approach differs from multilayered CAPTCHA [9] on the fact that characters are completely hidden. Their CAPTCHA propose character, background, and foreground interference layers which are suspect to visual attacks [5] [11] [23] [31].

3. Proposed CAPTCHA Design Principles for a Secure CAPTCHA

After studying the behavior of attacks, we have observed that they compromise CAPTCHA using pattern recognition algorithms. Being designers, we must ensure that challenges must be resistant against these visual attacks and OCRs (to some extent). So our focus is to either take away or defer these attacks by following design principles:

- Hiding the challenge. Most attacks are directly segmenting (or recognizing) characters based on the physical appearance of CAPTCHA challenge. Therefore hiding seems to be the right decision because now bot has to find where the challenge is before what it is.
- Delay time of the top layer. It's an interesting measurement because it directly correlates with security and usability of the challenge. We have achieved small delay for comparable security.
- When to remove underlying challenge. This was very important because it required human intelligence to stop scratching scratched regions. By limiting the exposure time of bottom layer; bot must perform precise actions on the given CAPATCHA.
- Flexibility of the underlying layer. Having security by obscurity adds scalability because our challenge is focused on additional (top) layer rather than obfuscation of bottom layer. So the underlining text layer can be changed (or improved) over time.

Based on above new design, we propose Scratch CAPTCHA challenge which (to the best of our knowledge) is resistant to known pattern recognition algorithms.

4. Scratch CAPTCHA

4.1. Introduction

Over the years, anti-segmentation was the sole core of resistance against attacks. If a CAPTCHA couldn't withstand the challenge then it was considered broken because state-of-the-art character recognition algorithms gave success rate of 95% [11]. Though obfuscation had consolidated anti-segmentation but it (obfuscation) directly influenced usability; because they were inversely proportional. So the idea is to take out any unnecessary obfuscation and replace it with a top layer (imagine scratch card) that provided security. For which texture (may be a random image) or carefully selected color are the viable options. We have proposed a novel technique, named Scratch CAPTCHA, which is resistant to the best segmentation attack [10] in the field. Crowding Characters Together (CCT) is one of the best design recommendations for robust CAPTCHAs but with this attack; the whole family based on this design is vulnerable. Most modern devices especially mobiles have got touch capability which is the motivation for our Scratch implementation. We did experiments on Android devices which are analyzed in the next sections.

4.2. Proposed Scratch CAPTCHA design

4.2.1. System Architecture

The system architecture of Scratch CAPTCHA consists of two major modules; client, and server. Clients simply request, and consume CAPTCHA while server is responsible for generation, and verification of client response.

4.2.2. Generation Algorithm

Right now server sends both top and bottom layer images and client creates Scratch CAPTCHA challenge. The

generation process starts with the selection of text-based CAPTCHA and a top layer image. It is flexible, and scalable to adapt any latest text challenge that is resistant to pattern recognition algorithms. Client is challenged to scratch CAPTCHA which is replaced with the text layer if user behaves intelligently. Intelligence comes from the fact that human must stop scratching the scratched regions otherwise CAPTCHA will disappear. Algorithm has two 2D PUZZLE matrices and an erasing BRUSH. The two matrices are called top (TPUZZLE) and bottom (BPUZZLE) puzzles. TPUZZLE is the challenge for removing top layer while BPUZZLE is for removing bottom layer. Both have the dimensions of text layer. Top layer puzzle (TPUZZLE) matrix has pseudorandom positive integer values where each value compliment single pixel of top layer; see table 4.1. Each BRUSH swipe results in convergence of TPUZZLE matrix values toward single value; default is one. Swipe involves basic division but model is flexible to adapt other mathematics operations like addition or subtraction. As soon as any of the matrix coordinate values get <= 1, algorithm replaces top layer pixel value with corresponding text layer value.

Table 4.1. TPUZZLE Matrix

2^0	2^{16}	...	M
2^0	2^{16}	...	M
.	.	...	
.			
.			
.	.	...	M
.			
.		...	M
N	N	N	MN

BPUZZLE matrix can either have a single (pseudorandom) SWIPE puzzle value or different puzzle values in the whole matrix. They are shown in table 4.2 and 4.3, respectively. Interestingly, single SWIPE matrix value will be easy (hence usable) but less secure than variable matrix because former will require same number of swipes unlike later. The BRUSH size can vary from single pixel to whole challenge region. Erasing single (or few) pixel (is un-realistic which) will consume a lot of time because human can't make such small contact on touch-screen. While BRUSH of the challenge size will limit user if BPUZZLE matrix has different puzzle values in the matrix. So area under finger is the good trade-off between both because it gives more control to the user.

Table 4.2. BPUZZLE Constant Value Matrix

< SWIPES >	< SWIPES >	...	M
< *SWIPES* >	< *SWIPES* >	...	M
.	.	...	M
.			
.			
.		...	M
.			
.			
N	N	N	MN

Table 4.3. BPUZZLE Random Value Matrix

2^0	2^{16}	...	M
2^0	2^{16}	...	M
.	.	...	M
.			
.			
.		...	M
.			
.			
N	N	N	MN

Algorithm 4.1 Scratch CAPTCHA

1: Define and initialize two 2D matrices, TPUZZLE and BPUZZLE
2: Define erasing BRUSH
3: Repeat
4: Update two PUZZLE matrices on each swipe
5: Update challenge layers on prescribed criteria.
6: Until challenge is submitted (or meets allocated time)

4.2.3. Dataset

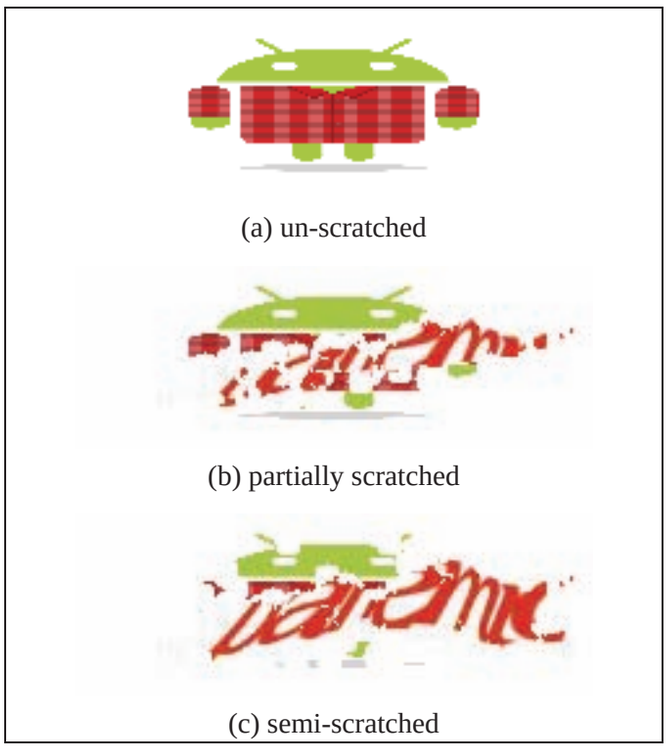
It is pointed out before that we aren't securing text challenge with new obfuscation but providing security through obscurity. There are two types of images involved in the creation of Scratch CAPTCHA; one each for top and bottom layer. Bottom layer uses latest text-based challenge like reCAPTCHA (which will be replaced with the maturity of Scratch CAPTCHA). It consumes human hours in resourceful activities like digitizing e-books. Top layer utilizes carefully collected (customized) images where customization includes re-sizing, and overlaying action messages; named "text confusion layer". It results

in more security and usability because humans are familiar with the semantics not bots.

5. Security Test and Analysis

Robustness of a CAPTCHA is one of the important factors that will decide; if our solution is deployable in real-life applications or not. It must be resilient against automatic bot attacks.

Figure 5.1. Bot Images



(a) un-scratched

(b) partially scratched

(c) semi-scratched

5.1. Bot Attack Design

In the last decade, most of the bot attacks (on text CAPTCHAs) are limited to one scheme though all of them take advantage of their visual appearance. Scratch CAPTCHA has changed this paradigm by challenging user to locate text; "where is it" before realizing "what is it". It is tested against a simple bot that can emulate human actions but not intelligence. It is divided into three general steps.

- Pre-processing
- Character Segmentation
- Character Recognition

Pre-processing involves preparing of image for later stages like segmentation and recognition. Monkey program is the prime addition of bot at this stage while common tasks involve enlarging, binaritizing image and smoothing of text. These tasks will be performed on the snaps returned from the monkey program. Second stage performs the core part i.e. segmentation and third stage is doing detection of unique character invariants. These stages can overlap depending on the scheme in-discussion. Scratch CAPTCHA successfully resist the visual challenge of CFS thrown at it.

5.2. Bot Attack on Scratch CAPTCHA

Most visual attacks make use of the design vulnerabilities in the CAPTCHA but Scratch CAPTCHA delays the attack by hiding the text. To further strengthen the security, exposure time of the text is also limited. It doesn't present itself unless TPUZZLE is solved. We have checked the robustness of Scratch CAPTCHA against the designed bot. It successfully resists this attack which is further discussed.

5.2.1. Pre-processing

Just because text is hidden in Scratch CAPTCHA; an attacker is circumspect in starting with visual attacks. The designed bot at preprocessing stage creates a monkey program. It starts scratching image in the hope of leveling or even beating human. Meanwhile, it takes snaps of the challenge so that they can be further processed in the later stages. It can do typical human actions like touch, press, swipe, and most importantly it can take snapshots.

Algorithm 5.2 Monkey Algorithm

1: Connect to the device
2: Install Package and start activity (or component)
3: Repeat until specified number of swipes
4: Swipe (left or right) on the challenge screen
5: Take snap-shot
6: End
7: Close the device

So we intentionally took three images at different times of scratching, as shown in figure 5.2. First image is completely unscratched while other two images are scratched from

twenty to eighty percent. We have preprocessed each of them one by one and made different observations. Starting with un-scratched bot image; it is threshold into binary image though eroding has zero effect because text layer isn't visible yet, as shown in figure 5.2a.

Figure 5.2. Pre-processed Bot Images

(a) Un-scratched bot image after pre-processing

(b) Partially scratched bot Image after preprocessing

(c) Semi-scratched bot image after pre-processing

Partially scratched bot image has sparsely visible characters. When we didn't use one color for text layer characters then it is quite difficult to pinpoint threshold value that can separate scratched regions of interest (or showed text layer) from top layer. Threshold on white background color results in image 5.2b. Any kind of morph operation like eroding at this image stage will be hindrance because text regions of interest are mostly small which will affect their existence. Semi scratched bot image are likely making sense for humans but not yet for bots because humans are good at Gestalt Perception [7]. Any non-intelligent eroding (mean bot must know if its required or not) will remove very small components hence strengthening our model like portion of last character "c" in figure 5.2c. Once monkey program has executed; image is purposefully enlarged to enhance the presence of text and binaritized to minimize the effect of coloring. It is morphed by shrinking or expanding text which simplifies attack.

5.2.2. Character Segmentation

Segmentation is the core part of any attacks because if it can't segment characters from the text layer, recognition get irrelevant no matter how good they are. Therefore, attack will produce zero success rate even in the presence of good recognition algorithms. Let's see how our segmentation attack digs-in.

Figure 5.3. Segmented Bot Images

(a) Un-scratched bot image after CFS

(b) Partially scratched bot image after CFS

(c) Semi-scratched bot image after CFS

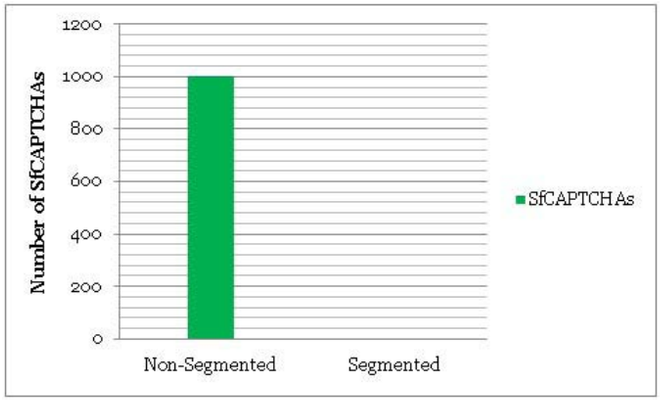
5.2.3. CFS Algorithm

CFS works by simply filling the image into unique color-filled components. Those components are then saved as collection of points for further examination like recognition. Color filling follows the steps of traversing and replacing black pixel with unique color. Once all its neighbors are traversed; algorithm locates another non-traversed black pixel, if available, and starts same procedure.

5.2.4. CFS on Scratch CAPTCHA

Binary images from the pre-processing stage are segmented through CFS. Process involves image filling connected components with unique colors which may reflect desired characters of text layer. The first binary bot image has no visible text. It only contains a black color. When image is given to the CFS; it highlights connected components with different colors, as shown in figure 5.3a. Result will be evident in the recognition stage as none of the connected component contains physical resemblance to any alphabets. Second binary bot image is partially scratched and may resemblance portions of alphabets, as shown in figure 5.3b. Though there are many connected components after CFS but still each component requires more work. Like attack may use tasks like "removing small components". It is evident that bot isn't able to recognize any character out of puzzle. Semi-scratched bot image may have some encouraging signs in human eye but bot; because not a single component has resemblance with any text character. Bot must be intelligent in selection of further operations like "removing small components" because it may damage the final outcome like in case of last character c. It is broken because of incomplete scratch so any removal will emit parts of character. Therefore the attack hasn't succeeded in achieving any segmentation i.e. zero percent, as shown in figure 4.7.

Figure 5.4. Scratch CAPTCHA-Segmentation Result



5.2.5. Character Recognition on Scratch CAPTCHA

Even if state-of-the-art recognition techniques have the solidarity to produce success rate of 95% in recognizing segmented characters [11]. Above segmentation attack produces overall zero success rate (0% * pow (95, 7)) on Scratch CAPTCHA challenge. Here seven represents number of average CAPTCHA characters in challenge. Scratch CAPTCHA is also tested against OCRs like Free OCR [74] and it perfectly resist characters recognition (precisely 0.04%), as shown in figure 5.5. This percentage is small enough that it won't accumulate to break a complete CAPTCHA. Free OCR uses Tesseract OCR engine [12] developed by HP labs but now actively maintained by Google for its services.

Figure 5.5. Scratch CAPTCHA - Recognition Result

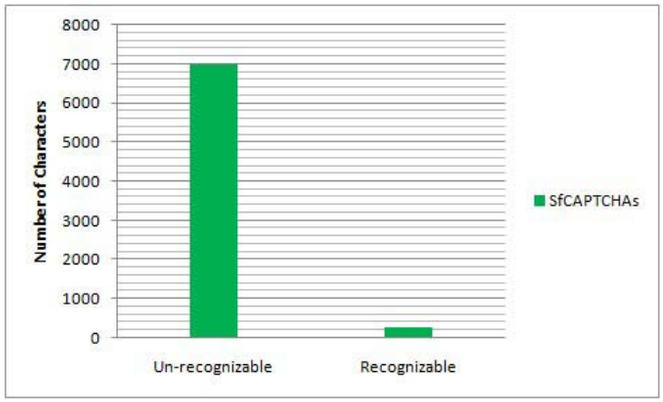


Table 5.1. Confusion Matrix

	Actual Value			Total
	p	n		
Predictive Value	p' n'	TP = 90 FN = 0	FP = 1 TN = 89	P' = 91 N' = 89
	Total	P = 90	N = 90	

5.3. Security Analyses

Scratch CAPTCHA is a binary classification test which classifies objects into two groups; here groups are bots and human. Bots are labeled positive (p) instance group while humans are negative (n) instance group. The result is formulated in a 2*2 confusion matrix. Performance of this test is measured through sensitivity (or true positive rate TPR) and false alarm rate (FAR). Sensitivity tells us the actual true positives out of positives while FAR are the fraction of false positives out the negatives. A perfect classifier has 100% TPR and zero FAR. Sensitivity is calculated through true positives (TP) and false negative (FN) instances. TP are the bots that are correctly identified

as bots and FN are the bots that are incorrectly identified as humans. Our experiment has used 90 positive and 90 negative instances. It has achieved TPR of 100% which mean all bots are rejected by Scratch CAPTCHA.

$$Sensitivity = \frac{TP}{TP+FN} = \frac{90}{90+0} = 100\%$$

FAR results in 0.01% which is the fraction of those humans that are rejected by the Scratch CAPTCHA test. It is calculated through false positive (FP) and true negative (TN) instances. FP are the humans that are incorrectly identified as bots while TN are the humans correctly identified as humans.

$$FalseAlarmRate(FAR) = \frac{FP}{FP+TN} = \frac{1}{1+89} = 0.01\%$$

5.4. Conclusion

A perfect classifier has 100% sensitivity and 0% false alarm rate while completely random guess would give around 50% accuracy. We have achieved 100% and 0.01% values, respectively, which means all bots are rejected in the test but 0.01% humans are incorrectly identified as bots. After surviving different challenges, CAPTCHA has definitely got the future in human test for proper allocation of human resources. It has resisted automatic machine attacks from day one but had a major scare, in last few years, from robust CAPTCHA in solving ecosystems [13]. These ecosystems have involved humans (from third-world countries) in solving CAPTCHAs; all in real-time. But now situation has eased-out in favor of CAPTCHAs after extreme low rates were offered to the workers (standing was $0.5/1000 in 2010). This has resulted in generation of fresh ideas.

6. Implementation

6.1. Development Environment

Scratch CAPTCHA is a generic solution to the design problems of CAPTCHAs. Server sends both the top and bottom layer images to the client. Client-specific implementation can be ported to any system from PC to touch mobile devices. It (Scratch CAPTCHA) is developed in Windows environment. It uses Eclipse tool and Java programming language. Client application uses Android SDK library for applications targeted at Android platform mobile devices.

6.2. Prototype User Interface (UI)

UI of Scratch CAPTCHA challenge is very simple, interactive and easy to understand. User is given the challenge to scratch, read and write the text in the image, as shown in figure 6.1. Application is designed to show the CAPTCHA, textbox for writing the challenge string, and a submit button. It is usable in both touch screens, and personal computers (where touch is replaced with mouse).

Figure 6.1. Scratch CAPTCHA - UI Prototype

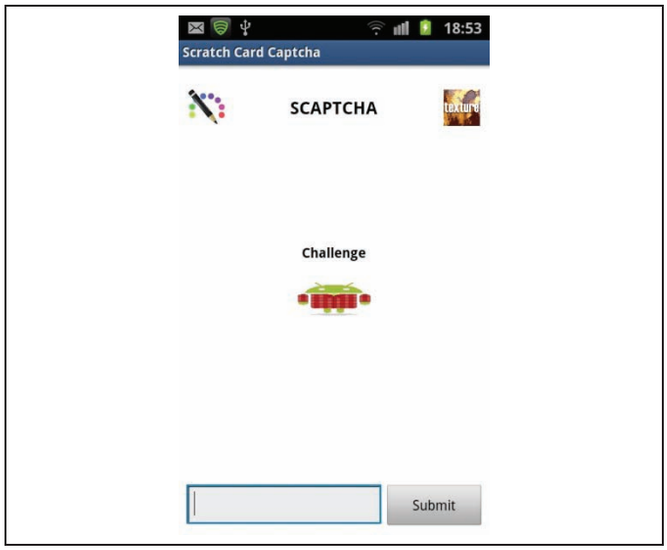
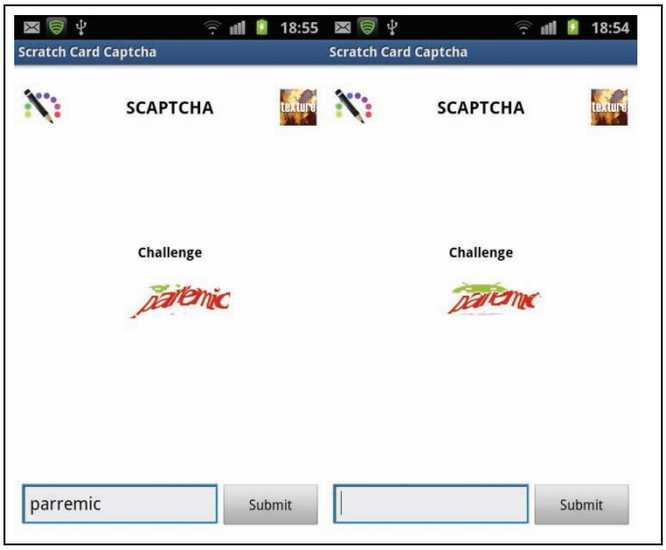


Figure 6.2a. Scratch CAPTCHA during Scratching



6.3. Scratch CAPTCHA in Touch and Desktop Environment

Users of touch devices will have the luxury of using finger for scratching while the PC users will use mouse. Both the modes of scratching are usable though touch environment has natural scratch feelings. Two sample states of Scratch CAPTCHA (while scratching) can be seen in the figure 6.2a, and 6.2b below. For example, user has typed "parremic" in the textbox before submitting it to the server.

7. Conclusions

Primarily security by obscurity is achieved so that user has to find "where", and "what" the challenge is. The old paradigm has been changed by hiding the existing CAPTCHAs schemes. Until new human test is devised, CAPTCHA will stay intact. Scratch CAPTCHA is a good candidate to replace existing techniques because it expects user to not only read and write text but scratch to qualify authorization. A good future direction from this point onward is to mould Scratch CAPTCHA for authentication as an alternative to password and pattern-based authentication. An alternative to CAPTCHA is highly desirable because of human involvement in solving CAPTCHAs [13].

References

[1] Ahn, L. V., Blum, M., Hopper, N. J. & Langford, J. (2003). C*APTCHA: Using Hard ai Problems for Security*. In Proceedings of the 22nd International Conference on Theory and Applications of Cryptographic Techniques (pp. 294-311). Berlin, Heidelberg, Germany: Springer-Verlag.

[2] Ahn, L. V., Blum, M. & Langford, J. (2004). Telling humans and computers apart automatically. *Communications of the ACM*, February, 47(2), 56-60.

[3] Elson, J., Douceur, J. R., Howell, J. & Saul, J. (2007). *Asirra: A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization*. In Proceedings of the 14th ACM Conference on Computer and Communications Security (pp. 366-374). New York, NY, USA: ACM.

[4] Chow, R., Golle, P., Jakobsson, M., Wang, L. & Wang, X. (2008). *Making CAPTCHA's Clickable*. In Proceedings of the 9th Workshop on Mobile Computing Systems and Applications. New York, NY, USA.

[5] Shah, B. M. (2008). *Drag and Drop Image CAPTCHA*. In Proceedings of 4th J&K Science Congress, November, 8(46), 330-331.

[6] Matthews, P., Mantel, A., & Zou, C. C. (2010). *Scene Tagging: Image-Based CAPTCHA using Image Composition and Object Relationships*. In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (pp. 345-350). New York, NY, USA: ACM.

[7] El-Ahmad, A. S., Yan, J. & Marshall, L. (2010). *The Robustness of a New CAPTCHA*. In Proceedings of the 3rd European Workshop on System Security (pp. 36-41). New York, NY, USA: ACM.

[8] Yalamanchili, S. & Rao, K. (2011). *Devacaptacha a Framework to Prevent BOT Attacks*. In International Conference on Computer Science, Engineering and Applications (pp. 272-279). UAE.

[9] Babu, P. K. R. & Rao, S. (2011). Implementation of secure multilayered CAPTCHA. *International Journal of Engineering Sciences and Technologies*, February, 6(2), 200-219.

[10] Ahmad, J. Y. A. S. E. & Tayara, M. (2011). *The Robustness of Google CAPTCHAS*. UK: School of Computer Science, Newcastle University.

[11] Chellapilla, K., Larson, K., Simard, P. Y. & Czerwinski, M. (2005). *Computers Beat Humans at Single Character Recognition in Reading Based Human Interaction Proofs (Hips)*. In 2nd Conference on Email and Anti-Spam.

[12] "Tesseract ocr Engine. Retrieved from http://code.google.com/p/tesseract-ocr/

[13] Motoyama, M., Levchenko, K., Kanich, C., McCoy, D., Voelker, G. M., & Savage, S. (2010). *Re: CAPTCHAS-Understanding CAPTACHA-Solving Services in an Economic Context*. In USENIX Security Symposium (pp. 435-462).