A FEW SIMPLE GUIDELINES RELATED TO IMAGE CAPTCHAS

Tyler Menezes

University of Washington
Seattle, WA, USA

Darko Kirovski

Microsoft Research
Redmond, WA, USA

ABSTRACT

Human interactive proofs (HIPs) or CAPTCHAs are a common tool to prevent automated attacks on Web services. Their prime objective is to detect whether a human is controlling the actions of a specific client. Although vulnerable to various adversarial activities, HIPs are simply a necessity when minimizing costs of running a Web service.

In this paper, we analyze the existing, widely adopted practice to assemble a HIP puzzle by warping an image that encapsulates a sequence of randomly selected, printed letters. We reach the following conclusions. First, on a large database of fonts, we showcase that various parameterization techniques such as font variability, are unlikely to maintain the efficacy of a specific HIP setting. Second, we point to the fact that it is suboptimal to use all letters of the alphabet as candidates for constructing HIPs; to that extent, we present an optimal procedure for letter selection that achieves minimal overall true positive detection rate. Finally, we point to the fact that under such circumstances humans are not necessarily always worse than computers in solving HIPs — a belief that has been promoted in related work.

***Index Terms*—** Automated Turing test, CAPTCHA, human interactive proof.

1. INTRODUCTION

Human Interactive Proofs (HIPs), also frequently known as CAPTCHAs, are a common tool to prevent automated attacks on Web services [1]. Their prime objective is to detect whether a human is controlling the actions of a specific client. Widely adopted practice is to assemble a HIP puzzle by warping an image that encapsulates a sequence of randomly selected, printed letters. A server prepares such a puzzle and sends it to the client when it raises suspicion of being adversarial with respect to the normal operation of the Web service. For example, a HIP is commonly sent to an e-mail client suspected of sending spam, i.e., when it sends a message to a relatively large number of destinations who have never contacted the sender.

Here the server defends the core of its functionality: Users expect to reach a service's functionality with minimal annoyances like complicated HIPs while still having filtered unsolicited spam. Meanwhile, 88-92% of all e-mail sent on the Web is spam according to some estimates [2] — this puts tremendous overload on hosts' traffic and storage.

To fight HIPs, adversaries usually use one of the following techniques:

- **Character Recognition Solvers** Attackers often deploy optical character recognition software (OCR) and automate the process of solving HIP puzzles using a collection of computers on the Web or a local network [3, 4, 5, 6].
- **Human Solvers** An attacker's automated attack relays pose HIPs to human solvers who provide answers (usually in exchange for free content [11]). "Deployed" humans in this case may even not know that their HIP solutions are used to generate spam.
- **Blended** Attackers may apply OCRs to solve a majority of HIPs, and send ones with a low certainty to human solvers. This is similar to the technique used by Google to digitize books with their ReCAPTCHA service [7].

Spam on the other hand has an economic backbone — roughly US$300 are provided to spammers for delivering one million messages to user's inboxes. To conclude, although vulnerable to various adversarial activities, HIPs are simply a necessity when minimizing costs of running a Web service.

1.1. Contributions

In this paper, we focus on analyzing HIP performance under the OCR attack only. We first issue a guideline to the adversary: regardless of the underlying variability of fonts used to construct HIPs, a successful strategy to build an attack OCR is to train it on a set of images where each letter is rendered using a single carefully selected font. We observed that this adversarial strategy provided detection performance comparable to the ccase when the OCR is trained using full knowledge of the font used to generate the HIP. Our strategy clearly simplifies the adversarial complexity in case of HIPs constructed using variable fonts. As a consequence we speculate that HIP parameterization via font selection is an ineffective way of maintaining HIP generators successful.

Second, we point to the fact that it is suboptimal to use all letters of the alphabet as candidates for constructing HIPs; to that extent, we present a procedure for letter selection that achieves minimal overall true positive detection rate. In other words, it is not effective to use letters that are easily disambiguated by OCR for constructing HIPs – they only reduce the overall efficiency of the HIP system.

At last, we note that under such circumstances humans are not necessarily always worse than computers in solving HIPs – a belief that has been promoted in related work [12].

2. TRAINING A HIP SOLVER

In this section, we review the practice that benefits the adversary: the process of training her automated HIP solver. Such a solver typically consists of two components:

- a segmentation engine, whose task is to isolate individual characters to be decoded, and
- a single character recognition (SCR) engine.

In this paper, we focus on the latter component for two reasons. First, it is unclear whether a segmentation engine is truly required if SCR is accurate; by applying a fast SCR tool on a moving window that follows the trace of the HIP, maximizing for the overall confidence in detection for all letters, one could envision a solver that can rely only on an SCR engine. The converse, i.e., a HIP solver designed without an SCR engine, is certainly less likely to craft. Second, techniques that focus on making segmentation difficult, e.g., [8], typically affect strongly the readability of HIPs and make system valuations more complicated.

For the remainder of this report, we adopt an existing SCR engine, a convolutional neural network [9] which has been widely used for document processing and achieves an overall error rate of 0.4% on the MNIST database of handwritten digits. We speculate that this limitation still makes our results general, as we expect that the phenomena observed and detailed in this report should manifest with other SCR engines.

2.1. Font Parameterization

In our first experiment, we want to understand how font parameterization affects the SCR engine. The objective is simple from the HIP designer's perspective – use a specific font until it is observed that the adversary has identified how to solve such HIPs, then switch to another font, forcing the attackers to retrain their SCR – a task that would impose a nontrivial economic obstacle to the adversary.

For the experiment we chose a subset of fonts available on a default Windows 7 installation. We excluded fonts which, after warping, would not be readable by humans, for example "dingbat" fonts, leaving us with approximately 200 designs. We used the 26 uppercase alphabetic characters from each of these fonts due to practical constraints. From each font and each letter in the training set, we produced 50 variations of the letter by rotating the letter between 0 and 30 degrees and applying a global warp, which produced total-character deformations, and a local warp, that caused localized waves throughout the strokes of the letter [10]. Because it has previously been shown that computers perform significantly better than humans at high levels of warp [9], we bounded the warping magnitude so that a quick glance over the resulting warped letters would reveal an expectation of exceptionally high recognition rates (97+%) by humans.

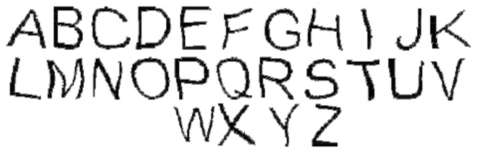

Fig. 1. Illustration of results of applying our letter warping procedure when constructing HIPs.

Of each 50 variations, 30 were placed into a training database and 20 were placed into a test database. This provided a training database of roughly the same size used in previous experiments with this SCR engine [12] while providing a fully representative test set. The SCR was trained on the 156,000 character training database and then accuracy was calculated by applying it across the 104,000 test characters. From this initial experiment, we observed an overall accuracy similar to that reported in previous reports [12].

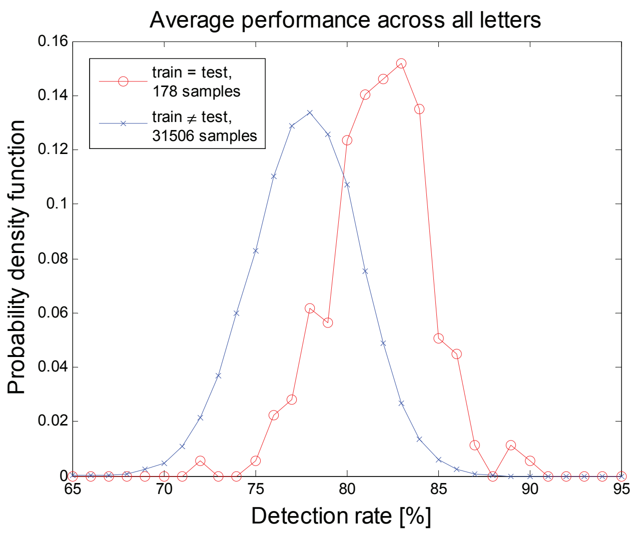

Fig. 2. Histogram of SCR performance in two distinct cases: first when training and testing are done on the same base font, and second, when the SCR was trained using one then applied to another font. Each performance result quantifies the average detection rate across the full English language alphabet.

Figure 2 reveals no surprises as SCRs trained on a specific font performed substantially better when tested on the same exact font rather than any other font in the database (81.7% vs. 77.7%). This performance is reported as an average across all letters of the English alphabet.

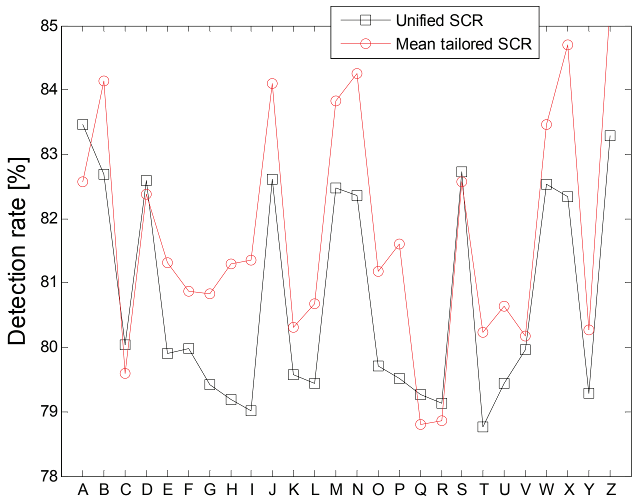

Fig. 3. Per-letter average performance of an SCR trained and tested on the same font and a unified SCR trained using the best font for each letter individually and tested across fonts in database.

Next, we devised a *unified SCR* by selecting for each letter a font that resulted in best detection performance on all occurrences of this letter across the entire font database. This way, the adversary would produce a single SCR engine that could be used to attack HIPs constructed using arbitrary fonts. All of that with hope that the performance of the unified SCR would lag only slightly the performance of SCRs optimized for specific fonts. Figure 3 illustrates the per-letter performance of the two SCR engines averaged of the entire font database – we recorded difference in performance equal to only 1% in favor of the tailored SCR – clearly, considering the universality of the resulting SCR, a success for the adversary. To compare the average performance of these two SCR engines we present Figure 4.

We conclude that HIP parameterization via continuous font substitutions, an approach that has been commonly believed as successful in forcing adversaries spend non-trivial effort to break HIPs (e.g., labeling and SCR training), is likely inefficient in lieu of the concept of a unified SCR engine trained to perform nearly equally well as an SCR tailored to an arbitrary font.

2.2. Letter Selection

We now turn our attention from the attacker to the party deploying HIPs. Looking at Fig.3, we observe that there is variance in detection rate across the deployed alphabet. Given

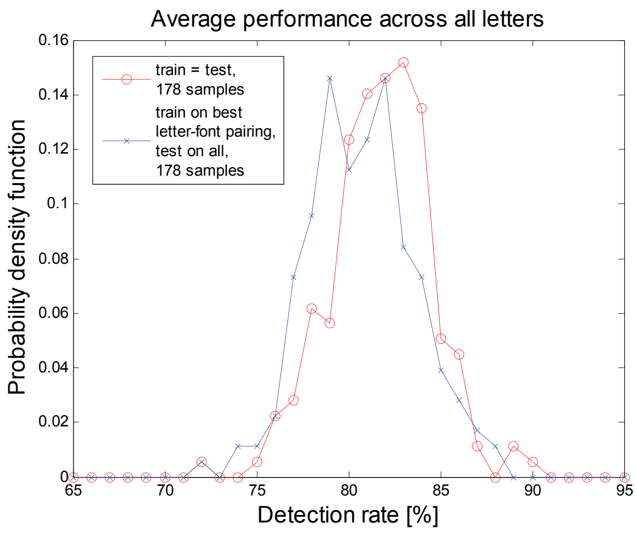

Fig. 4. Average performance of an SCR trained and tested on the same font and a unified SCR trained using the best font for each letter individually and tested across the font database.

this, we pose a question whether by removing a carefully selected subset of letters from the considered alphabet we can increase the overall ambiguity, i.e., improve system performance.

A Glyph Confusion Matrix was generated from the OCR generated as a result of the previous experiment. The matrix shows the percentage of times the OCR guessed a letter on a letter-by-letter basis. An ideal OCR would have a vertical line from the top-left to bottom-right, indicating 100% success at identifying each glyph and 0% confusion.

In Fig. 5 we can specifically see how certain letters were very infrequenly confused for the OCR — D, M, N, and S. We conclude that, in order to increase the overall ambiguity of the puzzle, these letters should be eliminated.

We perform a simple depth-first search for the permutation which produced the highest overall error rate:

Alphabet $\mathbb{F} = f_1 \ldots f_26$

error rate = $1 - \sum f_i^{\mathbb{F}} pr[f_i|f_i, F^*]$

Objective
$\forall \mathbb{F}^* \in \mathbb{F}$: arg max $\sum_{f_i \in f^*} pr[f_i|f_i]$

Selection Method
```
for ($f^*$ in $\mathbb{F}$)
$\sum_{f^* \setminus \mathbb{F}} pr[f_i|f_i] > \epsilon$
end
```

We conclude that any given image-based HIP, such a

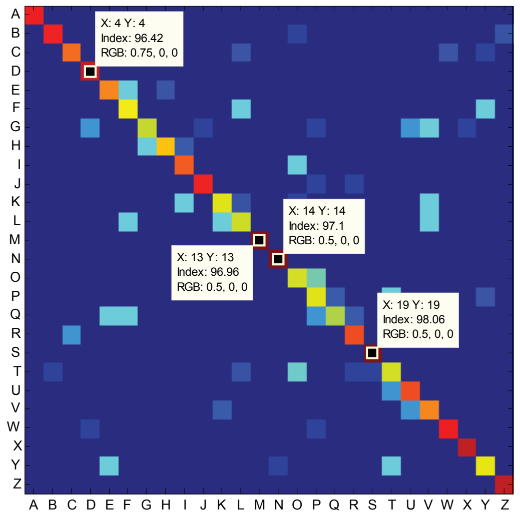

Fig. 5. A confusion matrix showing the confusion between various letters for the OCR.

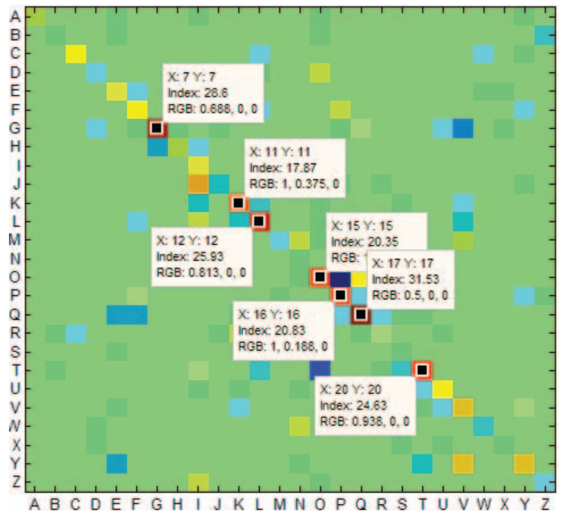

Fig. 6. A confusion matrix showing the difference in confusion between the OCR and humans.

method can be used to find the most effective set of letters given a sufficiently large train & test database. Over our experiments we observed a decline of 4% in accuracy of the OCR after performing the set reduction, a very significant improvement for spam-filtering, and quite surprising considering the simplicity of the method.

3. USER PERFORMANCE

We conducted a user study which replicated the HIPs presented to the computer, simulating a live production environment. HIPs were generated as described in the OCR Training section above. Study participants were asked to identify these characters in sets of five; the characters were spaced 50 pixels apart to eliminate any need for segmentation by the user. The study was conducted electronically, and participants were given ten minutes to identify as many of the pairs as possible, and were scored by the total pairs answered correctly multiplied by the accuracy. The highest scoring participant was compensated with an Xbox and Kinect video game console. Participants were 144 employees and interns at a software company.

For purposes of statistical significance, we chose to limit the number of fonts from the user study to one sans-serif font, one serif font, and one block-serif font, which we felt represented the majority of fonts used in HIPs.

We found that users performed at an average of 98.4% accuracy on 4,230 HIP puzzles taking an average of 4.2 seconds to respond to the five-character puzzle presented. We generated a confusion matrix from the user data, and then subtracted the confusion matrix previously generated for the OCR (Fig. 6). This figure allowed us to identify points where the users performed much better than the machines. We conclude that the selection of letters is more significant than the amount of distortion applied to the HIP puzzle.

4. REFERENCES

[1] L. von Ahn et al. CAPTCHA: Using Hard AI Problems for Security. Eurocrypt, 2003.

[2] Messaging Anti-Abuse Working Group. Email Metrics Program: The Network Operators Perspective. Report (no.12) Third and Fourth Quarter 2009, issued March 2010.

[3] G. Moy et al. Distortion estimation techniques in solving visual CAPTCHAs. IEEE CVPR, 2004.

[4] G. Mori and J. Malik. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. IEEE CVPR Recognition, 2003.

[5] J. Yan and A. S. El Ahmad. A low-cost attack on a Microsoft CAPTCHA. ACM CCS, pp.543–554, 2008.

[6] K. Chellapilla and P. Simard. Using machine learning to break visual human interaction proofs. NIPS, 2004.

[7] L. von Ahn et al. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. Science, 2008.

[8] K. Chellapilla et al. Designing Human Freiendly Human Interaction Proofs. ACM Human Factors In Computing Systems, 2005.

[9] P. Simard et al. Best Practice for Convolutional Neural Networks Applies to Visual Document Analysis, International Conference on Document Analysis and Recognition, 2003.

[10] R. Deriche. Fast Algorithms for Low-Level Vision. IEEE PAMI, 1990.

[11] C. Doctorow. Solving and creating CAPTCHAs with free porn. Boing Boing, 2004.

[12] K. Chellapilla et al. Computers beat Humans at Single Character Recognition in Reading based Human Interaction Proofs (HIPs). Conference on Email and Anti-Spam, 2005.