SIDE-CHANNEL ATTACK ON THE HUMANAUTH CAPTCHA

Carlos Javier Hernandez-Castro, Arturo Ribagorda
Security Group, Department of Computer Science, Carlos III University, 28911 Leganes, Madrid, Spain

Yago Saez
EVANNAI Group (Artificial Neural Networks and Evolutionary Computation)
Carlos III University, Computer Science Department, Madrid, Spain

Keywords: CAPTCHA, HumanAuth, Image labeling, Watermarking, Automatic classification.

Abstract: We propose a new scheme of attack on the HumanAuth CAPTCHA which represents a significant shortcut to the intended attacking path, as it is not based in any advance in the state of the art on the field of image recognition. After analyzing the HumanAuth image database with a new approach based on statistical analysis and machine learning, we conclude that it cannot fulfill the security objectives intended by its authors. Then, we analyze which of the studied parameters for the image files seem to disclose the most valuable information for helping in correct classification, arriving at a surprising discovery. We also analyze if the image watermarking algorithm presented by the HumanAuth authors is able to counter the effect of this new attack. Our attack represents a completely new approach to breaking image labeling CAPTCHAs, and can be applied to many of the currently proposed schemes. Lastly, we investigate some measures that could be used to increase the security of image labeling CAPTCHAs as HumanAuth, but conclude no easy solutions are at hand.

1 INTRODUCTION

The last decade has seen increasing interest in abusing services provided in the Internet, mainly for economical reasons. There has been misuse of services like e-mail account creation for spam sending and phishing, abuse of sites where anonymous posting is encouraged (Wikipedia, blogs comments, news sites, etc.) for adding links for commercial promotion, harassment or vandalism. There has also been abuse of remote voting mechanisms (Hernandez, 1997). Automatic (script-guide) site wandering has also been described as a way to facilitate resource consumption and thus remote denial-of-service attacks. Anonymous abuse of on-line games (Golle and Ducheneaut, 2005), inclusive for commercial promotion, is not new. Other denial-of-service attacks on public posting sites (like employment listings or CV reception e-mails addresses) is also possible. Thus, there are lots of sounding economical reasons to abuse services provided through the Internet.

The main trend to prevent this automatic abuse has been to develop the ability to tell humans and computers apart - remotely and through an untrustworthy channel. Many tests -generically called CAPTCHAs[1] or HIPs[2]- have been developed with that aim. Those tests rely on capacities inherent to the human mind but supposedly difficult for computers, problems that have been traditionally hard to solve algorithmically in computers (as problems that still remain wide open for Artificial Intelligence researchers).

Moni Naor seems to have been the first (Naor, 1996) to propose theoretical methods of telling apart computers from humans remotely to prevent the abuse of web services. In 1997, primitive CAPTCHAs were developed by Andrei Broder, Martin Abadi, Krishna Bharat, and Mark Lillibridge to prevent bots from adding URLs to their search engine (Abadi, 1996). The term CAPTCHA was coined in 2000 by Luis von Ahn, Manuel Blum, Nicholas Hopper and John Langford of Carnegie Mellon University (Ahn et al., 2003). At the time, they developed the first CAPTCHA to be used by Yahoo. Those earlier designs were mostly text-based: the computer chose a random sequence

[1]Completely Automated Public Turing test to tell Computers and Humans Apart

[2]Human Interactive Proof

Javier Hernández-Castro C., Ribagorda A. and Saez Y. (2010).
SIDE-CHANNEL ATTACK ON THE HUMANAUTH CAPTCHA.
In *Proceedings of the International Conference on Security and Cryptography*, pages 59-65
DOI: 10.5220/0002994000590065

of letters and rendered them in an image after applying different kinds of distortions. The human challenger, supposedly far better than a computer in character recognition, was to identify the characters. But even after graphic distortion and degradation, some approaches have been able to "read them and thus solve the test automatically around 92% of the time (Mori and Malik, 2003), specially so when it is possible to divide the graphic into its constituent letters. Some approaches have focussed on making this division (segmentation) harder, typically to the expense of making it also harder to the human challenger.

Some CAPTCHAs that rely on similar grounds as text CAPTCHAs may seem slightly stronger, but are also more difficult for the common user, so can only be used as special-purpose CAPTCHAs for certain types of human clients. Among them there is MAPTCHA, the Mathematical CAPTCHA, that shows a math formula and asks the user its numerical solution. Similar CAPTCHAs have been shown to be easier than expected (Hernandez-Castro and Ribagorda, 2009a).

1.1 Image CAPTCHAs

General vision seems to be a harder problem than character recognition, so more designs have focused on using pictures instead - even though most of those CAPTCHAs do not really rely on a "general vision problem but in a downsized version of categorizing images.

Figure 1: Results from Google Image Search for "river.

Chew and Tygar (Chew and Tygar, 2004) were the first to use a set of labelled images to produce CAPTCHAs challenges. For that purpose, they used the labelled associated with images in Google Image Search. This technique is not well suited for CAPTCHAs, as Google relates a picture to its description and its surroundings (figure 1).

Ahn and Dabbish (von Ahn and Dabbish, 2004) proposed a new way to label images by embedding the task as a game, the "ESP game (figure 2). However, it has a fixed number of object classes (70) and the image database seems not large enough. The site HotCaptcha.com (figure 3) used a large-scale human labeled database provided by the HotOrNot.com web-

Figure 2: The ESP game.

site, a site that invites users to post photos of themselves and rate others' in a numerical scale of "hotness. This proposal is no longer active, as in January of 2009, HotCaptcha.com was down. Oli Warner had the idea of using photos of kittens to tell computers and humans apart (Warner, 2006).

The main trend in image-labeling CAPTCHAs has been to use already classified images, either from internal databases or "crowd computing (typically using on-line databases) of pictures. The first proposal was KittenAuth, which features nine pictures of animals, only three of which are feline. Its database of pictures is small enough (< 100) to manually classify them, and this limitation seems troublesome even if we apply new methods involving image distortion. ASIRRA (Elson et al., 2007) uses a similar approach but using a giant database of more that 3 million photos from Petfinder.com, a web-site devoted to finding homes for homeless pets. It has been later shown weak against machine learning attacks (Golle, 2008) and side-channel attacks (Hernandez-Castro et al., 2009).

HumanAuth, the CAPTCHA that we analyze in this article, is somewhat related to both KittenAuth and ASIRRA. It is also a image-labeling CAPTCHA, that uses an internal database of pictures that depict items that are either artificial (a car, a bar, a wheel) or natural (a bird, a river, etc.). The user has to select the ones that are natural, that is, depict no artificial item. HumanAuth is a very interesting proposal in many terms: it proposes a picture classification problem that goes well beyond two different types of pictures (as KittenAuth and ASIRRA), thus relying on a harder (from the AI standpoint) problem, as solving it using the intended path means understanding what a picture represents, and then deciding if that item is or not artificial. On the other point, it is also one of the few CAPTCHAs available for vision impaired people, as every picture is associated to a description ("green flowers, "wall paint, etc.). A problem with HumanAuth is that the database of pictures it proposes is small, although it includes a watermarking algorithm to difficult indexing (of pictures: not of descriptions). The attack path we propose here is usable against any image-labeling CAPTCHA, and we have selected HumanAuth as an example because of the included picture watermarking algorithm.

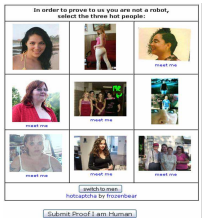

Figure 3: An example of a HotCaptcha challenge.

1.2 Motivation

We know that the problem in which a new CAPTCHA scheme is based has to be easy for humans, and hard for machines. But, more than that, it has to be that way not in a general circumstance, but in the implementation that a particular CAPTCHA proposes - which is always a subset of a more general problem. Basically, there are two questions that remain to be addressed:

1. Is a particular CAPTCHA open to a side-channel attack? We do not have a clear methodology to check whether a particular CAPTCHA's design and implementation is open to a side-channel attack. A side-channel attack is one that solves the CAPTCHA but not the artificial intelligence problem upon which it is based, therefore not improving the state-of-the-art of AI. It is named side-channel, as it solves the problem using a method that does not follow the intended attacking path.
2. Is the hardness of the underlying AI problem fully transmitted to the CAPTCHA design? We lack a method to check if the particular problem that a CAPTCHA challenge poses is as hard as the underlying AI problem.

Recent CAPTCHA design history is filled with failures, which have made them much weaker than intended. In recent work (Hernandez-Castro and Ribagorda, 2009a; Hernandez-Castro et al., 2009; Golle, 2008; Hernandez-Castro and Ribagorda, 2009b), it has been shown some of security problems of different CAPTCHA schemes. By analyzing those CAPTCHA proposals we can find and classify various flaws. That is why it is so interesting to try to break current CAPTCHAs, especially in a way that helps to find pitfalls in their design, to make the state-of-the-art advance and get to a point when well known and tested assumptions give base for more secure designs.

1.3 Organization

The rest of the paper is organized as follows: In the next section, we introduce the HumanAuth CAPTCHA in greater detail. After this, in Section 3 we describe the ENT tool, which will be helpful in the attack against HumanAuth described in Section 4. Later, we analyze if the watermarking algorithm proposed by HumanAuth is able to counter our attack. Finally, in Section 5 we extract some conclusions and propose possible improvements together with future research lines.

2 THE HumanAuth CAPTCHA

The HumanAuth CAPTCHA[3] is based on the ability of humans to distinguish between images with natural and non-natural contents. The source code of the HumanAuth application comes with a image repository consisting of 45 nature images and 68 non-nature ones in jpeg format. The idea is quite interesting, and the CAPTCHA is specially easy for humans and purportedly very difficult for non-humans. The user is presented with a collection of 9 pictures, and has to click on the three that represent items that are natural, that is, no artificial. HumanAuth also includes a watermarking algorithm (figure 4) to difficult indexing of pictures.

Figure 4: HumanAuth CAPTCHA capture with the GigoIt logo as an example watermark.

The HumanAuth CAPTCHA is implemented in PHP. The authors are Peter Schmalfeldt and John Kramlich, working for the GigoIt Inc. association, who released it under a GNU GPLv2[4].

3 DESCRIPTION OF THE ENT TOOL

The ENT (Walker, 2008) tool is a program comprising a suite of statistical tests that check for information density and randomness in a byte sequence. It applies various tests to the sequence of bytes and reports

[3] http://sourceforge.net/projects/humanauth/
[4] http://www.gnu.org/licenses/gpl-2.0.html

their numerical results. As these tests evaluate information quantity and randomness quality, this results are of interest for evaluating pseudorandom number generators, and studying the output of compression algorithms. In the novel way presented in this paper, they can also be used for file/image classification. The tests included in the ENT program are:

- Entropy: information density of the contents of the file, expressed as the mean number of bits necessary to represent a character of 8 bits (byte).
- Compression: this test tells us the size shrink (in percentage) we could obtain if the file was compressed using a lossless compression algorithm of the Lempel-Ziv type (one pass).
- Chi-square test: this tests computes the expected p-value for a distribution with 256 degrees of liberty (dividing the file in 8 bit chunks of data). This p-value represents how frequently a uniform distribution would exceed the computed value.
- Arithmetic mean: this tests computes the mean value of all the bytes of the input file.
- Monte-Carlo value for Pi: uses a Monte-Carlo (probabilistic) algorithm to compute the value of Pi, using the input file as the source of randomness for such algorithm.
- Serial correlation: measures how a byte of the file can be approximated by its preceding byte.

For an idea of the results one can expect of the ENT test, we show the output obtained when applied to the following input files:

- An ASCII English version of Don Quijote de la Mancha, by Miguel de Cervantes, from the Gutemberg Project [5]
- A BMP non-compressed image [6]
- A WAV non-compressed sound [7]
- A JPEG compressed image from Flickr.com [8]
- The Chase, an MP3 compressed music file from the Internet Audio Open Source Archive [9]

The entropy and compression tests (table 1) give information about data density. The mean and Monte Carlo Pi tells us about value distribution, the serial correlation about data interrelation (also data redundancy), and the chi square test is the most sensible to non random data distribution.

Table 1: Results of the ENT test for different kinds of files.

test	*ASCII*	*BMP*	*WAV*	*JPEG*	*MP3*
size	2347772	1683594	116904	4914423	2916331
entropy	4.49	7.24	6.16	7.91	7.85
compression	43 %	9 %	22 %	1 %	1 %
chi square	37,346,041	2,796,525	484,762	669,239	1,643,472
chi square p-value	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
mean	88.92	73.21	125.74	137.88	119.89
Monte-Carlo Pi	4.00	3.45	3.97	2.84	3.20
serial correlation	0.016057	0.537042	0.928775	0.004862	0.168447

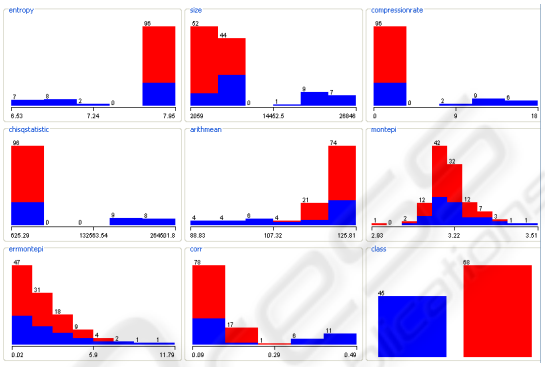

Figure 5: Distribution of classes for each ENT test result.

4 ATTACK TO HumanAuth

We downloaded the image database of the HumanAuth CAPTCHA, and analyzed all of the jpeg files contained herein with the help of the ENT tool, producing a formatted output (in ARFF format, so to be used with Weka (Winiwarter and Kambayashi, 1997)) with its results. As can be seen in figure 5, although some of the ENT tests do produce interesting information for classification, none of them is able of producing a good classifier on its own.

Then, we test different classifiers for approaching the classification of human made (artificial) vs. from nature. Even though we are processing the compressed file, the best classifier (in this case RandomForest) was able to show an accuracy rate of 77.8761% (table 2), which is significantly better than the $\frac{68}{68+45} = 60.177\%$ that a trivial classifier (that always predicts the larger class) will do.

The information disclosure of the values that ENT outputs is distributed very evenly, except for the values of the MonteCarlo estimation of π (table 3), implying that devising measures to difficult side-channel attacks against this scheme is going to be specially hard.

Even after completely removing the three most significant attributes for the HumanAuth classifier,

[5] http://www.gutenberg.org/files/996/996.txt

[6] http://www.lossip.com/wp-content/uploads/marc-anthony-y-ricardo-arjona.bmp

[7] http://amazingsounds.iespana.es/oceanwaves.wav

[8] http://farm4.static.flickr.com/3126/3153559748_b0ee7fd24b_o.jpg

[9] http://www.archive.org/download/Green-LiveAtSmallTheAQ/TheChase.mp3

Table 2: Classification of the HumanAuth image database using a RandomForest classifier.

```
=== Run information ===
Scheme:       weka.classifiers.trees.RandomForest -I 10 -K 0 -S 1
[.....]
Test mode:    10-fold cross-validation
Random forest of 10 trees, constructed considering
                                4 random features.
=== Summary ===
Correctly Classified Instances          88               77.8761 %
Incorrectly Classified Instances        25               22.1239 %

=== Confusion Matrix ===
  a  b   <-- classified as
 34 11 |  a = nature
 14 54 |  b = nonnature
```

Table 3: Information disclosure of the different values for classification.

```
=== Run information ===
Evaluator:    weka.attributeSelection.ChiSquaredAttributeEval
[.....]
Evaluation mode:    10-fold cross-validation
=== Attribute selection 10 fold cross-validation (stratified),
seed: 1 ===
average merit      average rank  attribute
38.987 +- 5.975     1.9 +- 1.58    8 corr
34.58  +- 3.872     2.6 +- 1.2     3 compressionrate
32.936 +- 6.025     2.7 +- 0.78    2 size
29.844 +- 4.878     3.9 +- 0.83    1 entropy
29.885 +- 4.57      4.6 +- 1.85    4 chisqstatistic
27.228 +- 1.393     5.3 +- 0.46    5 arithmean
 0     +- 0         7.2 +- 0.4     7 errmontepi
 0     +- 0         7.8 +- 0.4     6 montepi
```

that is correlation, compression rate and size, a SMO algorithm is able of achieving a 75.2212% accuracy.

As table 4 shows, it seems extremely difficult to protect these set of images against the proposed side-channel analysis, which seriously casts a doubt over the security of the derived HumanAuth CAPTCHA.

4.1 HumanAuth CAPTCHA Watermarking

To prevent easy image library indexing, the authors of the HumanAuth CAPTCHA decided to randomly merge a PNG image with the JPG image taken from the library. It locates the PNG in a random position into the JPG canvas and merges both using a certain of transparency, so the PNG appears as a watermark, not distorting the JPG image as much as to make it difficult for the human eye to recognize the image. The HumanAuth source includes one PNG image (a logo) that can be used for testing.

We have created a set of 20,000 images, 10,000 of each class (nature and non-nature) (figure 6) and have extracted statistical information from them with the ENT tool, building an ARFF file for Weka processing.

After analyzing it, we have tried different classifiers, obtaining the following results (table 5).

This result can be somewhat expected: with the

Table 4: Different classification accuracies by classifier and parameters allowed.

classifier	*non-allowed parameters*	*parameters used*	*accuracy*
RandomForest	-	all	77.8761 %
SMO	correlation, compression rate, size	rest	75.2212 %

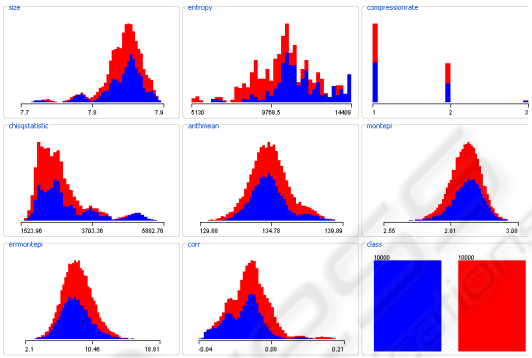

Figure 6: Distribution of classes for the different ENT tests, 20000 images.

little randomness introduced by merging the logo in a random position, the precision of the classifier slightly drops to 72.585 %.

Using a classification tree of 481 leaves, J48 is able to reach almost a 91 % accuracy (table 6). This is due to the "repetition of images, as we have created an image set of 20,000 images which are statistically close (enough) to the 45 original ones. This suggests that the initial small set of images, when used with the scheme proposed by the HumanAuth authors of merging with a watermark, may not be of use against this type of attack, even though might be enough to prevent hash-function (like MD5) indexing - which probably was the intention of the designers. One can argue that we can chose a different watermark that alters more the original image, but that would be also at the expense of human visual recognition. It can also be argued that other possible approach could be randomly using a set of different watermarks, but that would be at the expense of creating an appropriate set, so we are just moving/distributing the original problem (images can be characterized because of a not enough uniform distribution). As can be expected when using as a seed such a small set of images, the classification accuracy raises as enough samples are given to build a good enough decision tree, reaching its top logarithmically (figure 7). The figures given here (table 7) correspond to another test set of 20000 images we have created:

Other classification schemes show even slightly better results than J48 (table 8). These results (table 9) are still good after the most significant attributes are removed from the classification scheme.

Table 5: Classification of 20K HumanAuth randomly watermarked images, using a DecisionStump tree classifier.

```
=== Run information ===
Scheme:       weka.classifiers.meta.LogitBoost -P 100 -F 0 -R 1 -L
              -1.7976931348623157E308 -H 1.0 -S 1 -I 10 -W
              weka.classifiers.trees.DecisionStump
Instances:    20000
[.....]
Test mode:    10-fold cross-validation
=== Classifier model (full training set) ===
LogitBoost: Base classifiers and their weights:
[.....]
=== Summary ===
Correctly Classified Instances       14517               72.585  %
Incorrectly Classified Instances      5483               27.415  %
Kappa statistic                          0.4517
Mean absolute error                      0.3631
Root mean squared error                  0.4229
Relative absolute error                 72.6115 %
Root relative squared error             84.5869 %
Total Number of Instances            20000
[.....]
=== Confusion Matrix ===
    a    b   <-- classified as
 8093 1907 |    a = nature
 3576 6424 |    b = nonnature
```

Table 6: Classification of 20K HumanAuth randomly watermarked images, using a J48 classifier.

```
=== Run information ===
Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     humanauth
Instances:    20000
Attributes:   9
[.....]
Test mode:    10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
------------------
[.....]
Number of Leaves  :     481
Size of the tree :  961
[.....]
=== Summary ===
Correctly Classified Instances       18187               90.935  %
Incorrectly Classified Instances      1813                9.065  %
Kappa statistic                          0.8187
Mean absolute error                      0.1113
Root mean squared error                  0.2776
Relative absolute error                 22.2538 %
Root relative squared error             55.5252 %
Total Number of Instances            20000
[.....]
```

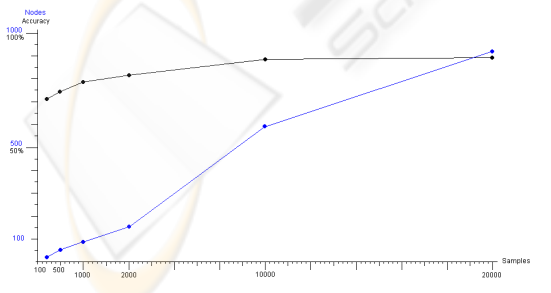

Figure 7: Increase in accuracy and number of nodes with images available.

Table 7: Classification accuracy for different numbers of watermarked images.

Images	*Nodes*	*Accuracy*
200	21	71.5 %
500	61	74.4 %
1000	85	78.6 %
2000	159	82.1 %
10000	597	88.8 %
20000	929	89.7 %

Table 8: Classification of 20K HumanAuth randomly watermarked images, using a RandomForest classifier.

```
=== Run information ===
Scheme:       weka.classifiers.trees.RandomForest -I 10 -K 0 -S 1
Relation:     humanauth
Instances:    20000
Attributes:   9
[.....]
Test mode:    10-fold cross-validation
=== Classifier model (full training set) ===
Random forest of 10 trees, constructed considering 4 random features
=== Summary ===
Correctly Classified Instances       18538               92.69   %
Incorrectly Classified Instances      1462                7.31   %
Kappa statistic                          0.8538
Mean absolute error                      0.1133
Root mean squared error                  0.2327
Relative absolute error                 22.658  %
Root relative squared error             46.5399 %
Total Number of Instances            20000
[.....]
```

5 CONCLUDING REMARKS

We address in the following, Section 5.1, the generality of the presented attack, and later some conclusions and ideas for future works.

5.1 Attack Generality

Our approach can be used as a very general analysis tool to realistically estimate the security parameters of any image-labeling CAPTCHA proposal, and we believe it will be advisable to use it in the future before similar systems are launched, to have adequate, well-reasoned, and founded security parameters and realistic estimations. One of its main advantages is that it does not depend on the underlying format (image, sound, video, etc.) or problem, and that it could be useful for avoiding pitfalls such as the existence of some trivial and irrelevant parameter values (i.e. size, or byte correlation) leaking too much class-relevant information. We have used this approach against other image-labeling CAPTCHAs with success, and are in the process of improving it for better image categorization.

Table 9: Classification accuracies for watermarked images, per classifier and parameters allowed.

classifier	*non-allowed parameters*	*parameters used*	*accuracy*
LogitBoost / DecisionStump	-	all	72.585 %
J48	-	all	90.935 %
RandomForest	-	all	92.69 %
RandomForest	correlation, compression rate, size	rest	82.805 %

5.2 Conclusions and Future Work

HumanAuth is a very interesting image-labeling CAPTCHA. Its basic idea is theoretically more powerful that that behind KittenAuth or ASIRRA, as it relies on a broader image recognition and classification problem: it delves into many different types of pictures (much more than KittenAuth and ASIRRA), thus relying on a harder AI problem. But, although the HumanAuth, along with many other image-labeling CAPTCHAs, are very interesting, our work has shown that their security is not carefully studied before they are put into use.

With the attack presented here, we are able to successfully bypass the HumanAuth challenge 92+% of occasions, as proposed by its authors (image library and watermarking image). This is an incredibly successful figure, as normally, CAPTCHAs that are possible to automatically bypass as low as 5% (or even less) are considered broken, taking into consideration that a program can try to bypass the CAPTCHA many times per second.

The lessons learned in this analysis are useful to improve other attacks based on more common approaches -like image processing- or, alternatively, can be used to improve the security of these CAPTCHA schemes, and this could be an interesting future work. One particulary interesting way would be filtering images taken from image databases, to force then to have a much similar average size and less standard deviation (among other statistical properties), which could harden a lot the task of the attacker, without affecting the overall good properties of the CAPTCHA proposal.

REFERENCES

Abadi, M. (1996). Method for selectively restricting access to computer systems. US Patent no. 6,195,698.

Ahn, L. V., Blum, M., and Langford, J. (2003). Captcha: Using hard ai problems for security. In *Proceedings of Eurocrypt*, pages 294–311. Springer-Verlag.

Chew, M. and Tygar, J. D. (2004). Image recognition captchas. In *Proceedings of the 7th International Information Security Conference*, pages 268–279.

Elson, J., Douceur, J. R., Howell, J., and Saul, J. (2007). Asirra: A captcha that exploits interest-aligned manual image categorization. In *Proceedings of 14th ACM Conference on Computer and Communications Security (CCS), Association for Computing Machinery*.

Golle, P. (2008). Machine learning attacks against the asirra captcha. In *ACM Conference on Computer and Communications Security*, pages 535–542.

Golle, P. and Ducheneaut, N. (2005). Preventing bots from playing online games. In *Proceedings of the ACM Computers in Entertainment, Vol. 3, No. 3*.

Hernandez, J. C. (1997). Compulsive voting. In *Proceedings of the 36th Annual 2002 International Carnahan Conference on Security Technology*, pages 124–133.

Hernandez-Castro, C. J. and Ribagorda, A. (2009a). Pitfalls in captcha design and implementation: the math captcha, a case study. *Computers & Security*.

Hernandez-Castro, C. J. and Ribagorda, A. (2009b). Remotely telling humans and computers apart: an unsolved problem. In *Proceedings of the iNetSec 2009, IFIP AICT 309*.

Hernandez-Castro, C. J., Ribagorda, A., and Saez, Y. (2009). Side-channel attacks on labeling captchas. http://arxiv.org/abs/0908.1185.

Mori, G. and Malik, J. (2003). Recognizing objects in adversarial clutter: Breaking a visual captcha. In *Computer Vision and Pattern Recognition CVPR03*, pages 134–141.

Naor, M. (1996). Verification of a human in the loop or identification via the turing test. Technical report, Weizmann Institute of Science.

von Ahn, L. and Dabbish, L. (2004). Labeling images with a computer game. In *ACM Conference on Human Factors in Computing Systems*, pages 319–326.

Walker, J. (2008). Ent: A pseudorandom number sequence test program. http://www.fourmilab.ch/random/.

Warner, O. (2006). Kittenauth. http://www.thepcspy.com/kittenauth.

Winiwarter, W. and Kambayashi, Y. (1997). Y.: A machine learning workbench in a dood framework. In *Proc. of the Intl. Conf. on Database and Expert Systems Applications*, pages 452–461.