iCAPTCHA: The Next Generation of CAPTCHA Designed to Defend Against 3rd Party Human Attacks

Huy D. Truong, Christopher F. Turner, Cliff C. Zou
University of Central Florida, FL USA

Abstract— **CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) is a simple test that is easy for humans but extremely difficult for computers to solve. CAPTCHA has been widely used in commercial websites such as web-based email providers, TicketMaster, GoDaddy, and Facebook to protect their resources from attacks initiated by automatic scripts. By design, CAPTCHA is unable to distinguish between a human attacker and a legitimate human user. This leaves websites using CAPTCHA vulnerable to 3rd party human CAPTCHA attacks. In order to demonstrate the vulnerabilities in existing CAPTCHA technologies we develop a new streamlined human-based CAPTCHA attack that uses Instant Messenger infrastructure. Facing this serious human-based attack threat, we then present a new defense system called Interactive CAPTCHA (iCAPTCHA), which is the next generation of CAPTCHA technology providing the first steps toward defending against 3rd party human CAPTCHA attacks. iCAPTCHA requires a user to solve a CAPTCHA test via a series of user interactions. The multi-step back-and-forth traffic between client and server amplifies the statistical timing difference between a legitimate user and a human solver, which enables better attack detection performance. A performance and usability study of iCAPTCHA shows the proposed scheme is effective in attack detection, is easy to use, and is a viable replacement of the current text-based CAPTCHA.**

Index Terms — **CAPTCHA, Experimentation, Human Factors, Security**

I. INTRODUCTION

The mechanism for using randomly generated images containing words or characters for human-user validation was developed by Alta Vista in the late 1990's [1]. The term CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) was coined in 2000 by Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford [1]. CAPTCHAs were designed to generate tests that distinguish humans from malicious computer programs. Today, CAPTCHA technology is widely used to defend against scripted registrations in web-based services such as web-based email accounts. Despite their widespread use, CAPTCHAs are not foolproof. CATPCHAs can and have been broken consistently. Besides the primary attack method using image processing to decode CAPTCHA tests, recently techniques have been developed for utilizing a 3rd party human user to break given CAPCHAs. This type of attack is particularly difficult to prevent, and our research shows that no effective solutions currently exist. One of the primary aims of this paper is to present a novel and effective method for coping with this security challenge.

The contributions of this paper include:

- In order to demonstrate the serious vulnerability of existing CAPCHAs against human-based attack, we developed a simple but effective human solver attack system called IMCA (Instant Messenger CAPTCHA Attack) by utilizing the mature Instant Messenger infrastructure for attack communication.
- We proposed a new CAPTCHA implementation (Interactive CAPTCHA or iCAPTCHA) that helps defend against 3rd party human CAPTCHA attacks.
- We conducted a performance and usability study of the proposed iCAPTCHA to evaluate the effectiveness of the approach against 3rd party human CAPTCHA attacks and to ensure ease of use.

The rest of the paper is organized as follows: In Section II, we describe vulnerabilities of existing CAPTCHA technology. Section III explains the design and implementation of IMCA, our streamlined Instant Messenger CAPTCHA attack. Section IV describes the iCAPTCHA design, prototype, and its security features. Section V presents and describes the iCAPTCHA usability study. In Section VI, we discuss the limitations of iCAPTCHA and provide ideas to enhance its performance. Finally, we summarize our findings in Section VII.

II. VULNERABILITIES OF EXISTING CAPTCHA TECHNOLOGY AND THEIR COUNTERMEASURES

There are two types of exploits that have been effective in attacking current CAPTCHA technology.

The dominant type of exploits is to defeat CAPTCHA challenge via image recognition techniques. Using image processing to attack a CAPTCHA is a multi-step process. First, background noise and anomalies are removed from a CAPTCHA image [2, 3]. Next, the image is segmented and passed to character and shape recognition algorithms [3]. Utilizing this methodology, Jitendra Malik and Greg Mori were able to defeat the E-Z Gimpy CAPTCHA with a 92% success rate [2]. Yan and El Ahmad had a success rate of 61% for breaking the MSN scheme [4]. In order to defend against image processing CAPTCHA attacks, modern CAPTCHAs include background noise, juxtaposition [4], and animation [5]

Huy D. Truong is a Graduate Student at University of Central Florida, Orlando, FL 32816 USA (e-mail: htruong@knights.ucf.edu).

Christopher F. Turner was with University of Central Florida, Orlando, FL 32816 USA. He is now with the Lockheed Martin Corporation, Orlando, FL 32825 USA (e-mail: ctuner80@gmail.com).

Cliff C. Zou is with Electrical Engineering and Computer Science Department, University of Central Florida, Orlando, FL 32816 USA, (czou@eecs.ucf.edu).

that are harder to recognize by computer software. Unfortunately, overuse of these countermeasures can sometimes make it too difficult for a human to read the image and therefore limit their practical usage.

The second type of exploits is through 3rd party human attack. An ideal CAPTCHA would be easily solved by a human user, but completely unreadable to computer software. Therefore, using humans is the guaranteed way to break CAPTCHA. The malware industry has found a variety of ways to use free or cheap 3rd party human labor to break CAPTCHAs [10]. An increase in the use of 3rd party human CAPTCHA attacks has been reported by several sources [8, 10]. Both Whale [8] and Acohido [10] highlighted the economy of breaking CAPTCHA in third world countries with cheap labor. In fact, some unethical businesses are profiting by providing 3rd party human CAPTCHA solving as a service [6, 7, 8, 10]. In India, a CAPTCHA solving economy has developed in which people work as human CAPTCHA solvers, earning $2 for every thousand CAPTCHAs solved [7]. Whale [8] went as far as declaring that "the CAPTCHA approach is doomed" because of the 3rd party human solver vulnerability. In the next section, our proposed streamlined 3rd party human solver attack system, IMCA, clearly illustrates that such attacks are a real threat.

III. IMCA (Instant Messenger CAPTCHA Attack)

In order to illustrate the real threat imposed by human solver attacks, we developed a more efficient 3rd party human CAPTCHA attack system that takes advantage of Instant Messenger network and server infrastructure. We call this system IMCA, or Instant Messenger CAPTCHA Attack. In current CAPTCHA implementations, timeout values for solving CAPTCHA are utilized in hope of detecting a 3rd party human attack. The logic here is that the time between the initial display of a CAPTCHA image to the time a 3rd party human solution is delivered will exceed the timeout and therefore be detected. This is where Instant Messengers come in. Instant Messenger (IM) allows real-time communication between two or more participants over a network. Real-time data exchange is the key for success in the human based CAPTCHA solver scheme. IMCA's use of IM technology allows CAPTCHA images to be delivered to 3rd party human solvers at speeds which defy detection by CAPTCHA timeout values.

A. IMCA Architecture

Figure 1 depicts the architecture of the IMCA system. IMCA contains two major components. The first component is an "Attack Script" that has been custom made to attack a particular website. This script would contain the ability to populate form data, scrape CAPTCHA images, and post web replies. The second component is an "IM Connector" that connects to an Instant Messenger provider to send and receive instant messages. These messages would contain CAPTCHA images obtained from the "Attack Script" that would be sent to the 3rd party human solvers. Multiple "attack scripts" can be used with a single "IM Connector" and multiple solvers in order to improve overall throughput.

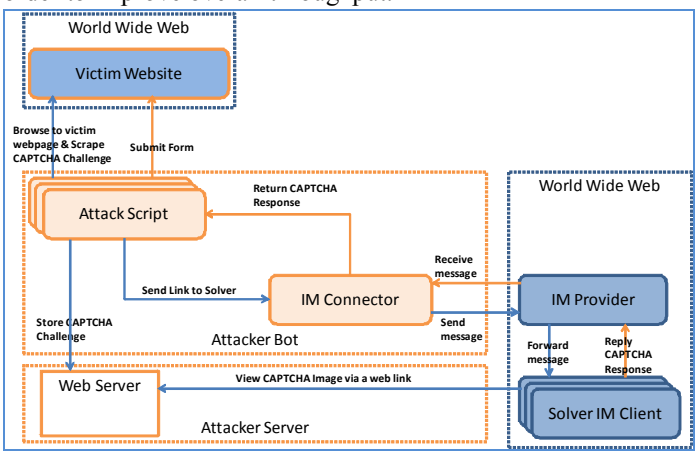

Figure 1: IMCA Architecture

B. IMCA Prototype and Attack Example

We developed an IMCA prototype using a Perl script (Attack Script) and C# application (IM Connector) to attack a CAPTCHA protected website. The prototype exploits AOL Instant Messenger (AIM) so that the 3rd party human solver receives and sends messages using standard AIM client software. We also developed a fictional Wii Tennis Club website with a CAPTCHA protected member submission form as an attack venue. The webpage was developed using .NET technology, and the CAPTCHA implementation is "CAPTCHA Server Control for ASP .NET" by wumpus1, a member of CodeProject [9]. Like most CAPTCHAs, this implementation utilizes a timeout feature to mitigate 3rd party human based attacks. Figures 2 through 4 depict the IMCA in action.

Figure 2 User Entering Data to Join Wii Tennis Club

Figure 2 shows the normal operation of the Wii Tennis Club website. A human joined the club by providing name, email, and a response to the CAPTCHA text. For our example, this page was accessed by running the "Attack Script" (attacker.pl). The "Attack Script" scraped the CAPTCHA image from the page and then utilized the "IM Connector" to send the CAPTCHA to a 3rd party human solver via AIM.

Figure 3: human solver's IM Client view

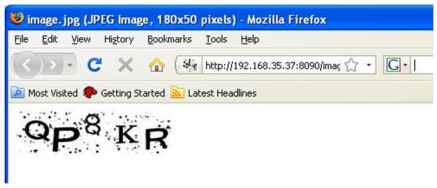

Figure 4: CAPTCHA provided to human solver by IMCA

Figures 3 and 4 show that the 3rd party human solver received a link to the delivered image, viewed the image, and responded with the CAPTCHA text. The "Attack Script" then received the reply. This reply, along with randomly generated form data, was used to generate a valid request to the website.

By exploiting the Instant Messenger infrastructure, IMCA has shown that CAPTCHAs can be delivered to 3rd party human solvers quickly and easily. IMCA is simple and light weight. Using the built-in Perl http library, our example Perl "Attack Script" was less than 50 lines of code. The "IM Connector" was developed from an AIM custom client example that is available from AIM Developer site (http://dev.aol.com/). Despite its simplicity, the IMCA proves an important point: the threat of 3rd party human solver attacks is real! Since IMCA delivers CAPTCHA images to 3rd party human solvers so efficiently, it renders the timeout values of existing CAPTCHA implementations useless; hence the need for a new design iCAPTCHA, the next generation CAPTCHA implementation, which will be introduced next.

IV. iCAPTCHA – Interactive CAPTCHA

Facing with the growing threat of 3rd party human attacks on existing CAPTCHA technologies, the industry is in need of a reliable defense technique. This section describes our proposed iCAPTCHA as the first and initial step towards defending against this type of attack.

A. *iCAPTCHA Concept and Implementation*

iCAPTCHA utilizes a sequence of mouse clicks to allow a user to interactively solve a CAPTCHA challenge. First, a normal CAPTCHA image is dynamically generated and displayed as shown in Figure 5a. The user clicks on the CAPTCHA image to begin the iCAPTCHA input sequence. Upon clicking on the CAPTCHA image, several buttons with obfuscated characters appear below the CAPTCHA image as shown in Figure 5b. This update is performed via an asynchronous JavaScript (Ajax) [11] request to the server that is rendered back to the user's web browser without refreshing the whole web page. Once the set of character buttons is displayed, the user must click on the button corresponding to the first character in the CAPTCHA image. Upon each click, a new set of buttons is rendered. This input sequence continues until one click has been performed for each character of the CAPTCHA image.

Figure 5: Operation of iCAPTCHA. (a) Initial display of iCAPTCHA (b) Display after test begins (c) Display after first input is processed

On the server side, session information is stored about the indices of the correct responses and the indices of the user clicks. When the input sequence is complete, the correct index sequence is then compared with the user clicked index sequence. If there is a match, the CAPTCHA has been correctly decoded by the user.

B. *iCAPTCHA Security Features*

As previously described, most current CAPTCHA implementations utilize a timeout value as a defense against 3rd party human solvers. The problem with this technique, which was made clear by IMCA, is that the time required to deliver a CAPTCHA image to a human solver is too small relative to the timeout value. Therefore there is not enough resolution to detect if the CAPTCHA response was from a legitimate user or an attacker.

To solve this problem, iCAPTCHA measures the time it takes for a user to respond on a per-character basis. This allows the timeout value for iCAPTCHA to be enforced for each character input. Therefore, the per-character timeout for iCAPTCHA can be set much lower than the timeout value for a standard CAPTCHA. This provides a much greater resolution in determining human attacks because the relative time between each input and the time it takes to send the CAPTCHA to a human solver is minuscule. Additionally, as shown in Figure 5a, iCAPTCHA allows users to take as much time as needed to decode the image first before starting the multi-step challenge/response sequence. Due to this separation of decoding and actions, subsequent interactive steps are expected to be swift.

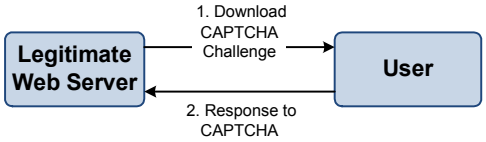

Figure 6: Legitimate user's interaction with iCAPTCHA

As shown in Figure 6, the per-character-response time for a legitimate user interaction with iCAPTCHA is calculated as shown below:

$$R_u = t_1 + t_2 + U \quad (1)$$

Where:

- R_u: the total response time for a single character in the legitimate user scenario
- t_1: network time to download and view the CAPTCHA and obfuscated characters
- t_2: network time to submit HTML post to the web server

- U: time for user to decode and click on the corresponding character

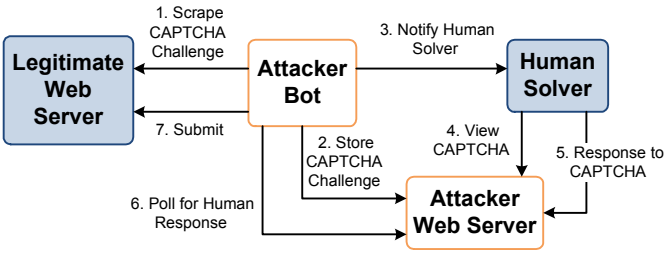

Figure 7: Attack-Bot's interaction with iCAPTCHA

Figure 7 depicts the complicated process of forwarding iCAPTCHA to a 3rd party human solver. In the first iteration, the attack script notifies the human solver of the iCAPTCHA location, which is served on an attacker's web server. For subsequent iterations, the Attacker-Bot proxies the challenge/response inputs back and forth with the target website. The human solver interacts with the CAPTCHA challenge as if he were visiting the victim website. In step 5, the human solver has to respond to the iCAPTCHA by clicking on the corresponding character button. At this point, the attacker's web server has the response information, but it cannot immediately deliver it to the attacker bot. The attacking script has to poll for the result. Depending on the polling interval and resource available, step 6 adds additional delay time to the process. It should be noted that each set of buttons is only sent out by the server after the server receives the click response for the previous set of buttons. Therefore, an attacker bot has to relay the sequence of challenges and responses between the target server and the 3rd party human solver. The replay process takes additional times beyond the network delay between attacker bot and the 3rd party human solver. The total per-character-response time for an attacker can be expressed as follow:

$$R_a = t_1 + t_2 + t_4 + t_5 + t_6 + t_7 + U \quad (2)$$

Where:

- R_a: the total response time for a single character in the 3rd party human attacker user scenario
- t_1: network time to download and scrape the CAPTCHA and obfuscated characters
- t_2: network time to upload or ftp CAPTCHA image and obfuscated characters to attacker's web server
- t_4: delay to download and view the CAPTCHA and obfuscated characters
- t_5: network time to submit HTML post to the web server
- t_6: the polling time for response
- t_7: network time to submit HTML post to the web server
- U: time for human user to decode and click on the corresponding character.

By comparing Equations (1) and (2), it is clear that a 3rd party human solver attack adds 4 additional time delays in solving a single CAPTCHA character in the proposed iCAPTCHA implementation. Since a typical R_u (legitimate user per-character response time) is on the order of 1-2 seconds, the added delay time in an attack scenario would be significant. This additional delay time allows iCAPTCHA to be effective in detecting and stopping 3rd party human attacks.

C. *iCAPTCHA Attack Detection Algorithms*

By recording CAPTCHA solving time on a per-character basis, we developed the following algorithms for iCAPTCHA that enable it to detect and reject 3rd party human attacks in ways not possible with existing CAPTCHAs.

Algorithm 1 – Single Slow Response Detection Algorithm: in our current iCAPTCHA implementation, if a test image text has n letters, a user will click n times and produce n per-character-response times. The first detection algorithm rejects any CAPTCHA test responses that have one or more per-character response times greater than a predefined threshold D.

Algorithm 2 – Two Consecutive Slow Responses Detection Algorithm: for a single user, he or she should have a similar decode time U to decode and click on each of the corresponding characters of an iCAPTCHA. However, as shown in Equation (1), the per-character response time is determined by both the user decode time and the Round Trip Time (RTT) between the web server and the user's web browser. If the network between a server and a client is not reliable or has very dynamic bandwidth, occasionally a user's per-character response time may be significantly higher due to network jitter, and hence, may cause a false positive detection in Algorithm 1.

Network jitter happens from time to time, but the per-character response times from an attack are consistently higher than normal. Taking advantage of this unique feature, we can improve the detection algorithm as follows: we only reject an iCAPTCHA test that has 2 consecutive per-character response times above the threshold D. This second detection algorithm significantly reduces the false positive rate, as confirmed by our performance study.

Dynamic Detection Threshold Algorithm: If a legitimate user has a high-latency connection, like a dial-up modem, the per-character response time may be significantly larger than a predetermined threshold D. We have accounted for this by utilizing a dynamic threshold D which adapts to an individual user's network latency. To achieve this, we detect the Round-Trip Time (RTT) between the server and a remote user based on the ongoing HTTP connection, and then set the detection threshold D for this client as:

$$D = RTT + U_{avg} \quad (3)$$

Where:

- U_{avg} is the average baseline time for a human user to decode and click on the corresponding character. The value of U_{avg} can be obtained based on observation of past client responses.
- RTT can be calculated by inserting an automatic round trip in the Ajax code as shown in Figure 8. After the user clicks on a button to reply to the first character, the client side Ajax script sends a request to the server for a new set of buttons for the second character. At this moment, the server responds with a ping command and the client side

code responds with a pong. This interaction allows the server to calculate *RTT*.

The dynamic rejection threshold concept can be applied to both algorithm 1 and algorithm 2 to further reduce their false positive rate.

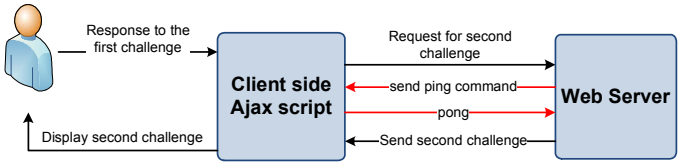

Figure 8: RTT (Round Trip Time) Calculation Method

V. iCAPTCHA Performance and Usability

A. Study Setup

We allowed real users to access our iCAPTCHA test server to help us determine if iCAPTCHA is a feasible replacement for existing CAPTCHA technology. The server utilized for the study can be accessed here: http://cns.eecs.ucf.edu/icaptcha/. Participants in the study were asked to try iCAPTCHA five times followed by two traditional CAPTCHAs that use the same image obfuscation style as iCAPTCHA. At the end, we asked the user to complete a survey about their experience. Participants were recruited via a Facebook shout and 63 random users participated in the study. The users' locations were spread all over the United States, and they also used a mix of different web browsers to access the study website. We were able to collect timing data for 226 iCAPTCHA tests. In the sample implementation, each iCAPTCHA has five characters; therefore, we have 1130 sample per-character response times for legitimate users (R_u).

Additionally, we set up a human-based attack as shown in Figure 7, which also solved 226 iCAPTCHA tests and generated 1130 per-character response time for human solver attacks (R_a). The 3rd party human solvers were two highly experienced CAPTCHA solvers that utilized Mozilla Firefox and a broadband connection.

B. Performance Results and Observations

Table 2 shows that the R_a average time of 5.05 seconds is significant larger than the R_u average time of 1.62 seconds. The results confirmed our hypothesis that we can rely on timing differences to detect human solver attacks. Figure 9 shows the distribution of R_u and R_a. Based on this figure we selected 3.35 seconds, where the two histogram lines crossed, as the detection threshold D for the per-character response.

Table 1: iCAPTCHA Performance Test Result (seconds)

	Legitimate User Time (R_u)	**Human Solver (R_a)**
Average	1.62	5.05
Std. Dev.	1.2329	1.4730
Sample Size	1130	1130

For the first detection algorithm introduced in section IV.C, all of the 226 human solver responses have at least one interaction above the threshold D=3.35 seconds, so this algorithm detected all human solver attacks, which means the algorithm had a detection rate of 100% for our test dataset. However, the algorithm also rejected 23 out of 226 valid user responses. This yields a marginal 10.17% false positive error rate as shown in Table 2.

The second detection algorithm described in section IV.C utilizes two consecutive slow responses. Since most human solver response times are consistently above the threshold of 3.35 seconds, this algorithm also detected all human solver attacks and provided a 0.0% false negative error rate. The real benefit of this algorithm is in its low false positive rate. As shown in Table 2, this algorithm only had a 1.77% false positive error rate, i.e., the algorithm rejected only 4 correct responses out of 226 iCAPTCHA tests from legitimate users. From this detection performance result, we believe that the proposed iCAPTCHA is effective in defending against human solver attacks.

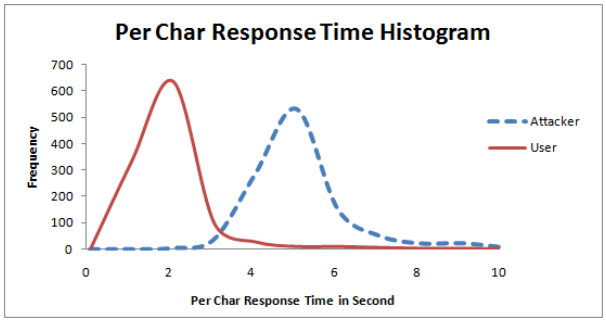

Figure 9: Per-Character Response Time Histogram (seconds)

Table 2: Comparison of Two Detection Algorithms

	Algorithm 1	**Algorithm 2**
False Negative	0.0%	0.0%
False Positive	10.17%	1.77%

The dynamic threshold algorithm was designed and implemented after the user study was conducted. Therefore, we do not have performance data for the dynamic threshold algorithm in this study but we do believe that it would be useful improving iCAPTCHA performance.

C. Usability Study Results

As part of the participant survey, some demographic information was collected. Among the 63 participants, most of them were male in the age group of 21 to 41 with computer and Internet skills. Table 3 shows that 82% of the participants answered that iCAPTCHA is easier to use or about the same as traditional CAPTCHA. This result confirms that iCAPTCHA is intuitive and does not require a steep learning curve for ordinary users. In addition to security, ease-of-use is critical for iCAPTCHA to be a successful and practical CAPTCHA replacement so we were pleased with these results.

Table 3: Ease of Use

iCAPTCHA is harder to use	18%
The same	44%
iCAPTCHA is easier to use	38%

Table 4: Response Speed

iCAPTCHA is slower	44%
The same	22%
iCAPTCHA is faster to use	34%

We anticipated that the total response time for an iCAPTCHA would be slower than traditional CAPTCHA. The collected timing data shows that on average a user takes 8.08 seconds to solve a five character iCAPTCHA versus 6.21 seconds for a traditional CAPTCHA using the same image obfuscation style. However, Table 4 shows that only 44% of participants thought iCAPTCHA was slower. The user's perception that iCAPTCHA is faster, despite its falsehood, is a plus for iCAPTCHA usability. Finally, nearly half of users preferred the use of the mouse to respond to CAPTCHA challenges.

VI. Limitations and Future Works

The iCAPTCHA prototype was implemented as a proof-of-concept and it does have some limitations:

1. *Users with impaired vision or motor skills:* these legitimate users will likely have a slow response time U, and iCAPTCHA may reject their responses. Further evaluation is needed to accommodate impaired users.
2. *iCAPTCHA performance against character recognition based attacks:* in order to break the current text-based CAPTCHA, an attacker algorithm needs to segment out each character of an image and identify it. iCAPTCHA may provide the attacker's algorithm some hints about each letter since it can match each character against a set of obfuscated candidate characters. The CAPTCHA image could be more distorted. iCAPTCHA can be integrated with any current text-based CAPTCHA with feature such as juxtaposition [3]. In addition, the candidate button images must be dynamically generated. The prototype used a static set of distorted buttons.

VII. Conclusion

CAPTCHA plays an important role in protecting Internet resources from attacks by automated scripts. However, CAPTCHA is believed to be vulnerable to 3rd party human attacks due to the nature of its design. We developed a streamlined 3rd party human CAPTCHA attack that uses Instant Messenger infrastructure to demonstrate how easily the current CAPTCHA implementation can be compromised by 3rd party human attacks. We then proposed the novel iCAPTCHA system which provides simple yet effective defense against 3rd party human solver attacks. The multi-step back-and-forth traffic between client and server amplifies the statistical timing difference between a legitimate user and a human solver attack, and hence, provides a better attack detection performance. As the first step towards defending against the growing threat of 3rd party human CAPTCHA attacks, we hope that the proposed iCAPTCHA system will encourage researchers and the security industry to develop more secure and reliable CAPTCHAs.

VIII. Acknowledgment

We thank the anonymous reviewers for their valuable comments and suggestions. This research work is conducted based on the course term project "The War against CAPTCHA: With implementation of the world's most accurate CAPTCHA Breaker" by Huy Truong and Kathleen Stoeckle in Spring 2009 at University of Central Florida.

IX. References

[1] Luis von Ahn , Manuel Blum , John Langford, Telling humans and computers apart automatically, Communications of the ACM, v.47 n.2, p.56-60, February 2004 [doi>10.1145/966389.966390]

[2] G. Mori and J. Malik. Recognizing objects in adversarial clutter – breaking a visual CAPTCHA. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, Jun. 18-20, 2003.

[3] G. Moy, N. Jones, C. Harkless, and R. Potter, "Distortion Estimation Techniques in Solving Visual CAPTCHAs", Proc. IEEE CVPR, 2004

[4] Jeff Yan, Ahmad Salah El Ahmad, A Low-Cost Attack on a Microsoft CAPTCHA, Proceedings of the 15th ACM conference on Computer and communications security, 2008.

[5] Elias Athanasopoulos, Spyros Antonatos: Enhanced CAPTCHAs: Using Animation to Tell Humans and Computers Apart. Communications and Multimedia Security 2006: 97-108

[6] Sam Hocevar, "PWNtcha – a Captcha Decoder Website", http://caca.zoy.org/wiki/PWNtcha

[7] Dancho Danchev, "Inside India's CAPTCHA Solving Economy", http://blogs.zdnet.com/security/?p=1835, 2008

[8] Albert E. Whale, "ABS computer technology, inc, Why the CAPTCHA Approach is Doomed", http://www.abs-comptech.com/home/headlines/news/why-the-CAPTCHA-approach-is-doomed , 2009

[9] Wumpus1, A CAPTCHA Server Control for ASP.NET, http://www.codeproject.com/KB/custom-controls/CaptchaControl.aspx

[10] Byron Acohido, "Cybergangs use cheap labor to break codes on social sites", http://www.usatoday.com/tech/news/computersecurity/2009-04-22-captcha-code-breakers_N.htm, 20092003

[11] J.J. Garrett, "Ajax: A New Approach to Web Applications", Adaptive Path Assays, 2005. http://www.adaptivepath.com/ideas/essays/archives/000385.php