Henry S. Baird
Daniel P. Lopresti (Eds.)

LNCS 3517

Human Interactive Proofs

Second International Workshop, HIP 2005
Bethlehem, PA, USA, May 2005
Proceedings

Springer

Henry S. Baird Daniel P. Lopresti (Eds.)

Human Interactive Proofs

Second International Workshop, HIP 2005
Bethlehem, PA, USA, May 19-20, 2005
Proceedings

Springer

Volume Editors

Henry S. Baird
Daniel P. Lopresti
Lehigh University
Computer Science and Engineering Department
Bethlehem, Pennsylvania, USA
E-mail: {baird,lopresti}@cse.lehigh.edu

Library of Congress Control Number: 2005926089

CR Subject Classification (1998): I.4, I.5, H.4, C.2, D.4.6, K.4.4, K.6.5, H.3

ISSN 0302-9743
ISBN-10 3-540-26001-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-26001-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN: 11427896 06/3142 5 4 3 2 1 0

Preface

E-commerce services are suffering abuse by programs (bots, spiders, etc.) masquerading as legitimate human users. Efforts to defend against such attacks have, over the past several years, stimulated investigations into a new family of security protocols – "Human Interactive Proofs" (HIPs) – which allow a person to authenticate herself as a member of a given group: e.g., as a human (vs. a machine), as herself (vs. anyone else), as an adult (vs. a child). Most commercial uses of HIPs today are CAPTCHAs, "Completely Automatic Public Turing tests to tell Computers and Humans Apart," which exploit the gap in ability between humans and machine vision systems in reading images of text. HIP challenges can also be non-graphical, e.g., requiring recognition of speech, solving puzzles, etc.

We are pleased to present the first refereed and archivally published collection of state-of-the-art papers on HIPs and CAPTCHAs. Each paper was reviewed by three members of the Program Committee, judged by the Co-chairs to be of sufficient relevance and quality, and revised by the authors in response to the referees' suggestions.

The papers investigate performance analysis of novel CAPTCHAs, HIP architectures, and the role of HIPs within security systems. Kumar Chellapilla, Kevin Larson, Patrice Simard, and Mary Czerwinski describe user trials of a CAPTCHA designed to resist segmentation attacks, including a systematic evaluation of its tolerance by human users. Henry Baird, Michael Moll, and Sui-Yu Wang analyze data from a human legibility trial of another segmentation-resistant CAPTCHA and locate a highly legible engineering regime. Amalia Rusu and Venu Govindaraju describe research towards CAPTCHAs based on reading synthetically damaged images of real images of unconstrained handwritten text. Yong Rui, Zicheng Liu, Shannon Kallin, Gavin Janke, and Cem Paya discuss the results of experiments with human subjects presented with two kinds of CAPTCHAs: one based on reading text, and a new one based on the detection of well-formed synthetic faces.

Monica Chew and J.D. Tygar discuss collaborative filtering CAPTCHAs which do not depend on absolute answers, but are graded by comparison with other people's answers. Tim Converse proposes CAPTCHA generation as a not-for-profit Web service and argues for open-sourcing the code. Daniel Lopresti proposes using instances of open pattern recognition problems to build CAPTCHAs in order to benefit both online security and pattern recognition research.

Jon Bentley and Colin Mallows describe methods for quantifying the assurance that can be inferred from a correct answer to a password query: the principles underlying this analysis are applicable to the evaluation of CAPTCHA security. Rachna Dhamija and J.D. Tygar investigate HIPs in which a user issues challenges to the computer, rather than the other way around, enabling the detection of phishing attacks.

We are warmly grateful for the time and skill volunteered by our Program Committee, as well as to our Advisory Board for kindly assisting in publicizing the event and suggesting ways to make the program more stimulating.

May 2005

Henry S. Baird
Daniel P. Lopresti
Workshop Co-chairs, HIP 2005
Department of Computer Science & Engineering
Lehigh University
Bethlehem, PA, USA

Organization

HIP 2005 was organized by the Department of Computer Science & Engineering, Lehigh University and was endorsed by IAPR, the International Association for Pattern Recognition.

Advisory Board

Jon Bentley	Avaya Labs Research
Manuel Blum	Computer Science, Carnegie-Mellon University
Andrei Broder	IBM Research
Gordon Legge	Psychology Depts., University of Minnesota
Richard Lipton	College of Computing, Georgia Tech
Patrice Simard	Microsoft Research
Doug Tygar	Computer Science & SIMS Depts., UC Berkeley

Program Committee

Luis von Ahn	Computer Science, Carnegie-Mellon University
Kumar Chellapilla	Microsoft Research
Monica Chew	Computer Science, UC Berkeley
James Clark	ECE Dept, McGill University
Brian Davison	Computer Science & Engineering, Lehigh
Irfan Essa	GVU Center, Georgia Tech
Venugopal Govindaraju	CEDAR, SUNY Buffalo
Greg Kochanski	Phonetics Lab, Oxford University
Richard Landsman	America Online
Cem Paya	Microsoft MSN/Passport
David Pletcher	Lawrence Livermore Lab, Berkeley
Chilin Shih	EALC & Linguistics, University of Illinois
Richard Sproat	Linguistics & ECE, University of Illinois

Sponsoring Institutions

We gratefully acknowledge financial support from Microsoft Research and Avaya Labs Research.

Table of Contents

CAPTCHAs and Performance Analysis

HIP Architectures

HIPs Within Security Systems

Building Segmentation Based Human-Friendly Human Interaction Proofs (HIPs)

Kumar Chellapilla, Kevin Larson, Patrice Y. Simard, and Mary Czerwinski

Microsoft Research, One Microsoft Way, Redmond, WA, USA 98052
{kumarc, kevlar, patrice, marycz}@microsoft.com

Abstract. Human interaction proofs (HIPs) have become common place on the internet due to their effectiveness in deterring automated abuse of online services intended for humans. However, there is a co-evolutionary arms race in progress and these proofs are becoming more difficult for genuine users while attackers are getting better at breaking existing HIPs. We studied various popular HIPs on the internet to understand their strength and human friendliness. To determine HIP strength, we adopted a direct approach of building computer attacks using image processing and machine learning techniques. To understand human-friendliness, a sequence of users studies were conducted to investigate HIP character recognition by humans under a variety of visual distortions and clutter commonly employed in reading-based HIPs. We found that many of the online HIPs are pure recognition tasks that can be easily broken using machine learning. The stronger HIPs tend to pose a combination of segmentation and recognition challenges. Further, the HIP user studies show that given correct segmentation, computers are much better at HIP character recognition than humans. In light of these results, we propose that segmentation-based reading challenges are the future for building stronger human-friendly HIPs. An example of such a segmentation-based HIP is presented with a preliminary assessment of its strength and human-friendliness.

1 Introduction

Human Interaction Proofs[1] (HIPs) [3] or Completed Automated Public Turing tests to tell Computers and Humans Apart (CAPTCHAs) [4] are systems that allow a computer to distinguish between another computer and a human. These systems enable the construction of automatic filters that can be used to prevent automated scripts from utilizing services intended for humans [4]. An overview of the work in this area can be found in [3]. Work on building HIPs dates back to 1997 with the first HIP being invented [13] at the DEC Systems Research Center for blocking abusive automatic submission of URLs to the AltaVista web-site (www.altavista.com). Since then numerous HIPs have been proposed and several have been adopted by companies to

[1] These are also referred to as "Human Interactive Proofs." The term "Human Interaction Proof" is preferred in this paper as it is clearer in indicating that these are tests for human interaction.

H.S. Baird and D.P. Lopresti (Eds.): HIP 2005, LNCS 3517, pp. 1-26, 2005.

protect various services on the web. However, the basic challenge still remains the same: design a computer program that can automatically generate and grade tests that most humans can pass but current computer programs cannot pass. For a HIP to be successful in practice, it should also be fast and be capable of generating millions of unique samples a day.

Construction of HIPs of practical value is difficult because it is not sufficient to develop challenges at which humans are somewhat more successful than machines. This is because the cost of failure from using machines to solve the puzzles may be very small. In practice, if one wants to block automated scripts, a challenge at which humans are about 90% successful and machines are 1% successful, may not be sufficient, especially when the cost of failure and repetition is low for the machine [2,7,12]. At the same time, the identical challenge must not put too much burden on the human in order to avoid discouraging the use of the service. This is summarized in Figure 1. The figure shows an ideal distribution of HIPs. The sweet spot, where the HIPs are easy for humans to recognize but difficult for hackers to crack, is not guaranteed to actually exist. Furthermore, automatically generated HIPs, being random in nature, will have a distribution of difficulty, with some particular instances extending beyond the hypothesized sweet spot.

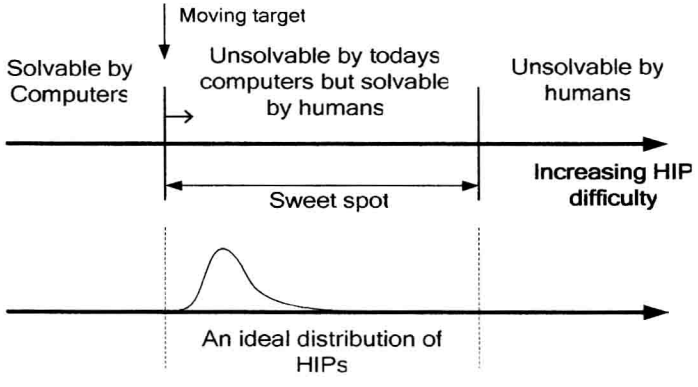

Fig. 1. Regions of feasibility as a function of HIP difficulty for humans and computers algorithms.

Depending on the cost of the attack and the value of the service, automatic scripts should not be more successful than 1 in 10,000 (0.01%). For good usability the human success rate should approach 90%. While the latter is a common requirement for reducing the number of retries a human user has to endure, the former is obtained by analyzing the cost of hiring humans to solve HIPs. For example, requiring a signup HIP for creating an e-mail account only imposes a maximal cost of about .002 cents per message, while the minimum estimate for the costs/potential revenue from sending spam are around .0025 cents, with many spammers charging or earning 5 to 10 times that [12]. Thus, a practical HIP must not only be secure but also be human-friendly. Human-friendliness encompasses both a) the visual appeal and annoyance factor of a HIP, and also b) how well it utilizes the difference in ability between humans and machines at solving segmentation and recognition tasks. While HIP secu-

rity considerations push designers to make the HIP difficult for both humans and computers, the human-friendliness requirements force the designer to make them only as hard as they need to be and still be effective at deterring abuse. Due to this inherent conflict between these two requirements, online HIPs are undergoing an arms race. As computer vision research advances, computers get faster, and attackers get sophisticated, existing HIPs will become ineffective and new HIPs will have to be created. Over time, the sweet spot will decrease in size. Unfortunately, humans are unlikely to get better at solving HIPs in the same timeframe [10,11].

Owing to their advantages, reading-based HIPs have become common place for protecting internet web sites against abuse. Section 2 presents motivations for a reading-based HIP and several examples of common reading-based HIPs that can be sampled on the web. It also presents the segmentation and recognition parts of the HIP challenge and key design choices that go into building a reading-based HIP. Section 3 addresses HIP security from the point of view of a computer attacker attempting to solve a HIP completely (both segmentation and recognition parts being solved) or simply the recognition part of the problem. Section 4 investigates human-friendliness of a HIP by understanding human ability in solving HIP segmentation and recognition. Section 5 reviews lessons learned from computer attacks and user studies and presents segmentation-based HIPs. A preliminary analysis of the security and human-friendliness of an example segmentation-based HIP is also presented.

2 Examples of HIPs

We have come across dozens of proposals for HIP designs, ranging from counting objects in a picture, segmenting faces, recognizing animations, identifying words in audio, etc. [4]. Among visual challenges, Reading-based HIPs are the most obvious favorite [4,5,8,12,13,14]. These HIPs are made of characters rendered to an image and distorted before being presented to the user. Solving the HIP requires identifying all characters in the correct order. Several reasons for their popularity are:

1) optical character recognition (OCR) is a well studied field and the state of the art is well known,
2) characters were designed by humans for humans and humans have been trained at the task since childhood,
3) each character has a corresponding key on the keyboard and 8 keystrokes span a space of over 1000 billion solutions,
4) localization issues are minimal using western characters and numbers (without dictionaries),
5) the task is easily understood by users without much instruction, and
6) character-based HIPs can be generated quickly[2].

Figure 2 presents reading based HIPs that can be sampled from the web while signing up for free e-mail accounts with Mailblocks (www.mailblocks.com), MSN/Hotmail (www.hotmail.com), Yahoo! (www.yahoo.com), Google (gmail.google.com), run-

[2] Over 300 8-character HIPs can be generated per second on a 3GHz P4 [2,12]

ning a whois query at Register.com (www.register.com) or searching for tickets at Ticketmaster (www.ticketmaster.com), etc.

Fig. 2. Example Human Interaction Proofs (HIPs).

Solutions to Yahoo (ver1) HIPs are common English words, but those for ticketmaster and Google do not necessarily belong to the English dictionary. They appear to have been created using a phonetic generator [8]. Examining the changes in MSN, Yahoo!, and Register.com HIPs, we note that these HIPs are becoming progressively more difficult. While MSN introduced more arcs as clutter, Yahoo! gave up their language model and replaced simple textures and grids with more random intersecting lines and arcs. Register.com's update was relatively minor as they simply introduced digits into their character set.

2.1 Segmentation and Recognition Challenges

Reading-based HIP challenges typically comprise a segmentation challenge followed by recognition challenges[3]. Solving the segmentation challenge requires the identification of character locations in the right order. The random location of characters, background textures, foreground and background grids or lines, and clutter in the form of arcs make the segmentation problem difficult. Image warp exacerbates the segmentation problem by reducing the effectiveness of preprocessing stages of a segmentation algorithm that attempt to estimate and remove the background textures and foreground lines, etc. Once character locations are reliably identified (in the right order) each of the characters needs to be recognized correctly giving rise to the recognition problem. The character recognition problem is made difficult through changes in scale, rotation, local and global warp, and intersecting random arcs.

2.2 HIP Design Choices

While the segmentation and recognition challenges provide a conceptual breakdown of the HIP challenge, building an actual reading-based HIP requires one to make several independent choices:

a) **Character set**: The character set to be used in the HIP.

b) **Affine transformations**: Translation, rotation, and scaling of characters

c) **Adversarial clutter**: Random arcs, lines, or other simple geometric shapes that intersect with the characters and themselves

d) **Image warp:** elastic deformations of the HIP Image at different scales i.e., those that stretch and bend the character itself (global warp) and those that simply jiggle the character pixels (local warp)

e) **Background and foreground textures**: These textures are used to form a colored HIP image from a bi-level or grayscale HIP mask generated by using a) through d)

f) **Language model**: the language model determines both the conditional and joint probabilities of character co-occurrence in the HIP. A HIP can use a) no language model (random equally likely occurrence of all possible combinations – Eg. Mailblocks, MSN, Register and Yahoo version 2), b) words from a dictionary (Yahoo! version 1), or c) a phonetic generator [8] (Ticketmaster and Google/Gmail).

Each of these choices affects both HIP security and human-friendliness of the HIP though commonly to different degrees.

[3] Solving a HIP need not require the segmentation and recognition problems to be solved separately.

3 HIP Security

Assessing the strength of a particular HIP is an approximate process at best. The strength of a HIP is determined by the cumulative effects of the HIP design choices. Each choice increases or decreases HIP difficulty and human-friendliness. However, comparing and quantifying contributions from each choice might not be possible as interactions between these choices can be non-linear. Some very general comments can however be made. In general, the larger the character set and the longer the HIP the stronger it is. In the absence of a language model, the strength of the HIP improves exponentially with the length of the HIP and polynomially with the HIP character set size. Affine transformations, clutter, and image warp also increase HIP security through not as dramatically. Background and foreground textures usually provide only a marginal improvement in HIP security. Using only words from a dictionary makes the HIP considerably easier to break. HIPs using phonetic generators also suffer from this drawback but to a lesser extent. The effects of using a dictionary or a phonetic generator are similar to reducing the effective character set size and HIP solution length.

One direct approach to obtaining a quantitative upper bound for HIP security is to build automated HIP breakers and assess their success in solving particular HIPs. This is exactly the approach adopted here to assess HIP security for popular on-line HIPs [2,12].

3.1 Breaking HIPs

Breaking HIPs is not new. Mori and Malik [7] have successfully broken the EZ-Gimpy (92% success) and Gimpy (33% success) HIPs from CMU. Thayananthan et al [15] have also been successful at breaking EZ-Gimpy [4]. Recently Moy et al [16] have broken the purely distortion based HIP gimpy-r [4] with a success rate of 78%. Our approach aims at an automatic process for solving multiple HIPs with minimum human intervention, using machine learning. In this section, our main goal is to learn more about the common strengths and weaknesses of these HIPs rather than to prove that we can break any one HIP in particular with the highest possible success rate. We summarize results for six different HIPs: EZ-Gimpy/Yahoo, Yahoo v2, Mailblocks, Register, Ticketmaster, and Google. Further details on these HIP breakers can be found in [2,12].

To simplify our study, we will not be using language models in our attempt to break HIPs. For example, there are only about 561 words in the EZ-Gimpy dictionary [7], which means that a random guess attack would get a success rate of 1 in 561 (more than enough to break the HIP, i.e., greater than 0.01% success).

Our generic method for breaking all of these HIPs is to build a custom segmentation algorithm (to locate the characters) and then use machine learning for recognition. Surprisingly, segmentation, or finding the characters, is simple for many HIPs, which makes the process of breaking the HIP particularly easy. Gimpy uses a single constant predictable color (black) for letters even though the background color changes. We quickly realized that once the segmentation problem is solved, solving the HIP becomes a pure recognition problem, and it can be solved using machine

learning. Our recognition engine is based on convolutional neural networks [6,9]. It yielded a 0.4% error rate on the MNIST database, uses little memory, and is very fast for recognition. Speed is important for breaking HIPs since it reduces the cost of automatic trials.

For each HIP, we have a segmentation step, followed by a recognition step. It should be stressed that we are not trying to solve every HIP of a given type, i.e., our goal is not 100% success rate, but something efficient that can achieve much better than 0.01%.

In each of the following HIP security experiments, 2500 HIPs were hand labeled and used as follows (a) recognition (1600 for training, 200 for validation, and 200 for testing), and (b) segmentation (500 for testing segmentation). For each of the five HIP types, a convolution neural network was trained and tested on gray level character images centered on the guessed character positions (see below). The convolutional neural network is identical to the one described in [6] and consisted of two layers of convolutional nodes followed by two fully connected layers. The output from each convolutional layer was subsampled by two before being fed to the next layer [6]. The architecture was exactly the same for all experiments in this paper. The trained neural network became the recognizer. Except for converting the original image to gray, **no preprocessing of any kind was used for recognition.**

Mailblocks: To solve the HIP, we select the red channel, binarize and erode it, extract the largest connected components (CCs), and breakup CCs that are too large into two or three adjacent CCs. Further, vertically overlapping half character sized CCs are merged. The resulting rough segmentation works most of the time. One example is presented in Figure 3.

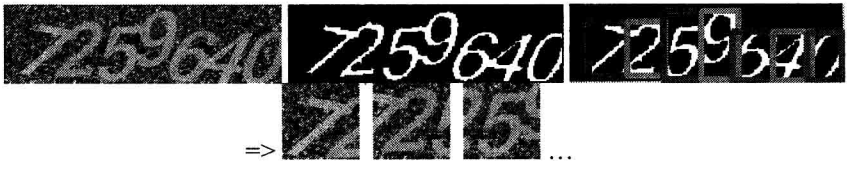

Fig. 3. Breaking Mailblocks HIP.

In the example above, the neural network would be trained, and tested on the segmented chars (right). Each HIP has exactly 7 characters. The segmentation algorithm had a success rate of 88.8% and the neural network recognition rate was 95.9% for recognition (given correct segmentation). The total end-to-end success rate is given by `seg_rate*(reco_rate)^(hip_length)` = $(0.888)*(0.959)^7$ = 66.2% total. Note that most of the errors come from segmentation, even though this is where all the custom programming was invested.

Register: The procedure to solve HIPs is very similar. The image was smoothed, binarized, and the largest 5 connected components were identified. The success rate is 95.4% for segmentation, 87.1% for recognition (given correct segmentation), and $(0.954)*(0.871)^5 = 47.8\%$ total. One example is presented in Figure 4.

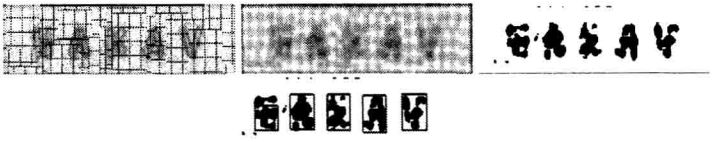

Fig. 4. Breaking Register HIP.

Yahoo!/EZ-Gimpy: Unlike the mailblocks and register HIPs, the Yahoo/EZ-Gimpy HIPs are richer in that a variety of backgrounds and clutter are possible. Though some amount of text warping is present, the text color, size, and font have low variability. Three simple segmentation algorithms were designed with associated rules to identify which algorithm to use. The goal was to keep these simple yet effective:

a) **No mesh**: Convert to grayscale image, threshold to black and white, select large CCs with sizes close to HIP char sizes. Figure 5 shows one example.

Fig. 5. Breaking Yahoo HIP: no mesh case.

b) **Black mesh**: Convert to grayscale image, threshold to black and white, remove vertical and horizontal line pixels that don't have neighboring pixels, select large CCs with sizes close to HIP char sizes. Figure 6 shows one example.

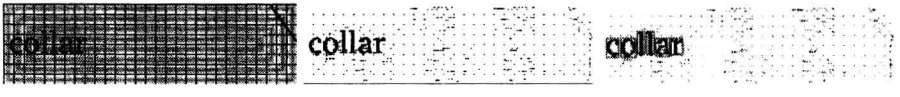

Fig. 6. Breaking Yahoo HIP: black mesh case.

c) **White mesh**: Convert to grayscale image, threshold to black and white, add black pixels (in white line locations) if there exist neighboring pixels, select large CCs with sizes close to HIP char sizes. Figure 7 shows one example.

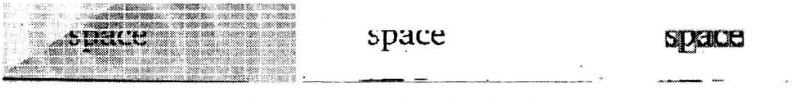

Fig. 7. Breaking Yahoo HIP: black mesh case.

Tests for black and white meshes were performed to determine which segmentation algorithm to use. The average length of a Yahoo HIP solution is 4.8 characters. The end-to-end success rate was 56.2% for segmentation (38.2% came from a), 11.8% from b), and 6.2% from c), 90.3% for recognition (given correct segmenta-

tion), and $(0.562)*(0.903)^{4.8} = 34.4\%$ total. Note that the same recognizer was used for all 3 scenarios.

Ticketmaster: The procedure that solved the Yahoo HIP is fairly successful at solving some of the ticket master HIPs. These HIPs are characterized by cris-crossing lines at random angles clustered around 0, 45, 90, and 135 degrees. A multipronged attack as in the Yahoo case (section 3.3) has potential. In the interests of simplicity, a single attack was developed: Convert to grayscale, threshold to black and white, up-sample image, dilate first then erode, select large CCs with sizes close to HIP char sizes. One example is presented in Figure 8.

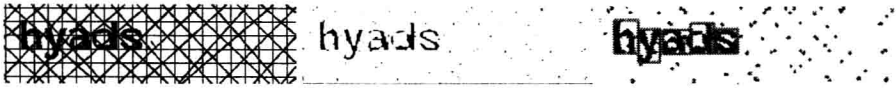

Fig. 8. Breaking Ticketmaster HIP.

The dilate-erode combination causes the lines to be removed (along with any thin objects) but retains solid thick characters. This single attack is successful in achieving an end-to-end success rate of 16.6% for segmentation, the recognition rate was 82.3% (in spite of interfering lines), and $(0.166)*(0.823)^{6.23} = 4.9\%$ total. The average HIP solution length is 6.23 characters.

Yahoo! (version 2): The second generation HIP from Yahoo had several changes: a) it did not use words from a dictionary or even use a phonetic generator, b) it uses only black and white colors, c) uses both letters and digits, and d) uses connected lines and arcs as clutter. The HIP is somewhat similar to the MSN/Passport HIP which does not use a dictionary, uses two colors, uses letters and digits, and background and foreground arcs as clutter. Unlike the MSN/Passport HIP, several different fonts are used. A single segmentation attack was developed: Remove the 6 pixel border, up-sample, dilate first then erode, select large CCs with sizes close to HIP character sizes. The attack is practically identical to that used for the ticketmaster HIP with different preprocessing stages and slightly modified parameters. Figure 9 shows an example.

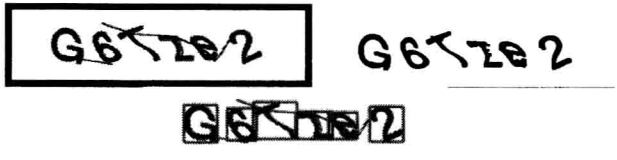

Fig. 9. Breaking Yahoo! (version 2) HIP.

This single attack is successful in achieving an end-to-end success rate of 58.4% for segmentation, the recognition rate was 95.2%, and $(0.584)*(0.952)^5 = 45.7\%$ total. The average HIP solution length is 5 characters.

Google/Gmail: The Google HIP is unique in that it uses only image warp as a means of distorting the characters. Similar to the MSN/Passport and Yahoo version 2 HIPs, it is also two colors. The HIP characters are arranged close to one another (they often touch) and follow a curved baseline. The following very simple attack was used

to segment Google HIPs: Convert to grayscale, up-sample, threshold and separate connected components.

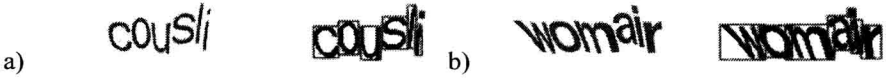

Fig. 10. Breaking Google HIP.

This very simple attack gives an end-to-end success rate of 10.2% for segmentation, the recognition rate was 89.3%, giving $(0.102)*(0.893)^{6.5} = 4.89\%$ total probability of breaking a HIP. Average Google HIP solution length is 6.5 characters. This can be significantly improved upon by judicious use of dilate-erode attack. A direct application doesn't do as well as it did on the ticketmaster and yahoo HIPs (because of the shear and warp of the baseline of the word). More successful and complicated attacks might estimate and counter the shear and warp of the baseline to achieve better success rates.

The above experiments show that using a very simple approach, even the strongest HIPs can be solved more often than 1 in 25 leaving us far from the 1 in 10,000 mark. While recognition success rates ranged from 82.3% (ticketmaster) to 95.9% (mailblocks), segmentation success rates ranged from 10.2% (Google) to 95.4% (Register.com). Clearly, the segmentation problem is crucial in determining HIP security, while recognition of HIPs characters appears to be a solved problem.

3.2 Single Character Recognition Using Machine Learning

Interestingly, the recognition problems posed by the HIPs in Section 3.1 were specifically designed to fool off-the-shelf OCR systems (e.g. Scansoft's OCR and several others [14]). However, as seen in Section 3.1 they can be effectively solved using a neural network that is trained on HIP characters. In this section we explore the abilities of such a neural network [6] for solving single character recognition problems under a variety of distortions and clutter. These distortions are similar to those commonly employed in HIPs and are described below. In this study, to better understand the NN's capabilities, the difficulty of the recognition problem is driven much higher than what would be deemed appropriate for use in a HIP.

3.2.1 Single Character Recognition Using Machine Learning

Character-based HIPs employ a set of character distortions to make them hard to OCR using computers. The basic character transformations include translation, rotation (clockwise or counterclockwise), and scaling. Rotation is usually less than 45 degrees to avoid converting a 6 into a 9, an M into a W etc. Examples of these distortions are presented in figures 11, 12, and 13.