Breaking reCAPTCHAs with Unpredictable Collapse: Heuristic Character Segmentation and Recognition

Claudia Cruz-Perez, Oleg Starostenko, Fernando Uceda-Ponga,
Vicente Alarcon-Aquino, and Leobardo Reyes-Cabrera

CENTIA, Department of Computing, Electronics, and Mechatronics,
Universidad de las Américas Puebla, Cholula, 72820, México
{claudia.cruzpz,oleg.starostenko,fernando.ucedapa,
vicente.alarcon,leobardo.reyesca}@udlap.mx

Abstract. In this paper we present a novel approach for automatic segmentation and recognition of reCAPTCHA in Web sites. It is based on CAPTCHA image preprocessing with character alignment, morphological segmentation with three-color bar character encoding and heuristic recognition. The original proposal consists in exploiting three-color bar code for characters in CAPTCHA for their robust segmentation with presence of random collapse overlapping letters and distortions by particular patterns of waving rotation. Additionally, a novel implementation of SVM-based learning classifier for recognition of combinations of characters in training corpus has been proposed that permits to increment more than twice the recognition success rate without time extension of system response. The main goal of this research is to reduce vulnerability of CAPTCHA from spam and frauds as well as to provide a novel approach for recognizing either handwritten or degraded and damaged texts in ancient manuscripts. Our designed framework implementing the proposed approach has been tested in real-time applications with sites used CAPTCHAS achieving segmentation success rate about of 82% and recognition success rate about of 94%.

Keywords: reCAPTCHA breaking, segmentation attack, unpredictable collapse, three-color bar character encoding, heuristic classifier.

1 Introduction

Since its first appearance in 2000, the security mechanisms based on CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) have been subjected to multiple attacks that seek to compromise their efficiency [1-5]. Constant evolution on CAPTCHA implementation today involves a variety of proposals for text-, and image-based Turing tests [6-8]. As a result, the numerous design guidelines have been suggested for generation of robust CAPTCHAs that are able to resist attacks by malicious programs [3, 9-11].

It is a fact, that there exist commitments between the efficiency of automatic recognition facilities and the ease with which users can pass the test [4]. For this reason, two basic concepts are assumed to address automatic recognition of CAPTCHA: to break anti-segmentation techniques used to protect regions

J.A. Carrasco-Ochoa et al. (Eds.): MCPR 2012, LNCS 7329, pp. 155–165, 2012.

corresponding to characters and to overcome anti-recognition techniques for each character. While anti-recognition mechanisms alter individual letter features such as font size and type, distortion, blurring and independent rotation of each character, the main secure mechanism to avoid breaking CAPTCHAs relies on anti-segmentation techniques that guarantee their robustness. Some of the principal anti-segmentation techniques used in the new versions of CAPTCHA and reCAPTCHA are: the collapse between letters in a word, the addition of lines of different size and cluttered backgrounds [7, 11]. According to Bursztein the unpredictable collapse in reCAPTCHA is the best option to avoid segmentation of characters now widely used by various sites like Google, Facebook, Twitter, among others [10, 11].

Developed techniques for breaking CAPTCHAs also are used in pattern recognition applications particularly, for handwritten text interpretation or for optical character recognition OCR during automatic degraded text scanning. For example, the proposed approach provides simple and fast character recognition mechanism for scanning books in large scale such as Google Books and News Archive Search and their conversion to plane text [12]. Thus, the main purpose of this paper is to reduce vulnerability of CAPTCHAs from spam and frauds as well as to introduce a novel approach for recognizing either handwritten or damaged texts in ancient books, manuscripts and newspapers.

The rest of paper is organized as follows. Section 2 presents an analysis of implemented security mechanisms in modern CAPTCHAs. Section 3 describes the proposed image preprocessing stage and Section 4 presents novel character segmentation and recognition approaches. Section 5 shows experiments and evaluation of performance of designed framework for reCAPTCHA breaking. Finally, Section 6 presents concluding remarks.

2 Related Works for CAPTCHA Breaking

Currently, there are numerous techniques breaking CAPTCHA security. Only in 2011 Bursztein et al. present a tool able to recognize CAPTCHAs of *13* popular Internet sites, which include Wikipedia, eBay, CNN, Megaupload, among others with accuracy rates ranging from *5% to 93%* [11]. However, this tool does not recognize CAPTCHAs provided by sites like Google or reCAPTCHA of new versions. In [4] Yan presents attack on previous 2010 version of Google CAPTCHA, where character segmentation is based on analysis of patterns grouped in following categories: 1) point-shaped patterns (letter such as, ***i*** or ***j***); 2) cycle shaped patterns (letters ***a, b, d,*** etc.); 3) cross-shaped patterns (letters ***t*** and ***f***, and 4) pattern, which juxtaposes three vertical lines to form the character (***m*** or ***w***).

Several well-known approaches have broken CAPTCHAs such as Yahoo early CAPTCHAs [13], the CAPTCHAs used by PayPal site [14], Windows Hotmail and Gmail free e-mail providers [15], LiveJournal, phpBB, e-banking CAPTCHAs used by a lot of financial institutions and other services [3, 16, 17]. The Newcastle University research team has broken segmentation of Microsoft CAPTCHA with a *90%* success rate, and claims that the complete recognition is possible with a greater than *60%* rate [2]. This approach creates histogram of black pixels found in column assuming that characters are not overlapped and subsequently, defines letter

separation point, when no pixels are found in column. When characters get merged because of added noise, they try to remove noise to provide segmentation. This segmentation approach fails when characters are connected at least by a single pixel.

A recent approach presented by Yan at al. [4] classifies characters by grouping them into four categories similar to report in [2]. They apply a Zhang´s algorithm to obtain image skeleton and to search for image patters corresponding to declared categories. The evaluation of segmentation step provides *46%* of precision. That in conjunction with reported by state-of-art recognition success rate about of *95%* and taking into account the average number of characters in reCAPTCHA approximately as *6.41* characters per word could led to overall segmentation and recognition rate about of *33%*, [11, 13]. Although these results could give an idea that the problem is already solved, unfortunately, these reports frequently present theoretical proposal and have not formal evaluation of whole CAPTCHA breaking process. The main security mechanisms implemented in reCAPTCHA are focused on exploiting different font sizes, which suffer from a particular pattern of waving rotation and random collapse overlapping characters in a word. As shown in Fig. 1 character-level blurring is also seen in certain areas. That represents a challenge for binarization and correct segmentation of characters.

Fig. 1. Examples of security mechanisms in CAPTCHAs classified by [11]

Well-known approaches to beat CAPTCHAs as usually apply the following stages: preprocessing for removal of background clutter and noise, segmentation for subdivision of image into single regions, and recognition of characters. The most difficult task is segmentation step although the development of fast and robust classifier is also a challenging task.

3 The Proposed Image Preprocessing with Text Alignment

At the first stage applying a morphological dilation filter we aim to detect not connected regions in image that may contain a CAPTCHA word. For facilitating segmentation we provide the straightness of general inclination of a word used in recent versions of reCAPTCHA. The regions are discriminated against a minimum area criterion to be candidates for a process that corrects the slope angle of word. In this process we only seek to correct a general angle of word, it does not address the alignment and rotation of each letter. The initial idea is to detect the general inclination of a word to the left or to the right that simplifies segmentation process. In the obtained skeleton of binarized image the most left and most right pixels of word are found crossing skeleton from left to right and from right to left respectively. Using this information we consider the slope of the line formed between these two pixels to emit a preliminary judgment about orientation of image (see red line in Fig. 2). Afterwards, an image is binarized by Otsu method, which is widely used in OCR conversion due to its high quality.

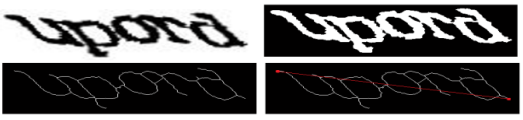

Fig. 2. Computing word orientation: input, dilated image, skeleton and word orientation line

However, some parts of letters are removed during binarization process as it is shown in skeleton of first line in Fig. 3. So, the morphological dilation filter with rectangular structural element of 3x5 pixels for connection is applied now to word (Fig. 3, first image in second line). This structural element is rotated by an angle of *45°* or *135°*, according to previously computed word orientation. We select the rotated rectangular structural element to avoid joined sections of image that would harm next angle correction of individual letter.

Fig. 3. Generation of word skeleton with correctly joined segments of characters

Finally, we compute correction angle for a word as average value of angles of lines found by Hough transform on the skeleton of image. These values are restricted only to lines with angles in the range of [*10°-80°*] and [*110°-170°*] considering preliminary assumption about word orientation. The average of angles is used for straighten a word before segmentation process as it is presented in Fig. 4.

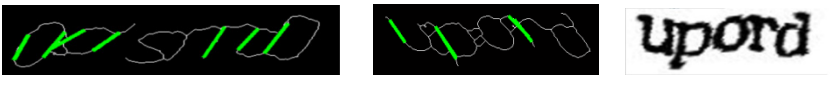

Fig. 4. Result of applying Hough transform and word alignment

4 The Proposed Algorithms for Segmentation and Recognition

4.1 Morphological Analysis of Characters

Based on analysis of character morphology in reCAPTCHA alphabet, we grouped them by certain distinctive characteristics in the following categories:

Characters with circular regions such as: ***a, b, d, e, g, o, p*** and ***q***. These letters generate regions of approximately *20* pixels wide.

Characters with occurrence of more than one pixel per column for letters like ***c, e, f, k, s, t, z***. There usually exist at least two pixels for each column.

u*-shape pattern characters* such as ***u, n, h*** are normally presented as pattern with two narrow sections with more than one pixel in column separated by a wide section of columns with only one pixel (the part of letter that connects vertical segments).

Characters of one pixel per column with slope representing letters like ***v, x, y*** with a slope about *45°±10°*.

Thin characters are letters such as ***i, j, l***; they consist of a small vertical block of approximately 5 pixels wide.

r*-shape pattern character* is formed by narrow stripe of black pixels followed by a much larger section of white pixels.

Double Characters are letters ***m*** and ***w***, which can be commonly confused with two letter ***n*** or ***v*** only when they are separated by column without black pixels.

In Fig. 5 vertical black and white stripes are shown on the right of each group, where the black ones present columns of letter with more than one black pixel, and the white show columns with only one black pixel. Fig. 5 also shows three-color code under each letter, where blue indicates that there are not pixels in column, white corresponds to columns with just one pixel and black indicates the presence of more than one pixel (i.e. blue=*0*, white=*1*, black>*1*).

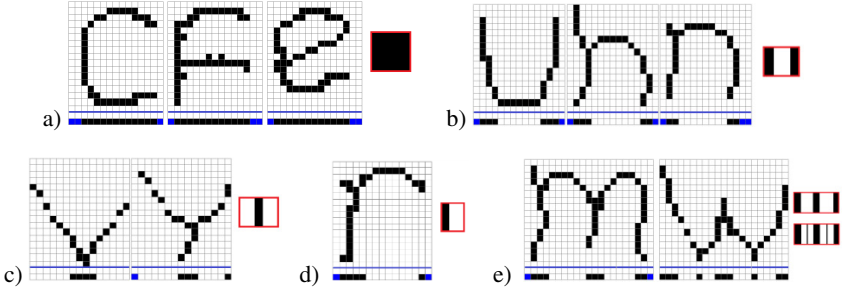

Fig. 5. The proposed classification of letters with striped patterns characterizing each group

4.2 Three-Color Bar Character Encoding

The aligned word should be segmented into letters. Now we opted for a strict threshold for eliminating as much as possible noise that may exist between junctions of letters. Subsequently, the binarized image needs two steps: the first attempts to find circular patterns in image detecting characters such as ***p, o, d, b*** among others, and another generates skeleton used later for segmentation. For this we consider closed regions smaller than half of the total area of character including space over and below it; otherwise we assume that region as background. A region corresponds to a circular shape if it satisfies: 1) if found region with the largest dimensions in width and height have more than *65%* of character area, then it is a candidate to be circular shape; 2) the ratio between width and height must be close to *1*, thus, the shapes as second and third images in Fig. 6 are discarted. The first and last images are considered as circular regions because they satisfy both criteria. The last one is wrong region, however, for distortions that occur in CAPTCHA it is acceptable error.

Fig. 6. Circular and erroneous patterns used for detection of letters with circular shapes

While algorithms generating morphological skeleton gives us a clear idea about the basic structure of a word, that skeleton still contains collapsed letters as result of applying anti-recognition mechanisms in CAPTCHA. Therefore, we propose to apply intelligent pruning branches in skeleton to leave only ones corresponding to a shape of letter. When skeletal branches are detected, then algorithm follows along branches until reaching a fork: if it has a length greater than a constant (threshold about 10 pixels), then the branch is maintained, if not the branch is eliminated as it is shown in Fig. 7 b).

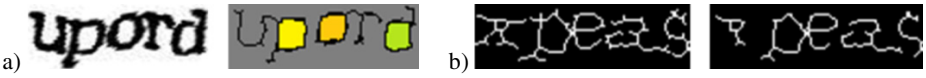

Fig. 7. Preprocessing step: a) detection of circular regions and b) pruned branches in skeleton

The results from these two processing steps are combined to form a new image shown in Fig. 8 a), in which circular regions are joined to skeleton without branches. Then a structure corresponding to the width of image is computed, where significant pixels in columns are analyzed. Fig. 8 b) shows this representation with three-color code bar: blue=*0*, white=*1*, black>*1* significant pixels in column respectively.

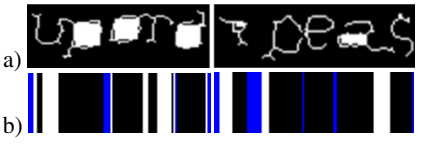

Fig. 8. Character encoding: a) pruned skeleton with circular patterns, b) corresponding three-color bar code

4.3 Heuristic Segmentation by Three-Color Bar Code

Once we have encoded histogram, it is subjected to a series of heuristics to improve segmentation. The proposed rules are applied to a color code (see Fig. 9) one at a time in following order to avoid interference between them:

Elimination of noise: change black columns to white columns where they have only one pixel wide in three-color bar code.

Slope calculation: calculate the slope of white segments. If the slope is about of *45°±10°*, those white segments meet the criteria of ***v***-type letters and now are colored in black.

*For **m**-type pattern* we propose heuristic to define cut points avoiding its wrong interpretation as two halves. This character is detected by a segment of pixels represented by two wide white bars separated by black bar (small block of columns with a higher concentration of pixels). Since ***m***-type pattern could be easily confused with two consecutive ***u***-shape patterns, the straight original image should be analyzed by SVM classifier. When ***m***-letter is found the patter is replaced by black rectangle.

The ***u****-shape pattern matching* is applied after failing to find ***m***-type pattern since it can generate two ***u***-type candidates. If classifier labels pattern as ***u***-type, the region is blacked.

Analysis of very thin regions: If black regions are less than *5* pixels wide, then we try to merge them with other adjacent thin regions. This is only done if they do not exceed a total of *20* pixels (average character wide).

Taking now optimized color bar code generated in previous step (see Fig. 9), the following steps are applied to separate characters in a word: 1) trace a vertical cut line at the middle of the white region or the blue region; 2) if there are adjacent white and blue regions, then we treat them as one region and trace cut line in the middle of it; 3) for the first character we cut from the beginning of a word; 4) the last character is cut at the end of a word.

Fig. 9. Heuristic classification of letters: a) preliminary and b) optimized three-color bar code

After vertical segmentation of characters they are cropped as it shown in Fig. 10.

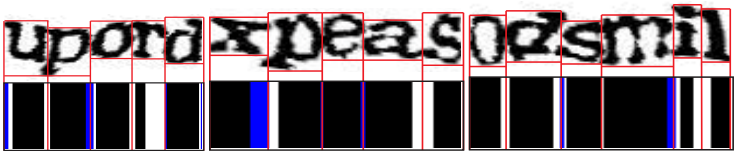

Fig. 10. Result of applying character segmentation algorithm

4.4 Character Recognition

After applying segmentation approach to CAPTCHA, each character is mapped to a fixed size (*32x32*) pattern shown in Fig. 11, and then it is represented by feature vector of *1024* values. The decimal digits, uppercase and lowercase letters have been mapped into *60* classes used then as a corpus for training of recognizer.

For character recognition the Multi-class Support Vector Machine SVM using Gaussian kernel has been implemented. The recognition begins when classifier receives input optimized by SMO (Sequential minimal optimization) algorithm feature vector. The output of classifier is a vector with number of votes for each class used then for decision making about which character has more probability to be accepted as final result. The preliminary training has been done for manually

Fig. 11. Representation of letters with binary values corresponding to symbol and background

segmented *1000* reCATPCHA images. Then we use these patterns to run an automatic segmentation on *2000* reCAPTCHA images. That has permitted self-improvement of automatic machine learning process using bootstrapping recursive technique.

During these processes the principal problem consisted in recognition of double characters like ***nn, vv, mm,*** and those which are formed by thin double characters such as ***il, li, ii, ll, it, ti, 11,*** etc. We have proposed to create additional classes for these combinations. This makes our final training set containing a total number of *231* classes. An increment of classes has improved significantly classification process without extension of total recognition time evaluated in the next section.

5 Experiments and Discussion

For evaluation of the proposed approach, our designed framework for automatic reCAPTCHA segmentation and recognition has been submitted to some tests. The first one includes cross-validations on our training set of *2000* reCAPTCHA images downloaded on local computer and run as stand-alone application. For manually segmented characters the overall recognition success rate (includes total precision of segmentation and recognition) using only *60* classes without double patterns was about *52%* and with automatic segmented characters and double pattern classes it achieved *82.5%* (significant improvement!).

The second live test has consisted in automatic recognition more than *35000* CAPTCHAs by designed framework directly in *http://recaptcha.org* site. When the framework used only *60* classes of single character corpus, the overall recognition success rate was only about *1.7%*. When the automatic segmented characters and *231* classes with double patterns have been used the overall recognition success rate is raised to *37.24%*. This low overall precision is because that recognizer has been not trained with some non-ASCII characters frequently appeared in live test (characters have not been presented in the training corpus). The average answer time using computer Core 2 Duo 2.5GHz, 4GB RAM, Windows 7 x64 is less than *2* seconds. These results are very competitive comparing them with reports resumed in Table 1.

Table 1. Performance of recently reported CAPTCHA recognition approaches

Author	CAPTCHA	Time	Segmentation precision	Recognit. precision	Overall recognition	Language, computer
Yan [4] 2010	Google CAPTCHA version 2010	7 s	46%	*95%	33%	Java, 2.4 GHz Intel Core 4 RAM: 4 GB
Bursztein [11] 2011	Google CAPTCHA, reCAPTCHA version 2011	-	-	-	0%	C# with AForge library
Our proposal	reCAPTCHA version 2011	2 s	31% single 82% double average 56.5%	*95%	average 40.4%	C# with AForge library, 2.5 GHz Intel Core 2 Duo RAM: 4 GB

* Some authors report success recognition rate *SRR(%)* about of *95%* as theoretical supposition [4, 5] of his work efficiency. They evaluated only segmentation performance. In order to calculate the overall precision *OP(%)* of segmentation and recognition the following equation is used [13]:

$$OP(\%) = SRR(\%)\left(\frac{SSR(\%)}{100}\right)^{NCW}, \quad (1)$$

where *SSR* is segmentation success rate, *NCW* average number of characters per word in reCAPTCHA equal to *6.41* [13]. For example, for Yan's [4] report with *SSR(%)=46%* and theoretical *SRR(%)=95%,* the overall precision is equal to *33%.* If we take the same theoretical *SRR(%)=95%* with *SSR(%)=56.5%* in our segmentation approach, the *OP(%)* will be equal to *40.4%.* In contrast to this, we report the practical results (*OP(%)=37.24%*) in real time application. Taking also into account the average *SSR(%)=56.5%* in our segmentation approach (due to *SSR(%)=31%* using classes with single characters and *SSR(%)=82%* using classes with double characters) the real *SRR(%)* of the implemented recognizer according eq. 1 is *93.7%.* This confirms that the implemented recognizer on SVM with our proposal exploiting classes for double characters is very promising approach for CAPTCHA breaking.

The proposed approach recognizes CAPTCHA with random unpredictable collapse affected by particular waving rotation shown in Fig.1 however, our character segmentation approach fails on CAPTCHAs presented in Fig. 12 because it cannot separate letters with rotation to different directions and with white intersections between them, or when the background is quite similar to letters, or when letters are crossed out by lines.

Fig. 12. CAPTCHAs, where the proposed approach is failed (classification by [11])

6 Conclusions

In this paper we have presented a novel approach for automatic segmentation and recognition of reCAPTCHA as well as CAPTCHAS which exploit different font sizes and suffer from a particular pattern of waving rotation and random collapse overlapping characters in a word. The proposed segmentation process based on three-color bar character encoding provides satisfactory separation of letters in CAPTCHA.

The segmentation success rate lies in the range from *31%* (for *60* classes of single character training corpus) to *82%* (for *231* classes that includes also labeling of double characters). Some tests have been done with the designed framework for real-time CAPTCHA recognition reporting practical recognition success rate about *94%*.

The obtained very satisfactory results encourage improving the proposed approach that may be considered as one of the robust techniques for CAPTCHA breaking.

Acknowledgments. This research is sponsored by European Grant #247083: Security, Services, Networking and Performance of Next Generation IP-based Multimedia Wireless Networks and by Mexican National Council of Science and Technology, CONACyT, project #154438.

References

1. Von Ahn, L., Blum, M., Langford, J.: Telling humans and computers apart automatically. J. Commun. ACM 47, 56–60 (2004)
2. Yan, J.: A low-cost attack on a Microsoft CAPTCHA. In: 15th ACM Conf. on Comp. and Com. Security, USA, pp. 543–554 (2008), http://homepages.cs.ncl.ac.uk/jeff.yan/msn_draft.pdf
3. Kluever, K.A., Zanibbi, R.: Balancing usability and security in a video CAPTCHA. In: 5th Symposium on Usable Privacy and Security, CA, USA (2009)
4. Yan, J., Salah, A., Ahmad, E.: The robustness of a new CAPTCHA. In: 3rd Workshop on System Security, NY, USA, pp. 36–41 (2010), http://doi.acm.org/10.1145/1752046.1752052
5. Chellapilla, K., Simard, P.Y.: Using Machine Learning to Break Visual Human Interaction Proofs (HIPs). In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, pp. 265–272. MIT Press, MA (2005)
6. Elson, J.: Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In: 14th ACM Conf. on Computer and Com. Security, New York, USA, pp. 366–374 (2007)
7. Zhu, B., Yan, J., et al.: Attacks and design of image recognition CAPTCHAs. In: 17th ACM Conf. on Computer and Com. Security, NY, USA, pp. 187–200 (2010)
8. Vikram, S., Fan, Y., Gu, G.: Semage: a new image-based two-factor CAPTCHA. In: 27th Comp. Security Appl. Conf., NY, USA, pp. 237–246 (2011)
9. Yan, J., Salah, A.: Usability of CAPTCHAs or usability issues in CAPTCHA design. In: 4th Symp. on Usable Privacy and Security, NY, USA, pp. 44–52 (2008)
10. Bursztein, E., Bethard, S., et al.: How good are humans at solving CAPTCHAs? A large scale evaluation. In: IEEE Symp. on Security and Privacy, Washington, USA, pp. 399–413 (2010)

11. Bursztein, E., Matthieu, M., John, M.: Text-based CAPTCHA Strengths and Weaknesses. In: 18th ACM Conf. on Computer and Com. Security, Ill, USA, pp. 125–138 (2011), http://ly.tl/p22
12. Ahn, L., et al.: reCAPTCHA: Human-Based Character Recogn. via Web Security Measures. Science J. 321(5895), 1465–1468 (2008), http://www.sciencemag.org/content/321/5895/1465
13. Mori, G.: Breaking a Vis. CAPTCHA (2012), http://www.cs.sfu.ca/~mori/research/gimpy/
14. Kluever, K., Zanibbi, R.: Breaking the PayPal CAPTCHA (2008), http://www.kloover.com/2008/05/12/breaking-the-paypalcom-captcha/
15. Dawson, K.: Windows Live Hotmail CAPTCHA Cracked, Exploited (2008), http://tech.slashdot.org/article.pl?sid=08/04/15/1941236&from=rss, and Gmail CAPTCHA Cracked, http://it.slashdot.org/article.pl?sid=08/02/27/0045242
16. Li, S., Syed, A., et al.: Breaking e-Baking CAPTCHAs. In: 26th Comp. Security Appl. Conf., NY, USA, pp. 171–180 (2010), http://www.acsac.org/2010/openconf/modules/request.php?module=oc_program&action=summary.php&id=53
17. Kruglov, S.: Defeating of weak CAPTCHAs (2012), http://www.captcha.ru/en/breakings/