IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 2, Apr-May, 2014
ISSN: 2320 – 8791 (Impact Factor : 1.479)
www.ijreat.org

Human Captcha using Video Technology

Thilagavathy.D[1], Priya .S.V[2]

[1]M.E. II Year, Department of Computer Science and Engineering, Sriram Engineering College, Perumalpattu – 602 024

[2]Associate Professor & HOD, Department of Computer Science and Engineering, Sriram Engineering College, Perumalpattu – 602 024

Abstract

The Captcha is the security code designed to perform password protection to secure the authenticated user. It uses word tests for easy human recognition. The CAPTCHAs are then random realizations of the random CAPTCHA word starts with an initial random field use Gibbs resampling to re-simulate portions of the field repeatedly using until the word becomes human-readable. In the proposed system Knw-CAPTCHA is used to differentiate humans from computers. It is readable only to humans. The Passwords are used means of authentication as passwords are very convenient for users, easier to implement and user friendly. Password based systems suffer from two types of attacks offline attacks and online attacks. Eavesdropping the communication channel and recording the conversations taking place on the communication channel is an example for offline attack. Video CAPTCHA is used to avoid online hackers.

Keywords: *Image processing, Markov random field, security, simulation, statistical information compression.*

1. Introduction

A CAPTCHA is a "Completely Automated Public Turing test to tell Computers and Humans Apart", widely used to protect online resources from abuse by automated agents. He suggests that hard artificial intelligence (AI) problems form the test basis and defines a (α, β, η)-CAPTCHA as a test that 1) Can be solved by at least α proportion of humans (e.g., the English speaking adult portion) with a probability of success greater than β; 2) if a computer program can solve it with probability greater than η in fixed time, then the program can be used to solve the hard AI problem.

A common CAPTCHA is an image of (usually alphanumeric) characters to that are easy to identify by English-reading humans yet translate hard AI problems for security. Segmentation of characters within a word image is error prone [3], and continues to be difficult for contemporary OCR algorithms [4].Therefore, segmentation should be hard error prone [3], and continues to be difficult for contemporary OCR algorithms. Therefore, segmentation should be hard to ensure an OCR-based CAPTCHA is resistant to computer programs. Herein, we introduce a general method for generating "KNW-CAPTCHAs" with the view that random than is really random field simulation meaning people rather than computers will know what they behave. A KNW-CAPTCHA is initialized as a random field, and the CAPTCHA is then generated via partial Gibbs re-sampling in order to provide enough information to make the test word human-recognizable, yet ensure that OCR remains hard.

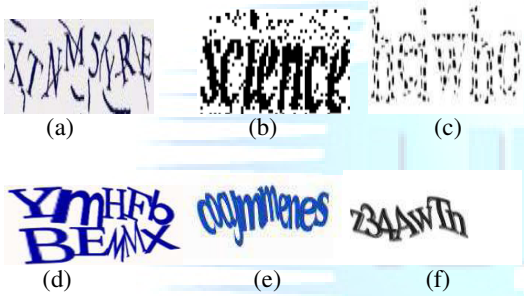
The target population for our KNW-CAPTCHAs is English-readers with better than 20/60 vision though we have little control over the participants in our readability studies). We establish high β via a readability study and endorse low η via experiments with modern OCR programs.

We begin with an overview of past and present text based CAPTCHAs. (While there are many alternatives to text based CAPTCHAs, such as the image-based IMAGINATION [5], which requires users to annotate images, text-based CAPTCHAs continue to be the de facto standard in industry.) The early, now broken PayPal and the Microsoft CAPTCHAs discussed in [4] and [6], respectively, both relied on background noise and random character strings to resist automated attacks but did not employ character crowing.The KNW-CAPTCHAH would be used in practice as the background noise provides additional security but the CAPTCHA remains highly readable to humans. It compares the human readability and attack resistance of the KNW-CAPTCHAH with several CAPTCHAs deployed by major corporations.

As the correct answers for the comparison CAPTCHAs are unknown, we use optimistic solving accuracy to determine human success. An OCR program is considered correct if it matches any of the human responses. It clearly illustrate that both the KNW-CAPTCHAE and KNWCAPTCHAH are highly readable and difficult to attack; even the KNW-CAPTCHAE appears to have resistance to OCR comparable

to or surpassing CAPTCHAs currently used by Google, YAHOO, and eBay.

There were no computer successes against the KNW-CAPTCHAH, yet it obtained over 94% human success. Mori *et al.* [7] successfully attack both EZ-Gimpy and Gimpy CAPTCHAs. KNW-CAPTCHAH would be used in practice as the background noise provides additional security but the CAPTCHA remains highly readable to human. The authors of [8] are able to remove the background clutter and segment the challenge into four character recognition problems, which are solved by determining which template character image requires the least distortion



(a) (b) (c)

(d) (e) (f)

Fig 1.Captcha examples

The authors of [8] are able to remove the background clutter and segment the challenge into four character recognition problems, which are solved by determining which template character image requires the least distortion to match the observed character image. (Performance is further improved using additional steps.)

Pessimal Print introduced in Coates *et al.* [9], simulates low-quality print images that challenge OCR. The CAPTCHA generation randomly selects a word, a font, and a set of image degradation parameters to thicken, crowd, fragment, and add noise to character images. 685 word images were generated; all were readable to the ten human volunteers, while almost all were unrecognizable to the ExpervisionTR, ABBYY Fine Reader, and IRIS Reader OCR programs. Furthermore, OCR performance was very sensitive to changes in the parameters.

The CAPTCHAs used by Google, Yahoo!, and Windows Live all share similar properties: a lack of background noise, distortion of character or word images, and extreme crowding of adjacent characters.

Segmentation resistance is largely accomplished by character crowding, notably lacking from earlier, now broken CAPTCHAs such as the CAPTCHA service.org CAPTCHAs in [12]. In the PayPal CAPTCHA in [6], the Microsoft CAPTCHA in [4], EZ-Gimpy in [7], and Gimpy-r in [8]. However, this extreme crowding also makes human recognition a challenge. For example, is it obvious what the character string in the Google CAPTCHA is? In contrast with the methods covered above, we view CAPTCHA generation as correlated random field simulation.

Like Pessimal Print [9], our images provide partial, noisy information .We also leverage Gestalt perception to maintain a human-readable image, as in [10] and [11]. However, our use of randomness is far more fundamental and thereby far harder for computers to deal with than prior methods. We observe that the human readability of random CAPTCHA images is captured by the site, i.e. pixel, marginal probabilities and the site-to-nearby-site covariances; the actual joint distribution of the sites is not so important.

Our method begins with a correlated random image that is evolved randomly a site at a time via Gibbs sampling until the random test word is human-readable. Our method of calculating each site's conditional probability mass function given the nearby sites that are either known or already simulated gives us exactly what is required for Gibbs sampling.

In this paper, we investigate two variants of the KNWCAPTCHA: the KNW-CAPTCHAE, an easy variant generated without any background noise, and the KNW-CAPTCHAH, which is generated using character fragments as the background noise.

The KNW-CAPTCHAE is used to investigate how the generation parameters (especially the number of Gibbs iterations, *NG*) affect the attack resistance of the resulting CAPTCHA. It shows both the attack resistance and human readability of the KNW-CAPTCHAE for various values of *NG*, where computer success is the proportion of CAPTCHAs where either of the OCR programs Tesseract or ABBYY FineReader successfully recognized it, and human success is the proportion of CAPTCHAs where a human successfully recognized it.

2. KNW-CAPTCHA

We now present how to estimate the required parameters for a particular KNW-CAPTCHA and use those parameters to generate a novel random CAPTCHA in Sections II-A and II-B, respectively.

2.1 Parameter Estimation

It begins by generating the data for the estimation process that consists of many independent instances of a particular word, where each instance varies randomly in many ways. The parameters learned from this data will represent the challenge Word by learning the parameters (site probabilities and site-to nearby- site covariances) from this data, we can construct the conditional probabilities of the previous section and, thereby, do the Gibbs resampling portion of our CAPTCHA creation.

The algorithm for generating the data consists of selecting a word to serve as the KNW-CAPTCHA's correct response and then generating a number of random images representing this word by varying fonts and placement of characters in the word.

The word images will be constructed by joining individual character images. Herein, we select a random word uniformly over a fixed dictionary of common English words with a length of at least three characters. For each letter in the English alphabet and for each of 18 fonts, we generate character images denoted { f1,1, . . . , f1,26, . . . , f18,1, . . . , f18,26}, i.e., fi, j is the character image of the j th letter in the i th font. To ensure that forming a word image by joining random character images results in consistent horizontal placement of individual character images, we work with character images that, for a given letter, all have the same width.

It accomplish this, we generate trimmed or scaled character images for each letter as appropriate. { f1, j , . . . , f18, j }, where a bounding box is the smallest rectangle that encloses the character. For i = 1. . . 18, j = 1. . . 26,. The letters {i, j, l, r, t} were chosen for trimming instead of scaling since scaling some of their images results in very tall bounding boxes due to their highly variable character widths. We then generate K images of the chosen character string with pixel state space {−1, 1} = {white, black}, and nc is the number of characters in the character string using the following algorithm

The horizontal distance between each adjacent character's bounding boxes is chosen using a random number selected uniformly over {1, 2, 3}. This is fixed for all K images.

The vertical displacements of characters are determined using the values {v0, v1, v2, . . .} of a reflecting random walk, moving upward or downward one with probability 12; upon hitting the boundary {−25, 25}, it reflects. The random walk is initialized randomly over {−10... 10}. The i th character image, where i ∈ {1, 2, . . . , nc}, will be placed vertically by centering it according to the vertical center of its bounding box, and then shifting it up or down according to the value v(i−1)×6 of the random walk. This produces (n = 6, p = 12)-binomial shifts before reflection. This is also fixed for all K images.

2.2 KNW-CAPTCHA Generation

It presents the KNW-CAPTCHA generation details, which consists of generating background noise and then simulating the character string, using modified Gibbs sampling with the parameters obtained in Section II-A, on top of the background noise. Introducing background noise is a common technique when generating CAPTCHAs since it introduces red herring character shapes that must be removed or ignored by a computer program. Our view is that the best red herrings are actual character pieces.

Background noise also makes segmentation more difficult since, for example, vertical projection will not detect gaps between adjacent characters bridged by appropriate background noise, and connected components will view two adjacent characters as one if they are connected by background noise. We generate background noise via the ScatterType algorithm in [11]. While the original intent of the ScatterType algorithm was to produce CAPTCHAs that were human readable but difficult to crack, our goal is the reverse, produce ScatterType CAPTCHAs that are clearly unreadable to humans yet "readable" to computers.

The character shapes produced will serve as effective red herrings. Automated attacks come from the original correlated random field and the pixel by pixel randomness in still, the character pieces are often erroneously detected by computer programs as part of the actual character string. The unreadable background noise is generated using the following algorithm.

1) Choose a five-letter character string uniformly, with replacement, over the English alphabet.
2) Apply the ScatterType algorithm using a fixed font and the following parameters KNW-CAPTCHA is generated using with CAPTCHA and without CAPTCHA.

3. SECURITY DISCUSSION

Generate an easy KNW-CAPTCHA for a character string consisting of two random (i.e., chosen uniformly over the English alphabet), lower case letters, with no background noise (i.e., the modified Gibbs sampling is initialized with a white image).

Find the global smoothed break point minimum within the boundaries (i.e., in the horizontal region between the first and last black pixels) of the generated KNWCAPTCHA using the above sophisticated vertical projection segmentation attack. In the case of a tie, select the point randomly amongst the global minima.

If the bounding boxes of the two characters overlap, and the segmentation point is within two pixels of the middle of the overlapping region, then consider the segmentation correct. If the bounding boxes do not overlap, then consider the segmentation point correct if it lies anywhere in the region between the bounding boxes.

The determination of whether the segmentation point is correct is slightly modified from the work which sought to isolate measures of segmentation performance from recognition engines. Since we are attempting to evaluate only segmentation resistance at this point, rather than recognition resistance, ours was an appropriate technique to adopt.

We estimate *ps* the probability of successful segmentation, using the maximum likelihood estimator ˆ *ps* . The results are summarized. The segmentation performance is low despite

targeting character crowding and presenting simplified two character images without scatter noise. Furthermore, the segmentation performance does not vary significantly with *NG* indicating the vertical projection algorithm has difficulty with the basic construction of the KNW-CAPTCHA. Collectively, the security mechanisms in the KNWCAPTCHAH will prove difficult to circumvent. However, should it be successfully attacked, other variants may take its place.

3.1 Variants

The mechanism described is more general than the particular example we study in this paper: it can easily be extended to counter new attacks. For example, if an attacker is able to remove the background noise, one can use the striped correlated noise if a dictionary-based attack succeeds, one can use pseudo-words; if a segmentation attack succeeds, one can increase character crowding or decrease *NG* (see In fact, one could deploy several variants simultaneously, effectively reducing the reward for successfully attacking any particular variant. We now provide a high-level view of potentially useful variants; furthermore, we will discuss how a variant can be selected by a user of the KNW-CAPTCHA.

Sample generation many random instances of a character string image are generated. Parameter estimation simulation parameters are estimated from image samples. Initial state an initial state for the KNW-CAPTCHA is generated. Simulation re-simulates random pixels of the KNWCAPTCHA until word appears.

This is a template method pattern meaning that we have given a high-level description of the algorithm while allowing variants to define the details of how each step is accomplished. Within, we studied two variants, the KNW-CAPTCHAE and the KNW-CAPTCHAH. The KNW-CAPTCHAE was deliberately designed to be vulnerable to attack by eschewing random vertical displacement in the sample generation step, and background noise. In contrast, the KNW-CAPTCHAH does use random vertical displacement and the initial state is generated using a ScatterType CAPTCHA. These two relatively simple differences produce significantly different CAPTCHAs, yet the overall algorithm remains the same. Now we introduce several variants to illustrate the flexibility of the KNW-CAPTCHA algorithm.

3.2 Clustered Correlated Noise

Instead of the ScatterType character fragments, we generate the initial state using the simulation algorithm detailed with pair-wise covariances set to the Euclidean distance from the pixel being simulated with _ = 2. The effect is an initial state with clustered random shapes.

3.3 Striped Correlated Noise:

It generate the initial state using one pass of the simulation algorithm with _ = 2. Let the pixel *p* being simulated have the coordinates *(x, y)*, and let pixel *pi* have the coordinates For each pixel *pi* in the neighbourhood of *p*, set the pair-wise covariance to 0 if $x < xi$ and $y < yi$, or if $x > xi$ and $y > yi$; otherwise set the pair-wise covariance according to the Euclidean distance between *p* and *pi* . This generates striped correlated noise.

3.4 Simulated Characters

It generate the initial state using the KNW-CAPTCHAE algorithm with a lower *NG* and a random character string. This produces a background noise that is distinct to humans but difficult to eliminate automatically due to the similarity in form to the CAPTCHA word. A random character string is used instead of a word to prevent confusion between the background noise and the CAPTCHA word.

3.5 Variant Selection

It is clear that it is easy to generate varied CAPTCHAs using the methods laid out in this paper by modifying parameters or the steps in the CAPTCHA algorithm we have not yet discussed how these variants can be compared and selected. In the following, we will present an idea of how to accomplish this automatically.

Any comparison should naturally take into account both attack resistance and human readability. However, different users of CAPTCHAs will place different on each quality and a CAPTCHA should be able to balance the two qualities. θ is the set of parameters determining the variant of KNW-CAPTCHA generated, $a(\theta)$ is the probability of an attack succeeding on an instance of the variant, $h(\theta)$ is the probability of a human being able to read an instance of the variant, and $w \in [0, 1]$ is a weight balancing the two qualities. Then $f(\cdot)$ is a cost function, and the goal of a CAPTCHA should be to minimize it.

The role of *w* is to allow the user of a CAPTCHA to balance attack resistance and readability. $a(\theta)$ and $h(\theta)$ are unknowable and can only be estimated. One method of estimating $a(\theta)$ would be to attempt to attack the many instances of the CAPTCHA with several OCR engines and consider it a success if any of them succeed (similar to how the re CAPTCHA project determines if a word image should be used as a CAPTCHA challenge if any of the OCR engines recognize the word, and $yi = 0$ otherwise. $h(\theta)$ could be estimated in a similar fashion, using human readability experiments. Then, selecting the appropriate CAPTCHA variant becomes a matter of minimizing the cost function over the evaluated variants.

4. Experiment and Result

In the following, we describe how we measure the properties of the KNW-CAPTCHA and provide results. In Section III-A, we attack a weak variant of the KNW-CAPTCHA with computer programs to establish a lower-bound to the KNWCAPTCHAs' attack resistance. we measure the human readability of the hardened KNWCAPTCHA finally; we measure the attack resistance together with the human readability of the hardened KNW-CAPTCHAs. We use KNW-CAPTCHAE to refer to the easy KNWCAPTCHA variant.

4.1 KNW-CAPTCHAE Experiments

This tactic will provide the most evidence that η is low, i.e., that the KNW-CAPTCHA is difficult to crack. Word accuracy is calculated based on the number of words recognized, and all Word comparisons are done ignoring case. For a particular word, the experiment is as follows
1) Generate a KNW-CAPTCHAE for the word word with no background noise and no vertical displacement.
2) Run the OCR program to obtain word *O*.
3) Compare word *K and* word *O*.
In order to validate the human readability of the KNW-CAPTCHAE, we also collect human results via Amazon Mechanical Turk (AMT) [16] Results are summarized in Table I, and an example of a KNW-CAPTCHAE is provided.
Based on these results, it appears that both OCR programs have great difficulty recognizing the KNW-CAPTCHAEs. In fact, the computer performance on the unhardened KNW-CAPTCHAE with *NG* = 200 is similar to the results on the Google CAPTCHA (Table I), which was the most difficult for OCR to recognize of Google, YAHOO, and eBay. In addition, human performance on the KNW-CAPTCHAE is very high. As expected, both OCR and human performance generally increase .

TABLE I
95% confidence interval of computer and human

NG	*nT*	ABBYY	Tesseract	Human
200	1000	0.012 ± 0.007	0.000 ± 0.000	0.990 ± 0.006
400	1000	0.015 ± 0.008	0.000 ± 0.000	0.99 1 ± 0.006
600	1000	0.007 ± 0.005	0.002 ± 0.003	0.991 ±0.006
800	1000	0.032 ±0.011	0.015 ± 0.008	0.996± 0.004
1000	1000	.034 ±0.011	0.020 ± 0.009	0.993 ± 0.005

4.2 KNW-CAPTCHAH Experiments

In the *(α, β, η)*-CAPTCHA context, our β is high. The KNW-CAPTCHA should be applied to literate English reading adults with normal eyesight. (In practice, alternative CAPTCHAs, such as an audio CAPTCHA, should be provided others.) Our task is to estimate β and the time to complete the challenge empirically. The following experiments use a set of 300 KNWCAPTCHAH images generated with *NG* = 800 based on the results of the previous sections along with visual inspection in order to balance attack resistance with readability.

A. Online Readability Study:

To collect these results, we set up the website http://www.knwcaptcha.org. Volunteers participating in this online study were anonymous. No incentive was provided. The procedure was as follows.
1) The visitor is presented with information on how the experiment is conducted and how the data will be used.

2) If the user does not accept, the experiment is terminated.

3) In order to familiarize the visitor with the process, he or she is presented with an example of a KNWCAPTCHAH along with the correct response. The example shows a KNW-CAPTCHAH with the encoded word outlined, and is designed to show the visitor how to recognize the encoded word in noise.

4) The visitor is shown a set of 25 KNW CAPTCHAHs.
A visitor is never shown the same word more than once. Beside each KNW CAPTCHAH, the visitor enters a response, and submits the entire data set upon completion.

A human's response to a KNW-CAPTCHAH is marked as correct if it matches the encoded word, ignoring case and incorrect otherwise. A human's response to a KNW-CAPTCHAH is marked as correct if it matches the encoded word, ignoring case and incorrect otherwise. In the analysis, we model the trials as (β)-Bernoulli random variables. The experiment yields n_T responses from humans. As before, we use the maximum likelihood estimator $\hat{\beta}$ to estimate β.

The time to solve each challenge is calculated using the time elapsed from when the user is first presented with the CAPTCHAs to the submission of the responses. Humans succeeded at solving a high proportion of KNWCAPTCHAHs.

A. Amazon Mechanical Turk:

In addition to collecting responses from volunteers at we used Amazon Mechanical Turk (AMT) [16]. AMT is an online service which allows requesters to submit tasks which will be completed by a pool of workers. The use of AMT for collecting human feedback in research has been established in several works for example, in order to be resistant to workers gaming the task, it is important that the task be as much effort to complete incorrectly as correctly.

The CAPTCHA schemes are evaluated in terms of human readability based on the amount of agreement between three workers on a CAPTCHA image containing an unknown word. Our task of evaluating the responses to a known CAPTCHA is relatively straightforward and an appropriate fit for AMT. Our

AMT task design is similar to that of knwcaptcha.org. Each task submitted to AMT consisted of a KNWCAPTCHAH image and a response field.

A batch of tasks is preceded by brief instructions and an example, as on knwcaptcha.org. No qualification pre-tests are administered, nor are workers penalized (via, for example, lack of payment) for wrong answers. AMT provides more diverse, international respondents than could be obtained by recruiting local volunteers While no demographic information is collected, a comprehensive survey of the AMT worker population conducted by Ross *et al.* [20] found a large population of international, young, educated workers. Furthermore, [19] examines the effect of demographics on CAPTCHA solving ability.

Its particular interest to us is that native English speakers are able to solve English or pseudo- English CAPTCHAs far faster, which indicates that the KNWCAPTCHAH is biased against non-native English speakers.

TABLE II
95% confidence interval of optimistic computer and human performance

nC	ABBYY	Tesseract	nH
KNW-CAPTCHA	300 0.00 ± 0.00	0.00 ± 0.00	9000.90 ±0.02
Google	300 0.00 ± 0.00	0.01 ± 0.01	900 0.81 ±0.03
YAHOO	300 0.01 ± 0.01	0.02 ± 0.01	900 0.93 ± 0.02
eBay	300 0.05 ± 0.02	0.29 ± 0.05	900 0.97 ± 0.01

4.3 SIMULATION RESULTS

In CAPTCHA, we developed and implemented a new method of generating random CAPTCHAs, called KNW-CAPTCHAs, using random field simulation that outperforms popular CAPTCHAs in use today. First, we estimated the marginal probabilities of sites and site-to-site covariances of the KNW-CAPTCHA based on randomly generated samples.

Second, we used an efficient algorithm to simulate a new KNW-CAPTCHA based on these parameters in a Gibbs-like manner. Furthermore, we established that the KNW-CAPTCHA is an effective separator of computer programs and humans. It established that the KNW-CAPTCHA is very readable to humans.

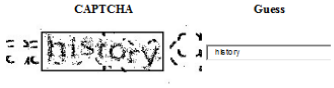


Fig 2. Knw-Captcha

Select a random alphabet letters and draw into an dummy image as a background. Then select three alphabets from randomly and paste on approximately center position of image. Split and throw the noise on that nearby characters to apply Black pixel values on the surrounding images Knw - captcha has been generated by these Security aspects.

Online OCR not able to recognize the character present in the Knw-captcha. Server has Default videos. First those videos have to be split into collection of frames. Then overwrite our Knw-captcha images in to particular position of Frames. Client can view the Knw-captcha as a video format called video CAPTCHA.

Fig 3. Video CAPTCHA

5. CONCLUSIONS & FUTURE WORK

In CAPTCHA, we developed and implemented a new method of generating random CAPTCHAs, called KNW-CAPTCHAs, using random field simulation that outperforms popular CAPTCHAs in use today. First, we estimated the marginal probabilities of sites and site-to-site covariances of the KNW-CAPTCHA based on randomly generated samples; second, we used an efficient algorithm to simulate a new KNW-CAPTCHA based on these parameters in a Gibbs-like manner. Furthermore, we established that the KNW-CAPTCHA is an effective separator of computer programs and humans.

The colors used for the background noise and the CAPTCHA could change from left to right; The intent of these changes would be to effectively add another dimension to the problem, further confusing an attacker without compromising readability. As above, the intent would be to increase the dimensionality of the problem for the attacker without decreasing readability.

In grey levels could be used to make distinguishing between the background and the letters themselves more difficult, or to make the shapes of the characters themselves less obvious. Finally, in this paper we used only black and white when generating samples. however, this method can be readily extended to generate CAPTCHAs with one or many grey levels. As above, the intent would be to increase the dimensionality of the problem for the attacker without decreasing readability.

The grey levels could be used to make distinguishing between the background and the letters themselves more difficult, or to make the shapes of the characters themselves less obvious.

Furthermore, we established that the KNW-CAPTCHA is an effective separator of computer programs and human's .The main challenge would be to adjust the random sample generation and parameter estimation methods used in this paper in such a way that maintains or improves readability.

REFERENCES

[1] M. Blum, L. Von Ahn, J. Langford, and N. Hopper. The CAPTCHA Project, Completely Automatic Public Turing Test to Tell Computers and Humans Apart. Dept. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA [Online]. Available: http://www.captcha.net

[2] L. von Ahn, M. Blum, N. Hopper, and J. Langford, "CAPTCHA: Using hard AI problems for security," in Proc. Adv. Cryptol.—EUROCRYPT Int. Conf. Theory Appl. Cryptograph. Tech., Warsaw, Poland, LNCS 2656. May 2003, pp. 294–311.

[3] R. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 18, no. 7, pp. 690–706, Jul. 1996.

[4] J. Yan and A. El Ahmad, "A Low-cost attack on a Microsoft CAPTCHA," in Proc. 15th ACM Conf. Comput. Commun. Security, 2008, pp. 543–554.

[5] R. Datta, J. Li, and J.Wang, "Imagination: A robust image-based CAPTCHA generation system," in Proc. 13th Annu. ACM Int. Conf. Multimedia, 2005, pp. 331–334.

[6] K. Kluever. (2008). Breaking the PayPal HIP: A Comparison of Classifiers [Online]. Available: http://ritdml.rit.edu/handle/1850/7813

[7] G. Mori and J. Malik, "Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., vol. 1. 2003, pp. 134–141.

[8] G. Moy, N. Jones, C. Harkless, and R. Potter, "Distortion estimation techniques in solving visual CAPTCHAs," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2. Jun.–Jul. 2004, pp. 23–28.

[9] A. Coates, H. Baird, and R. Faternan, "Pessimal print: A reverse turing test," in Proc. 6th Int. Conf. Document Anal. Recognit., 2001, pp. 1154–1158.

[10] M. Chew and H. Baird, "Baffletext: A human interactive proof," Proc. SPIE, vol. 5010, pp. 305–316, Mar. 2003.