CREATING A RANDOM CHARACTERS FOR CAPTCHA SCHEMES FROM EXISTING FONTS

Ondrej Bostik
Doctoral Degree Programme (2), FEEC BUT
E-mail: bostik@feec.vutbr.cz

Supervised by: Karel Horak
E-mail: horak@feec.vutbr.cz

Abstract: For nearly a two decades, development of new mechanisms for securing web applications from unwanted traffic caused by automated programs called CAPTCHA schemes are under constant improvement. As significant fraction of this defense mechanisms are based on known issues of Optical Character Recognition methods, attackers must help to improve current algorithms to overcame the security measures.

The focus of this work is to present an idea of randomly generated pseudo characters for purpose of forming the text-based Captcha challenges. The main part of this article is dealing the problem of transforming the common computer fonts into abstract characters.

Keywords: OCR, CAPTCHA, Bubble Captcha, computer vision

1 INTRODUCTION

An anonymity of web services often leads to the situation when computer programs substitute humans in monotonous interaction with essential web services. Automated services can watch stock markets, make contracts and earn or lose money in a matter of moment without human interaction.

This situation leads to the creation of an automatic system to differentiate the human user from a machine. Resulting test was called CAPTCHA (Completely Automated Public Turing Test to tell Computers and Humans Apart) [1]. It is defined as a general task that must be very easy for the humans to solve, but it must be enormously difficult to create an autonomous machine to solve the task both for the computing resources and for the algorithm complexity.

Calling Captcha a Turing test is notable because in the original meaning of the concept arbiter must be a human, not the machine.

2 TEXT-BASED CAPTCHA SCHEMES

Common most used approach to Captcha implementation for web services is based on OCR (Optical Character Recognition) problem. Current OCR algorithms can be very robust, but they have some weakness. This imperfection limits the usage of this algorithms but can be utilized for Captcha purposes with great advantage. The server sends an image (as can be seen in fig. 1) with a sequence of characters to the client side. This image is prepared in the way that uses known OCR issues against the computers. At the same time, people who try algorithmically solve this kind of Captcha challenges helps to improve OCR algorithms ([2]).

This kind an iteration process help booth sides, but development advanced so far, that current Captcha schemes are very complex for both computers and humans. Many current Captcha challenges are so complicated, that humans cannot solve them, but machines can. Automated versatile systems for cracking Captcha can beat many schemes without any kind of human interaction. Some of these systems can be tweaked to learn new unknown Captcha challenge. As previous research shown us in [3], this kind of system can overcome almost any possible Captcha scheme with high success rate.

Previous study [3] recommend several techniques to improve the security of Captcha schemes. The first to consider is to utilize some kind of anti-segmentation technique. Many systems use lines crossing the letters, but the common mistake is to use long lines (longer than the size of a letter), that can be filtered with Hough transformation. The most straightforward technique is to use variable keyword length, that makes more difficult to guess the position of individual letters.

The next stage of security is at the level of single characters. It is good practice to apply characters of different fonts types, sizes, and rotations. On the other hand, it is not recommended to use of random noise because the current algorithms are better suited to handle the noise than human brains. It is also not recommended to use alike characters like number 0, letter O and big D, which cannot differentiate either by computer either by a human.

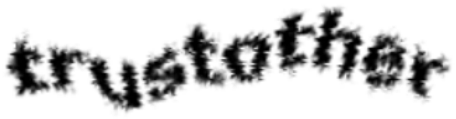


Figure 1: Wikipedia Captcha, example of ordinary Captcha scheme, (taken from [3])

A very interesting idea called reCaptcha was described in [4]. The further development and improvements were later held in Google. The rough estimations published in [4] indicated about 100 million Captcha challenges solved every day with various time from 5 to 20 seconds. This leads us to a situation when humanity wasted dozens of years every day solving Captcha schemes.

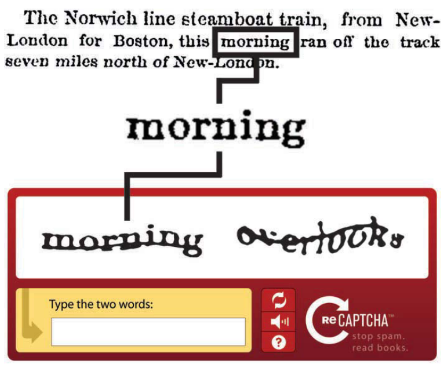
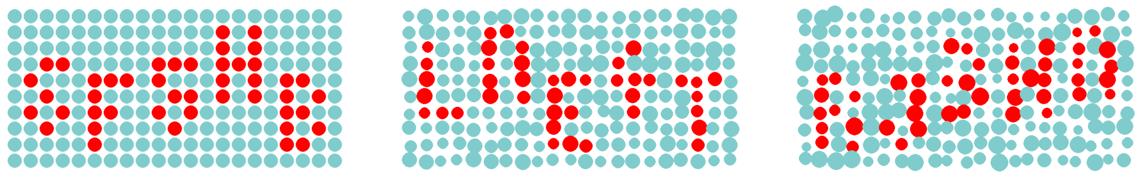

The Norwich line steamboat train, from New-London for Boston, this morning ran off the track seven miles north of New-London.
morning
morning
overlooks
Type the two words:
reCAPTCHA
stop spam. read books.

Figure 2: Sample of Google reCaptcha text scheme, (taken from [4])

Professor Ahn and his team ask a question, how to utilize this time to help humanity. The solution is simple. Archives contain a great number of documents which are not digitalized and so not available for further processing. Original system reCaptcha (see fig. 2) consist of two parts. The preparation stage utilizes two OCR algorithms, that tries to transcribe submitted document independently. Outputs are then compared. Matched parts are then marked as correctly solved. Any disagreement in outputs is used to create Captcha challenge [4].

2.1 Bubble Captcha Concept

In previous work [5], we developed elementary bi-color Captcha scheme with randomly positioned circles/bubbles forming the font. The system was designed as a web application in PHP. The main reason was to prepare a platform for rapid Bubble Captcha testing with a wide variety of people. The system can be parameterized and can be used to generate single Captcha challenge for security purposes on web pages or can be used to quickly generate the set for testing OCR algorithms.



(a) (b) (c)

Figure 3: Bubble Captcha variants overview with (a) no displacement - answer 6F5HB (b) medium displacement - answer LNE4T (c) big displacement - answer RJ3HN (from our previous work [5])

A two-dimension array representing binary grid for every character used is one of the key elements entering the algorithm. Generative algorithm randomly selects characters from used character set and position them bubble by bubble to generate our Captcha scheme alongside with bubbles of different color. Every bubble is randomly positioned approximately to its correct position in the grid and randomly scaled in size within predefined limits. The three different variants (denoted as *a*, *b* and *c* hereinafter) of Bubble Captcha are generated in this way as an input dataset of the supervised machine learning algorithms.

Implemented Captcha scheme is shown in the figure 3 displaying 3 main levels of randomness in the picture. Differences between the three variants are the rate of randomness in bubble diameters and in bubble placements. The simplest variant (a) presented on figure 3(a) contains every bubble in precise centers of the grid with uniform diameters for each bubble.

Variant (b) and (c) presented on figure 3(b) and 3(c) respectively contains various rate of randomness in positioning and bubble diameters. All the parameters of all three variants are presented in table 1.

Variant	Buble diameter	Buble variation	Grid dimension	Grid variation	Number of letters
$[-]$	$[px]$	$[\%]$	$[px]$	$[\%]$	$[-]$
a	30	0 %	35	0 %	5 - 7
b	30	10 %	35	20 %	5 - 7
c	30	15 %	35	30 %	5 - 7

Table 1: Buble Captcha parameters for each variant (from our previous work [5])

3 PROPOSED CHARACTER GENERATION ALGORITHM

One of the recommended design ideas for every Captcha scheme stated in [3] is to use various fonts. The next section expands this idea and presents an algorithm to randomize every font to make the Captcha challenge more secure.

3.1 Initial idea

In previous work [5] we used simple 5x3 pixel font to generate the Bubble captcha scheme. The font was created from scratch by us, but some of the characters used were not easily distinguishable one from another. For example, the numbers 5 and 6 differed only in one pixel and many humans cannot differentiate one from the other.

During the initial testing with human subjects, we came with the idea to use a higher resolution. One reason is making the letters be more readable for humans. The other one is to make more space to variate the letters. The variations of the font are achieved by randomizing the common text font into a new one as presented in the following section.

3.2 IMPLEMENTATION

The process of generating each new letter for the Bubble Captcha scheme starts with empty square canvas with each side measure exactly 100 *px*. Then a letter of selected font is placed to the image. The size of the letter is chosen as a half of the image side. As we tried to keep this binary, the color of the character is white. To speed up the process in next steps and lower the computational costs, the image is now cropped out of the blank space.

The next step is slightly tilting the letter. For initial testing, we choose random tilting only in the range of $\pm 18°$. In this part, we can utilize more transformation to furthermore disrupt the letter.

The most crucial part of the algorithm is to downscale the image to low resolution. The goal of this is to bring up some error which leads to the entirely new and abstract font. In our case, the resulting image has the resolution of 6x10 pixels.

The last part of the process is binary thresholding with fixed level. The resulted image is then saved to PHP file and used as an input to the Bubble captcha scheme.

The whole process of generating a letter Z is depicted in figure 4.

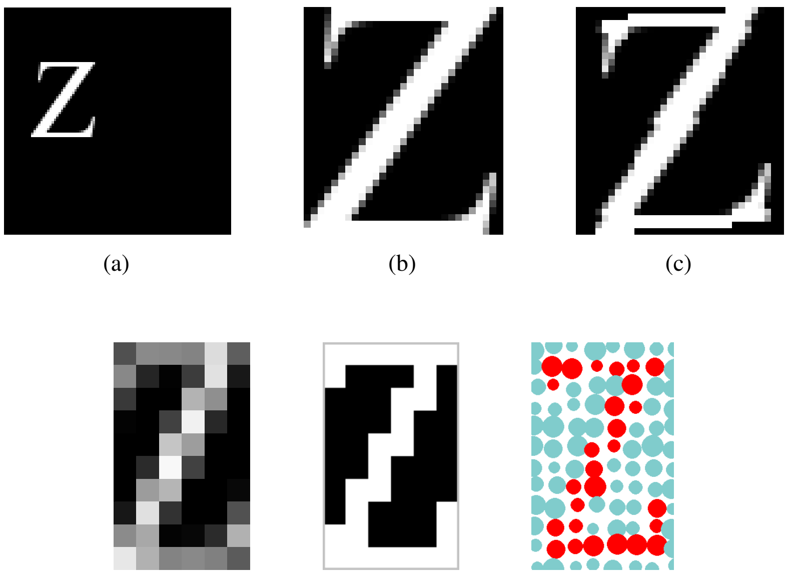


Figure 4: Process of generation new font (a) initial step with a letter of selected font draw into blank image (b) cropped image without a blank space (c) tilted image (d) rescaled image to 6x10px (e) binary image (f) resulted Bubble captcha letter

4 RESULTS OF EXPERIMENT

The generated font from previously described algorithm was used as the input for our Bubble captcha scheme. The resulting Bubble captcha challenges are shown in figure 5.

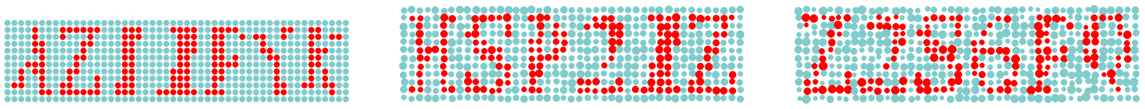

(a) (b) (c)

Figure 5: Sample of generated Bubble Captcha scheme from random fonts (a) with no displacement - answer AZILFYK (b) with medium displacement - answer HSP2IZ (c) with big displacement - answer Z2S6FQ

As can be seen, the resulting Bubble captcha challenges are much more user-friendly than the previous version. In contrast, we believe, that this kind of randomness will bring more difficulty to the Optical Character Recognition algorithms used to solve the Captcha algorithmically.

5 CONCLUSION

This paper extends our previous work on Bubble captcha scheme. We present a new way how to randomize font during Captcha challenge construction. With this concept, we are able to create unique Captcha challenges which utilize one of the greatest weaknesses of Optical Character Recognition algorithm to better protect web services from automated machines. At the same time, human users are more likely to recognize the letters as the rough outlines are preserved.

The following work will focus on the implementation of some advanced anti-segmentation techniques like using dynamically selected colors to increase the difficulty of Bubble Captcha scheme. The next idea is to generate more object and stack them layer on layer to achieve more complex structure.

ACKNOWLEDGEMENT

The completion of this paper was made possible by the grant No. FEKT-S-17-4234 - 'Industry 4.0 in automation and cybernetics' financially supported by the Internal science fund of Brno University of Technology and Competence Center realized by TACR (reg. number TE01020197).

REFERENCES

[1] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, “CAPTCHA: Using Hard AI Problems for Security,” in *Lect. Notes Comput. Sci.*, pp. 294–311, Springer, Berlin, Heidelberg, 2003.

[2] K. Kaur and S. Behal, “Designing a Secure Text-based CAPTCHA,” in *Procedia Comput. Sci.*, vol. 57, pp. 122–125, Elsevier, 2015.

[3] E. Bursztein, M. Martin, and J. C. Mitchell, “Text-based CAPTCHA strengths and weaknesses,” *Proc. 18th ACM Conf. Comput. Commun. Secur.*, vol. 2011, pp. 125–138, 2011.

[4] L. von Ahn, B. Maurer, C. Mcmillen, D. Abraham, and M. Blum, “reCAPTCHA: Human-Based Character Recognition via Web Security Measures,” *Science (80-.).*, vol. 321, pp. 1465–1468, sep 2008.

[5] O. Bostik, K. Horak, J. Klecka, and D. Davidek, “Bubble Captcha - A Start of the New Direction of Text Captcha Scheme Development,” in *Mendel 2017, 23rd Int. Conf. Soft Comput.*, vol. 23 of *23*, pp. 57–64, Brno University of Technology, 2017.