Breaking e-Banking CAPTCHAs

Shujun Li
University of Konstanz, Germany

S. Amier Haider Shah
National University of Science and Technology, Pakistan

M. Asad Usman Khan
National University of Science and Technology, Pakistan

Syed Ali Khayam
National University of Science and Technology, Pakistan

Ahmad-Reza Sadeghi
Ruhr-University of Bochum, Germany

Roland Schmitz
Stuttgart Media University, Germany

ABSTRACT

Many financial institutions have deployed CAPTCHAs to protect their services (e.g., e-banking) from automated attacks. In addition to CAPTCHAs for login, CAPTCHAs are also used to prevent malicious manipulation of e-banking transactions by automated Man-in-the-Middle (MitM) attackers. Despite serious financial risks, security of e-banking CAPTCHAs is largely unexplored. In this paper, we report the first comprehensive study on e-banking CAPTCHAs deployed around the world. A new set of image processing and pattern recognition techniques is proposed to break *all* e-banking CAPTCHA schemes that we found over the Internet, including three e-banking CAPTCHA schemes for transaction verification and 41 schemes for login. These broken e-banking CAPTCHA schemes are used by thousands of financial institutions worldwide, which are serving hundreds of millions of e-banking customers. The success rate of our proposed attacks are either equal to or close to 100%. We also discuss possible improvements to these e-banking CAPTCHA schemes and show essential difficulties of designing e-banking CAPTCHAs that are both secure and usable.

Categories and Subject Descriptors

K.6.5 [**Computing Milieux**]: Management of Computing and Information Systems—*Security and Protection*

General Terms

Security, Electronic Commerce, Human Factors

Keywords

CAPTCHA, e-banking, man-in-the-middle attack, malware

1. INTRODUCTION

Due to their ease and ubiquity of use, e-banking systems have experienced worldwide deployments. A 2009 survey of the American Bankers Association reveals that e-banking has been the preferred banking method of bank customers [1]. Security of e-banking systems is a major concern for the financial institutions and their customers. The highly sensitive nature of data processed by e-banking systems necessitates a robust security framework to protect the users' privacy, identity and assets.

Many financial institutions around the world have deployed CAPTCHAs[1] to protect their e-banking systems from automated attacks. In addition to traditional CAPTCHAs for preventing automated login attempts, some financial institutions have also deployed CAPTCHAs for transaction verification. The main goal of these CAPTCHAs is to make automated transaction manipulation by malicious programs (e.g., Trojans) more difficult. These CAPTCHAs are supposed to provide security against Man-in-the-Middle (MitM) attacks, which can manipulate the communication between the user and the e-banking server on the fly. Such attacks are much more difficult to detect than credential stealers (like email-based phishing attacks and keyloggers) because they can circumvent many existing e-banking protection mechanisms including multi-factor authentication schemes. While the number of such attacks remains unknown, large-scale attacks are becoming more and more likely with the rising infection rate and evolving sophistication of malware on desktop PCs and smart phones.

Existing e-banking solutions counter MitM attacks by introducing some means of transaction verification into the system. These solutions can be broadly divided into the following classes: trusted out-of-band channel [2]; hardware for establishing encrypted channel over untrusted network and/or computer [3]; hardware with trusted display/keypad for generating transaction-dependent TANs or signatures [4, 5]; and solutions based on CAPTCHAs (see Sec. 2.2). The main advantage of CAPTCHA-based solutions is that they do not depend on special hardware and therefore the implementation and maintenance costs are very low for both financial institutions and customers.

The main premise of e-banking CAPTCHAs for both login and transaction is that some pattern recognition tasks are extremely difficult for computers (e.g., automated programs like malware) but easy for humans. Based on this premise, e-banking systems protected by CAPTCHAs are considered secure against automated attackers which aim to interpret

[1]CAPTCHA is an acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart".

ACSAC '10 Dec. 6-10, 2010, Austin, Texas USA

or forge CAPTCHAs. A further challenge of breaking transaction e-banking CAPTCHAs is that the automated attack needs to run in real time to avoid being noticed by users.

A number of prior efforts have been made for analyzing the security of general-purpose CAPTCHAs (which are only for login). However, to the best of our knowledge, the security of e-banking CAPTCHAs has not yet been evaluated thoroughly.

Contribution: This paper presents the first comprehensive study on e-banking CAPTCHAs, and shows that existing e-banking CAPTCHAs do not meet the expected security requirements. More precisely, we report practical attacks on three e-banking CAPTCHA schemes for transaction verification and 41 schemes for login.[2] Our attacks are based on a new set of image processing and pattern recognition tools, including k-means clustering [6], digital image inpainting [7,8], morphological image processing [9], character recognition based on cross-correlation [10] and an image quality assessment (IQA) method called CW-SSIM [11]. As far as we know, it is the first time that image inpainting and IQA algorithms are used to break CAPTCHAs. Our attacks are alarmingly successful: all of the e-banking CAPTCHA schemes are broken with a success rate equal to or close to 100%. Most of our proposed attacks can run in real time.

Our further investigation shows that it is nontrivial to enhance the security of the broken e-banking CAPTCHAs. CAPTCHAs have some essential drawbacks rooted in their design principle that makes it difficult to simultaneously achieve both usability and security. We thus call for cautions in deploying e-banking CAPTCHAs.

Outline: The rest of this paper is organized as follows. In the next section, we give a survey of related work on the cryptanalysis of CAPTCHAs and the use of CAPTCHAs in e-banking systems. In Sec. 3 we introduce the new set of CAPTCHA-breaking tools used in our attacks. Section 4 demonstrates how these tools are used to break a typical e-banking CAPTCHA scheme for transaction verification (used by around 800 German banks). Then, Section 5 reports our attacks on another two CAPTCHA schemes for transaction verification, which are deployed by two major banks in China. Section 6 shows that 41 e-banking login CAPTCHA schemes deployed by many financial institutions all over the world cannot resist automated segmentation attacks. Based on the proposed attacks, in Sec. 7 we outline some possible improvements to the broken e-banking CAPTCHA schemes, and discuss whether CAPTCHAs are at all appropriate for protecting e-banking system. The last section summarizes the salient findings of our work.

2. RELATED WORK

2.1 CAPTCHAs in general

A CAPTCHA [12] scheme is a challenge-response authentication protocol based on a hard AI problem, which can be easily solved by humans but not by machines. Here, the goal differs from traditional user authentication protocols: to accept humans but reject automated programs. CAPTCHAs can be designed in many forms. The most well-known and widely-deployed form is distorted texts shown as CAPTCHA images. The distortions are chosen in a way that automated programs cannot achieve a comparable recognition rate to what humans can. Figure 1 shows some CAPTCHAs of this kind. Similarly, audio CAPTCHAs designed for the blind use distorted audio as the challenge shown to the prover.

Figure 1: Three CAPTCHAs based on distorted texts (left to right): Google, Microsoft, Yahoo!

Another form of CAPTCHA is based on hard AI problems on image understanding. A typical CAPTCHA of this kind is Asirra [13], which challenges the prover to select all cat pictures from 12 pictures of cats and dogs. The idea of image-based CAPTCHAs has also been generalized to be based on animation, video, and 3-D models. Readers are referred to [14] for a recent survey on CAPTCHAs.

The idea of breaking CAPTCHAs has been around for a while. The first public report we know appeared in 2003 [15], in which Mori and Malik proposed recognition based attacks on Gimpy and EZ-Gimpy, two early CAPTCHA schemes based on distorted texts. Later, several other attacks were reported, showing insecurity of more CAPTCHA schemes based on distorted texts [16, 17]. Moreover, Hocevar demonstrated results of his attacks on quite a number of CAPTCHA schemes on his web site [18], which reveals some common pitfalls of weak CAPTCHAs. In [19], Chellapilla et al. reported an interesting finding: once a CAPTCHA image based on distorted texts is well segmented, automated programs can recognize those single characters even better than humans. This implies that making segmentation harder is the main way to enhance security of CAPTCHAs based on distorted texts. In [20], Yan and Ahmad showed that a simple pixel-count based attack can break a number of CAPTCHAs offered at Captchaservice.org and deployed by some other web sites. In [21], Yan and Ahmad proposed a new attack on some distorted texts based CAPTCHAs, which can be used to segment CAPTCHA images into single characters with high accuracy.

Most of the existing attacks are designed for CAPTCHA schemes that use distorted texts. There are also some attacks on other kinds of CAPTCHA schemes. In [22], Golle showed that a machine learning based attack can achieve a success rate of 10.3% for a 12-image challenge of the image-based CAPTCHA scheme Asirra. In [23,24], attacks to some deployed audio CAPTCHA schemes were reported.

There are also attacks exploiting implementation flaws. In [25], Hernandez-Castro and Ribagorda proposed a side-channel attack on a CAPTCHA scheme based on solving mathematical problems. In [26], Hindle et al. showed that reverse engineering can help to design new attacks. Recently, Hernandez-Castro and Ribagorda pointed out some common problems of many CAPTCHA schemes [27].

2.2 CAPTCHAs for e-banking

One of the main applications of CAPTCHAs is to prevent automated online password attacks [28]. Therefore, many financial institutions have deployed CAPTCHAs on the login pages of their e-banking systems to protect their customers from such attacks. In addition to login CAPTCHAs, many financial institutions have also deployed CAPTCHAs

[2]These are *all* the e-banking CAPTCHA schemes that we had found when we submitted the final edition of this paper.

for transaction verification, in order to prevent automated MitM attacks. The CAPTCHA-based transaction verification works as follows. After receiving a transaction request, the server generates a CAPTCHA image by embedding the transaction data, a dynamic TAN (Transaction Authentication Number) and probably some other information, which is sent to the user for confirming the transaction. In case an automated MitM attacker cannot recognize the textual information embedded in the CAPTCHA image, it will be unable to forge a CAPTCHA image. Although the security of transaction CAPTCHAs depend on the same principle of login CAPTCHAs, there are some essential differences between transaction CAPTCHAs and login CAPTCHAs: 1) a transaction CAPTCHA image generally contains much more characters than a login CAPTCHA image; 2) some (often most) characters in a transaction CAPTCHA image are known to the MitM attacker; 3) forging CAPTCHA images can also break transaction CAPTCHAs. While there has been a large body of previous work on breaking login CAPTCHAs, transaction CAPTCHAs are unique for e-banking and security analysis of them remains unexplored.

We did not find a comprehensive report on e-banking CAPTCHAs deployed by the worldwide banking sector, so we manually checked web sites of many financial institutions. We found e-banking CAPTCHA schemes deployed by a large number of financial institutions in different countries such as China, Germany, USA, Australia and Switzerland.

Deployment of e-banking CAPTCHAs in China is so popular that it has become a standard components of almost every e-banking system. We checked 30 major Chinese banks, among which almost all have deployed login CAPTCHAs, and at least two have deployed transaction CAPTCHAs. In Germany, the pattern is a bit different: many banks have deployed login CAPTCHAs and some have also deployed transaction CAPTCHAs. A similar pattern is observed for the banking industry in the USA: a major e-banking solution provider serving several thousand financial institutions has developed several different login CAPTCHA schemes.

In addition to China, Germany and USA, we also found e-banking CAPTCHAs deployed by financial institutions in other countries. These include a major bank in Switzerland (with branches in many other countries in Europe, Asia, North America and Africa), which has deployed login CAPTCHAs in its e-banking system. A Pakistani bank is also using this Swiss bank's system. Similarly, a private bank based in Latin America has also deployed login CAPTCHAs in its e-banking system, which serves its customers in Latin America, Europe, Asia, Australasia and Africa. Some Australian credit unions are also using login CAPTCHAs.

As a whole, we have found three e-banking CAPTCHA schemes for transaction verifications, one deployed by many German banks and the other two by two major Chinese banks. We found 41 e-banking CAPTCHA schemes for login. These e-banking CAPTCHA schemes involve hundreds of millions e-banking customers all around the world. In this paper, we report our successful attacks on all of these e-banking CAPTCHA schemes. We sanitized the paper to anonymize the names of all affected financial institutions and e-banking security service providers to give them the chance to amend their systems and to avoid our research results being abused by criminals. To this end, we use pseudonyms of the three e-banking CAPTCHA schemes for transaction verification: GeCapatcha refers to the e-banking CAPTCHA scheme used by German banks, ChCaptcha1 and ChCaptcha2 refer to the two used by Chinese banks.

There are also some research proposals about applications of CAPTCHAs for e-banking. In [29], Mitchell discussed the possibility of applying CAPTCHAs to e-commerce environment, where the traditional "security codes" (i.e., TANs) can be replaced by CAPTCHAs to resist automated attacks. In [30], Fischer and Herfet proposed to use CAPTCHAs for e-banking transaction verification. In [31], Szydlowski et al. proposed a CAPTCHA-based software keypad for securing web input of online transactions. In [32], a combination of CAPTCHAs and hardware security tokens is proposed to enhance e-banking security. Security and usability of these proposals remains a topic for further research.

Security analysis of e-banking CAPTCHAs is either largely unexplored or kept confidential. There are very few public reports on e-banking CAPTCHAs available. In [33], Wieser described an attack on a login CAPTCHA scheme deployed by a German bank. This attack depends on a design flaw, which has been fixed in the current deployed system.

3. CAPTCHA-BREAKING TOOLS

Despite the diversity of the e-banking CAPTCHA schemes under study, we managed to find a new set of image processing and pattern recognition tools that can break all the e-banking CAPTCHA schemes with very high success rate. Some of the tools (such as k-means clustering and morphological operations) have been widely used in the field or reported by other researchers, however, two basic tools – digital image inpainting and CW-SSIM based pattern recognition – are introduced for the first time in this paper. In the following, we briefly describe these tools, and discuss implementation issues that are common for all our attacks.

k-means layer segmentation: The first step of any attack on a CAPTCHA scheme is to extract targeted objects from the CAPTCHA image. This normally requires segmentation of the CAPTCHA image into several layers. A classic segmentation method is k-means clustering [6]. Its basic principle is to look for k cluster centroids minimizing the average distance of all points to the nearest centroid. The algorithm starts from an initial condition, and the final solution is obtained by dynamically updating the centroids.

Morphological image processing: Mathematical morphology is a theory for analysis and processing of geometrical structures [9]. It is widely used in binary images processing. The basic idea is to probe an input image with one or more pre-defined "structuring elements". There are many different morphological operations, such as dilation, erosion, opening, closing, which can be used to filter noises and refine the shape of object(s) segmented from a given image.

Line detection: Some e-banking CAPTCHAs use random lines to form a grid in order to make segmentation more difficult. To break these CAPTCHAs, we can try to detect these grid lines and then remove them. Traditionally, lines can be detected by the Hough transform [34]. In e-banking CAPTCHAs, normally grid lines have only two orientations (vertical and horizontal) and they go through the whole image. In this case, a simplified Hough transform can be used.

Digital image inpainting: This is the technique to fix missing parts in a digital image [7]. The theory behind image inpainting is to predict missing pixels from their neighbors. Some of our attacks make use of a fast image inpainting technique proposed in [8] to remove real transaction data

and replace them with fake ones in the CAPTCHA images, and to remove unwanted objects like random grid lines.

Character segmentation: For an attack on an e-banking CAPTCHA scheme, the ultimate goal is often to recognize some characters in the CAPTCHA image. This requires segmentation of each character out of the image. By applying k-means clustering or simple thresholding, we can get a layer (i.e., a binary image) containing all characters. Then, we can segment those characters out of the layer as separate connected objects. When a connected object contains more than one character, they can be split if those characters have different colors. Sometimes we also need to merge some disconnected objects into a single character (e.g. "i" and "j") according to the geometric relationship between different parts of the character. To ensure the accuracy of the character segmentation process, different kinds of morphological operations are often used to remove noises and refine the shape of segmented characters.

Character recognition: After a character is segmented, it can be further recognized. In our attacks on transaction CAPTCHAs, two training-free character recognition methods are used: CW-SSIM [11] and cross-correlation [10]. Both methods are based on template matching; i.e., they compare the input with a number of reference images (templates) to look for the best match. We avoided training based methods in this study, due to the following reasons: 1) for transaction CAPTCHA schemes it was difficult to collect a large number of images as the training set; 2) the two training-free character recognition methods work well for the CAPTCHA schemes we studied; 3) training-based methods are normally faster than template matching based methods, but the latter are easier for our proof-of-concept implementations.

Recognition error detection and correction: Due to close correlation between some reference images, the character recognition algorithm may produce erroneous results for some inputs. We developed postprocessing methods to automatically detect and correct some of these recognition errors. These methods mainly exploit the context semantics and some inherent features of the recognized characters.

To simplify implementations of our proposed attacks, we chose MATLAB as the main programming language and platform. MATLAB has a very convenient Image Processing Toolbox and an interactive programming environment. Since MATLAB is an interpreted language, its programs are significantly less efficient than those developed in compiled programming languages like C/C++. Despite this fact, most of our attacks still can run in real time. All experiments reported in this paper were done on a laptop with an Intel Core2 Duo 2.4 GHz CPU and with 2 GB memory.

4. BREAKING GECAPTCHA

GeCaptcha is a typical e-banking CAPTCHA scheme for transaction verification and being used by around 800 German banks. To use the e-banking system with GeCaptcha, each user gets a paper list of indexed TANs from the bank in advance. After the user submits a transaction request, the e-banking server sends the user a GeCaptcha image like the one shown in Fig. 2, which is a mixture of a random grid, the user's birthday, transaction data and other texts including one line with a TAN index and the transaction time. After the TAN index n (in Fig. 2, $n = 158$) is observed, the user looks for the n-th TAN in the paper list, sends the TAN back to the e-banking server for confirming the transaction.

Figure 2: A GeCaptcha image. The big digits in the background compose the user's birthday. English translation of the three text lines: Line 1 – "GeCaptcha control picture for transfer"; Line 2 – "Amount in EUR: 999,99 Bank code: 10203040 Account Nr.: 12345678"; Line 3 – "Please enter the 158th TAN".

In a GeCaptcha image, the user's birthday is used as a shared secret between the user and the e-banking server, so that the server's identity can be authenticated by the user. To defeat automated attacks, the birthday is rendered using the following operations: 1) each digit is randomly rotated; 2) the font style of each digit is randomly determined; 3) each digit is randomly located; 4) all the digits are drawn between the transaction data and the random grid, which act like decoy objects (noises) in traditional login CAPTCHA schemes. The transaction data are on top of the other layers, and it is assumed that they cannot be easily manipulated without leaving any noticeable distortion to the original GeCaptcha image.

4.1 Two Approaches to Breaking GeCaptcha

To launch a MitM attack on GeCaptcha, the attacker (i.e., the malicious program) needs to manipulate the transaction data in real time without being noticed by the user. Let us assume that the user sends transaction data TD_1 to the server, and the data are manipulated by the malicious program to $\mathrm{TD}_2 \neq \mathrm{TD}_1$. Then, the malicious program will get a GeCaptcha image with transaction data TD_2 from the server. To spoof the user, the malicious program has to manipulate the GeCaptcha image by changing TD_2 to TD_1. There are two possible approaches: 1) locate TD_2, and replace them with TD_1; 2) recognize the user's birthday and the TAN index, and forge a GeCaptcha image with TD_1.

For both approaches, the malicious program needs to first segment different objects (the transaction data, the user's birthday and the TAN index) from the GeCaptcha image. We examined histograms of many GeCaptcha images and found out that different objects correspond to different peaks in the histogram. Since we know the number of layers, the k-means clustering method [6] can be applied to segment the GeCaptcha image. For the GeCaptcha image in Fig. 2, the segmentation result is shown in Fig. 3. Based on the successful segmentation of GeCaptcha images into several layers, two automated attacks can be developed using the two approaches enumerated above.

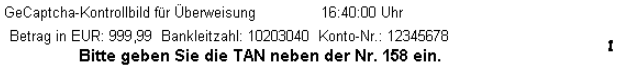
4.2 Automated Attack 1

In this attack, the malicious program achieves transaction manipulation via the following steps: Step 1) locate the text line with TD_2; Step 2) remove the text line with TD_2; Step 3) add a new line text with TD_1.

4.2.1 Step 1: Locating transaction data

The task of Step 1 is to further locate transaction data from the text layer segmented from the GeCaptcha image. The text layer contains three lines of texts: the line with transaction data, the line with TAN index, and the line with time. The order of the three lines is time-varying, so the

(a) different layers rendered with different colors



(b) the text layer

(c) the birthday layer

Figure 3: The segmentation result of the GeCaptcha image in Fig. 2.

malicious program needs a way to differentiate the line with transaction data from the other two lines.

One method to differentiate the transaction data from other texts is to recognize all the texts in the layer and then search for the keywords "Betrag in EUR", "Bankleitzahl" and "Konto-Nr.", which always appear in the line of transaction data. The recognition task can be done by an existing OCR tool due to the nearly perfect segmentation of the text layer. We tested MODI, the OCR tool included in Microsoft Office 2007, and the recognition rate is 95% for the text layer shown in Fig. 3(b). Based on such a nearly perfect recognition rate, the malicious program can easily know which line the transaction data belongs to.

While the OCR-based method works well, there is another more light-weight and robust method. It is based on the following observations: 1) the line with the TAN index is always boldfaced; 2) the line with time contains less characters than the other two lines; 3) the line with time contains a large white space. These observations imply that the average font weight (AFW) of the three lines can be very different. Let us denote the number of black pixels in a line by N_1, the number of all pixels in the bounding box of the line by N_2, the actual font size by b and the normalized font size by b_0. Then, the average font weight is defined by $\text{AFW} = (b \cdot N_1)/(b_0 \cdot N_2)$. We tested the AFWs of the three lines in 100 GeCaptcha images, and confirmed that the AFW can be used to differentiate different text lines reliably.

Based on the above finding, a more light-weight method can be easily designed to locate all the three lines. In addition to being more efficient, another advantage of the AFW-based method over the OCR-based method is that it is more robust against noise and segmentation errors.

4.2.2 *Step 2: Removing transaction data*

After locating the line with transaction data, we can try to remove the whole line by applying an image inpainting method. However, most image inpainting methods do not work well when there are too many edges around the missing parts. We found that the random grid lines and color shading of the background do introduce noticeable distortions. Figure 4 shows the results of applying the image inpainting method in [8] to the GeCaptcha image in Fig. 2 by taking the line with transaction data as the mask of the to-be-filled region. We can see some noticeable distortions such as broken grid lines.[3] We tried two other image inpainting methods [35, 36] and got similar results.

[3]Although users are not always sensitive to those subtle distortions, they can be easily trained to be more careful.

Figure 4: The result of removing transaction data by applying the image inpainting method in [8].

To overcome the problem, we have to handle pixels on the grid lines separately: they should be predicted from closest (not necessarily neighboring) pixels on grid lines and should not be used for predicting pixels that do not lie on grid lines. Based on this idea, we extended the fast image inpainting method proposed in [8]. The extended method needs to know where the grid lines are. As we described in Sec. 3, these (horizontal and vertical) grid lines can be detected by a simplified Hough transform. After the grid lines are localized, pixels on the random grid can be handled differently, so that no visible distortion will occur on and around grid lines. Figure 5 shows the result of the extended inpainting method. Comparing Fig. 5 with Fig. 4, we can see that our proposed inpainting method, while having the same level of complexity, creates significantly lesser distortion than general-purpose inpainting methods.

Figure 5: The result of applying the extended inpainting method to the GeCaptcha image in Fig. 2.

4.2.3 *Step 3: Adding user-expected transaction data*

After removing the transaction data TD_2, it is trivial to add the user-expected transaction data TD_1 to the vacant place in the GeCaptcha image. Figure 6 shows a forged GeCaptcha image by changing the transaction data to "Betrag in EUR: 1,00 Bankleitzahl: 18635402 Konto-Nr.: 1211855". We tested the image inpainting based attack on 100 GeCaptcha images collected from real user accounts and no visual distortion is observed in any forged GeCaptcha image, thus leading to the ideal success rate of 100%.

Figure 6: A forged GeCaptcha image from the GeCaptcha image in Fig. 2.

4.3 Automated Attack 2

The image inpainting based attack described above is blind, in the sense that it does not depend on a recognition task. However, if we can recognize the user's birthday and the TAN index embedded in the GeCaptcha image, a second attack can be developed. An additional advantage of the second attack is that the attacker can get the user's birthday, which is the user's private information that plays a key role in some backup authentication systems. The second attack consists of the following two stages.

Stage 1 – birthday recognition: The malicious program collects a number of GeCaptcha images, and tries to recognize the user's birthday. This stage is completely offline.

Stage 2 – transaction manipulation: For a new transaction request received from the user, the malicious program manipulates the transaction data, then locates (probably also recognizes) the TAN index from the server-generated GeCaptcha image, and finally sends a forged GeCaptcha image with the user's transaction data back to the user. This stage has to be done online in real time.

4.3.1 Stage 1: Offline birthday recognition

As shown in Sec. 4.1, the image segmentation process can produce a segmented layer with birthday. This layer can be further segmented to extract each birthday digit. Some morphological operations are needed to filter small objects and noises, and to refine the shapes of segmented birthday digits. Figure 7 shows the digits segmented from the birthday layer shown in Fig. 3(c).

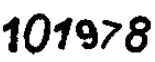


Figure 7: The eight birthday digits segmented from the birthday layer shown in Fig. 3c.

Since the birthday digits are normally rotated and the segmentation result is not always perfect, OCR tools do not work very well. Instead, we chose to use a training-free algorithm CW-SSIM [11] to recognize these digits. CW-SSIM denotes "complex wavelet based structural similarity", which is a full-reference image quality assessment (IQA) algorithm invariant to translation and small scaling/rotation. In [11], it was demonstrated how CW-SSIM can be used to achieve robust and highly accurate digit recognition.

To use CW-SSIM, we need a database of reference images of the to-be-recognized digits. We used a database with three rotation angles (0 and ± 15 degrees) and two different font styles (boldfaced, boldfaced italic). As a whole, there are $10 \times 3 \times 2 = 60$ reference images. Based on such a database, the birthday in Fig. 7 can be successfully recognized. For 100 GeCaptcha images, the success rate is 91%.

The success rate can be further improved by using image inpainting. The idea is to remove the whole text layer and the grid lines, which normally leads to a better segmentation result of the birthday layer and thus also the birthday digits. Figure 8 shows the result of removing all those unwanted objects from the GeCaptcha image Fig. 2. One can see that in the inpainted image the birthday becomes more salient in the background. For the simplified GeCaptcha image, a 3-means clustering process is used to extract the birthday for recognition. With the improved method, the success rate of birthday recognition becomes 100%.

Figure 8: The inpainting result of the GeCaptcha image in Fig 2, by removing all unwanted objects.

4.3.2 Stage 2: Online transaction manipulation

Once the user's birthday is broken, the malicious program can start manipulating transaction data and forging GeCaptcha images. To do so, the malicious program needs to locate the line with TAN index in the server-generated GeCaptcha image because the server expects a specific TAN for confirming the (manipulated) transaction. As we described in Sec. 4.2.1, we can segment the line with TAN index from the text layer by using the AFW-based method.

After extracting the line with TAN index, the malicious program can synthesize a fake GeCaptcha image from the following known information: the user's birthday, the line with TAN index, the original transaction data TD_1 and the current time. We developed an image generation engine to do this task. Figure 9 shows an example of the forged GeCaptcha image by our image generation engine.

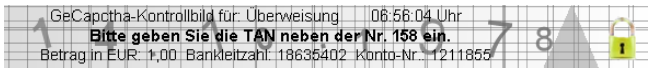


Figure 9: A GeCaptcha image synthesized from the image in Fig. 2 after the birthday is recognized.

4.4 Human-involved semi-automated attack

In Automated Attack 2, the first stage is to recognize the user's birthday, which is done offline. Instead of building its own recognizer, the malicious program can also send the segmented birthday to a human solver to recognize the birthday. Such an attack will be useful if GeCaptcha is enhanced to make the birthday recognition task very difficult.

The human solver is not necessarily the attacker himself. The malicious program can create a CAPTCHA image from the segmented birthday and send it to a web site under its control as a challenge for login, which will be solved by a visitor of the web site. After the malicious program obtains the recognized birthday from a human observer, the second stage of Automated Attack 2 can be launched as usual.

A salient feature of this attack is that the human solver is needed only once and the whole process afterwards is fully automated. This explains why we call it "human-involved semi-automated attack".

4.5 Efficiency of the proposed attacks

For 100 test images, the average running time of the inpainting based attack is around 250 ms, and that of Stage 2 (online transaction manipulation) of the recognition based attack is around 190 ms. The running time starts from reading the real image from hard disk and ends with storage of the forged image on hard disk.

Stage 1 (the birthday recognition part) of the birthday recognition attack is relatively slow because the CW-SSIM values have to be calculated for all the birthday digits and all the reference images. The average time of birthday recognition is around 5 seconds. The efficiency problem is not a big issue because: 1) the recognition stage runs offline; 2) the recognition can be made faster by replacing CW-SSIM with a training-based recognizer; 3) the MATLAB code we used has significant room for further optimization.

5. BREAKING CHCAPTCHA1 AND CHCAPTCHA2

ChCaptcha1 and ChCaptcha2 are e-banking CAPTCHA schemes used by two major banks in China. The two schemes are very similar to GeCaptcha, so they can be broken by generalizing the attacks described in the above section.

5.1 Breaking ChCaptcha1

ChCaptcha1 is similar to GeCaptcha, but with three main differences. First, there is no paper list of TANs issued to each user. Instead, four digits in the receiver's account number are randomly selected and rendered in color. The user is asked to input these four digits as a transaction-dependent TAN. Second, there is no secret shared between the user and the server for server authentication. Third, there are no random grid lines. Figure 10 shows a ChCaptcha1 image we collected from a real bank account. In the background of the ChCaptcha1 image, multiple copies of the bank's logo are embedded. We replace these logos by white disks with gray edges to avoid exposing the bank's identity.

Figure 10: A ChCaptcha1 image. English translation of the texts: Line 1 – "receiver's account"; Line 2 – "receiver's name"; Line 3 – "TAN Please input the big red digits in receiver's account".

In ChCaptcha1, the TAN is embedded into the CAPTCHA image, so a recognition based attack is able to break the CAPTCHA scheme. The attack is similar to Automated Attack 2 on GeCaptcha: layer segmentation → transaction data localization → TAN digit segmentation → TAN digit recognition. The segmentation results of the ChCaptcha1 scheme is nearly perfect and the TAN digits are not rotated, so the simpler correlation based method can be used for recognition. We tested the recognition based attack on 100 ChCaptcha1 images and achieved a success rate of 100%. The average running time of the attack is less than 150 ms.

5.2 Breaking ChCaptcha2

ChCaptcha2 does not depend on a paper list of TANs, either. A 5-digit TAN is dynamically generated and embedded into the CAPTCHA image like the user's birthday in a GeCaptcha image. Different from the ChCaptcha1 scheme, the ChCaptcha2 TAN is not transaction dependent. There are no random grid lines, either. Figure 11 shows a ChCaptcha2 image we collected from a real bank account.

Figure 11: A ChCaptcha2 image. English translation of the texts: Line 1 – "Attention! Please check the following information carefully"; Line 2 – "receiver's account"; Line 3 – "receiver's name"; Line 4 – "amount".

Compared with ChCaptcha1, ChCaptcha2 is more similar to GeCaptcha. The two attacks on GeCaptcha can both be generalized. The processes are nearly the same as those on GeCaptcha, except that the color information is also used for k-means clustering. We tested both attacks on 103 ChCaptcha2 images, and the success rate is 100%. The efficiency of the recognition based attack is relatively low, with an average running time of about 6-7 seconds. Note, however, that the malicious program does not need to respond to the server in real time as the CAPTCHA images are supposed to be solved by human users who can be very slow. On the other hand, the malicious program can still interact with the user in real time because it does not need to wait for the recognition result to forge a CAPTCHA image.

6. BREAKING LOGIN CAPTCHAS

In addition to the three e-banking CAPTCHA schemes for transaction verification, we found 41 e-banking CAPTCHA schemes for login. Our study on these e-banking CAPTCHA schemes is alarming: *all* of them are insecure against automated segmentation attacks. Some of them are designed in such a naive way that the segmentation information of the CAPTCHA images have been fully or partially encoded in the images themselves. Since character recognition is not difficult if the characters have been well segmented [19, 21], the success of our segmentation attacks have shown that all of these e-banking login CAPTCHA schemes are not secure.

Our segmentation attacks on the 41 e-banking CAPTCHA schemes for login are based on the same set of CAPTCHA-breaking tools described in Sec. 3, so we do not repeat the detail about how each login CAPTCHA scheme is broken. Instead, in Table 1 we show segmentation results of some selected login CAPTCHA schemes[4]. We also list the tool(s) we used and weakness(es) we exploited in our attacks. Character segmentation is used for all schemes, so it is not listed in the table to save space. For each e-banking login CAPTCHA scheme, we have tested the segmentation attack on at least 60 sample images to estimate the success rates.

7. MORE DISCUSSIONS

Our attacks on e-banking CAPTCHAs raise the question of whether financial institutions should continue to use CAPTCHAs for their e-banking services or they should leave them for more secure solutions. That is, the following question needs to be answered: *can we enhance the broken e-banking CAPTCHA schemes so that they are immune to the proposed attacks?* In this section, we first take a look at the case of e-banking CAPTCHAs for transaction verification and then discuss all e-banking CAPTCHAs as a whole.

7.1 Can transaction CAPTCHAs be enhanced?

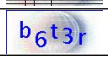
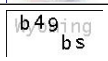
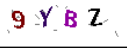
Due to the similarity of the three transaction e-banking CAPTCHA schemes under study, in this subsection we will focus on GeCaptcha to ease our discussion.

One simple improvement is to compress the image with a lossy algorithm like JPEG, in the hope that the boundaries between different objects are blurred so that the attacks become difficult. Unfortunately, our attacks can be easily enhanced to tolerate lossy compression by adding an additional noise filter. Our experiments showed that the inpainting based attack works even when the lowest quality factor of JPEG compression is used. As a consequence, lossy compression cannot enhance the security of GeCaptcha.

There are some other possible improvements: replacing the random grid lines by random curves, balancing the three text lines so that they have similar AFWs, changing the birthday to a different form such as a number of secret

[4]Due to the page limit, we cannot list all login CAPTCHA schemes here. A full list is available at `http://www.hooklee.com/default.asp?t=eBankingCAPTCHAs`.

Table 1: Selected e-banking login CAPTCHA schemes we studied, with results of our segmentation attacks.

Financial institution(s)/e-banking login CAPTCHA scheme	*CAPTCHA image(s)*	*Segmentation result(s)*	*Tool(s) used, Weakness(es) exploited*	*Success rate*
13 German banks		534261	3-means clustering, morphological operations	100%
Hundreds other German banks		Q4B254	2-means clustering, line detection, image inpainting	100%
A Swiss bank with branches in Europe, Asia, North America and Africa		49575	2-means clustering	100%
A bank based in Latin America with branches in Europe, Asia, Australasia and Africa		31289	2-means clustering	100%
US e-banking CAPTCHA 1		454e	2/3-means clustering, line detection, image inpainting	100%
US e-banking CAPTCHA 2		b6t3r	3-means clustering	100%
US e-banking CAPTCHA 3		b49bs	3-means clustering	100%
US e-banking CAPTCHA 4		3264	2/3-means clustering	100%
Three CUs in Australia		bQUM	3-means clustering, morphological operations	99.5%
Chinese e-banking CAPTCHA 1		3571	3-means clustering	100%
Chinese e-banking CAPTCHA 2		9YBZ	2-means clustering, image inpainting	100%
Chinese e-banking CAPTCHA 3		88071	4-means clustering, morphological opening	100%
Chinese e-banking CAPTCHA 4		WLAV3	morphological cleaning, character intensity < 120	100%
Chinese e-banking CAPTCHA 5		bYjg	3-means clustering, morphological cleaning	98.3%
Chinese e-banking CAPTCHA 6		3m9nxn	grayscale foreground vs. colored noises	100%
Chinese e-banking CAPTCHA 7		smib	2-means clustering	100%
Chinese e-banking CAPTCHA 8		xjva3	3-means clustering	100%
Chinese e-banking CAPTCHA 9		MKJq	2/3-means clustering	100%
Chinese e-banking CAPTCHA 10		mj4k	2-means clustering, morphological operations	95.1%

icons, changing the order of different layers, etc. Unfortunately, none of these improvements can resist the two proposed automated attacks simultaneously. Even if we combine all of them, the human-involved semi-automated attack still works, as long as the text layer can be extracted.

A more effective improvement is to change the gray scale of different objects in the GeCaptcha image so that they overlap with each other in the histogram. This will make k-means clustering fail. For an enhanced GeCaptcha image shown in Fig. 12, none of our proposed attacks works.

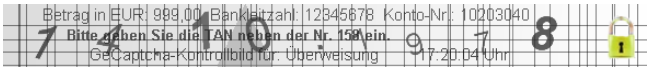
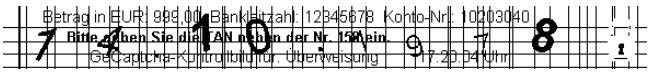


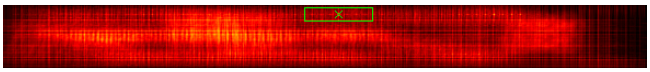
Figure 12: An enhanced GeCaptcha image in which different foreground layers have similar gray values.

Unfortunately, the failure of our proposed attacks does not mean that more advanced attacks cannot be developed. In fact, based on an idea similar to the generalized Hough transform [37], we can develop a more advanced attack. The basic idea is as follows: since the malicious program knows many texts embedded in a GeCaptcha image and the types of distortions applied to these texts, it can build a database of shape templates of these texts according to all the possible rendering configurations. Then, the malicious program tries to search all shape templates in the GeCaptcha image to find the one leading to the best match at a specific location. This will tell where the target texts and their contextual texts are, which can then be segmented and manipulated or recognized. Here, we can show an example for the enhanced GeCaptcha image in Fig. 12. Let us assume that the malicious program wants to manipulate the receiver's account number. Since the malicious program knows the receiver's account number, it can create a number of templates and search for them in the GeCaptcha image. The maximal correlation will show the correct location of the receiver's account number. A 2-means clustering process can be performed before the searching process starts so that only the foreground will be matched. Since the background and the foreground have to maintain a considerable contrast to make the foreground visible, the 2-means clustering should always work very well. Figure 13 shows the result of searching for the receiver's account number. We can see that it is exactly localized, and hence transaction manipulation becomes easy.

The template searching based attack is very powerful. It



(a) result of 2-means clustering based segmentation



(b) correlation map

Figure 13: The result of searching for the account number "12345678" in Fig. 12. The green rectangle shows the location with the maximal correlation.

works well for transaction e-banking CAPTCHAs because many characters known to the malicious program (e.g., transaction data/time) have to be embedded into the CAPTCHA image. To improve security against such an attack, we must increase the number of distinct text rendering and distortion methods so that the searching process becomes extremely slow and/or storing all templates becomes impossible. But this will increase the complexity of the CAPTCHA scheme itself. It is also doubtful if there are enough rendering parameters and distortions because: 1) both machines and humans are not sensitive to small changes of rendered texts; 2) distortions have to remain light to maintain visibility of the texts and usability of the CAPTCHA scheme.

7.2 Are CAPTCHAs good for e-banking at all?

While it seems difficult to improve the security of transaction CAPTCHAs, we still have the last (somewhat circular reasoning) resort: to render all texts as strong CAPTCHAs. This is also the way to enhance e-banking CAPTCHAs for login. Here, "strong" means that *any* automated attacks based on the state-of-the-art techniques is impractical. Then, the question becomes if such strong CAPTCHAs do exist. This question is difficult to answer conclusively as an accurate definition of hard AI problems does not exist. Moreover, unavailability of (publicly) known attacks on a specific CAPTCHA scheme does not mean that such attacks do not exist. For instance, Google's reCAPTCHA uses words that cannot be recognized by the state-of-art OCR tools to generate strong CAPTCHA images, which is believed to be secure due to the creative way of CAPTCHA image generation. However, recently some automated attacks on reCAPTCHA were reported [38]. Although reCAPTCHA can be updated, the attacks will also evolve and new attacks may emerge.

In addition to the security problem of CAPTCHAs, there is also a well-known tradeoff between security and usability [39]. To make a CAPTCHA scheme more secure, often usability has to be compromised, and vice versa. For e-banking systems, this security-usability tradeoff is more critical. This is because customers who have trouble with strong CAPTCHAs may complain and even switch to other financial institutions. We believe this is part of the reason why many financial institutions have not deployed CAPTCHAs or have deployed less secure (but more usable) CAPTCHAs.

Relying on CAPTCHAs for e-banking has a salient drawback related to the tradeoff between security and usability: financial institutions have to be prepared to patch their system at *any* time since new attacks may appear at *any* time. This will inevitably increase maintenance costs. Financial institutions may choose to patch their e-banking systems less frequently, thus leaving security holes in their systems.

The nature of CAPTCHAs implies that they are vulnerable to human-involved attacks. Compared to other applications of CAPTCHAs, attackers will have more incentives to employ cheap labor to solve e-banking CAPTCHAs [40]. Although the attacker has to ensure real-time response in some cases, this can be achieved if the attacker can exploit the user base of a popular web site. In case the attacker can infect a large number of computers, which has already been happening in today's Internet, the chance to be successful for at least one victim can be practically high. Since 2007, some malware has been found to use this strategy [41].

We can also compare e-banking CAPTCHAs with traditional schemes and from an economic perspective. In traditional applications of CAPTCHAs, breaking a CAPTCHA scheme leads to only abuse of the resources protected by the CAPTCHA scheme. However, for e-banking systems, breaking a CAPTCHA scheme can cause a potentially huge loss for both users and banks. As a whole, we have the feeling that CAPTCHAs may be incapable of protecting e-banking systems, due to the higher security requirements. In [42], Jakobsson expressed the same concern and proposed an alternative solution called "CAPTCHA-free throttling".

Based on the above discussion, we call for cautions in deploying e-banking CAPTCHAs. For financial institutions relying only on CAPTCHAs, we suggest moving to alternative solutions or at least combining CAPTCHAs with other solutions. Among all alternative solutions, we feel that hardware security tokens are more promising. Not all hardware based solutions can resist MitM attacks, but at least some can, using transaction-dependent TANs, encrypted channels, and/or trusted display/keypad. For instance, if a hardware token is equipped with a trusted display and can sign the transaction data, the user can ensure "what she sees is what she signs", thus rendering MitM attacks impossible. Of course, hardware based solutions are not perfect, either. Their main disadvantage is that either the financial institution or the customer has to bear the additional costs.

8. CONCLUSIONS

This paper reports a comprehensive study on e-banking CAPTCHA schemes, including three for transaction verification and 41 schemes for login. We propose a new set of image processing and pattern recognition techniques to break all the e-banking CAPTCHA schemes with a success rate equal to or close to 100%. We have also shown that it is a nontrivial task to enhance e-banking CAPTCHA schemes to ensure both security and usability. We thus raise the question if financial institutions should rely on e-banking CAPTCHAs at all. Our opinion is that e-banking CAPTCHAs are better replaced by other alternative solutions such as those based on hardware security tokens.

9. ACKNOWLEDGMENTS

Shujun Li was supported by the Zukunftskolleg ("Future College") of the University of Konstanz, Germany, which is part of the "Excellence Initiative" Program of the DFG (German Research Foundation). The work of S. Amier Haider Shah, M. Asad Usman Khan and Syed Ali Khayam was partially supported by the Pakistan National ICT R&D Fund.

10. REFERENCES

[1] American Bankers Association. Consumers prefer online banking. `http://www.aba.com/Press+Room/`

092109ConsumerSurveyPBM.htm, 2009.

[2] Bank Austria. mobileTAN information. http://www.bankaustria.at/de/19741.html, 2007.

[3] Cronto Limited. Cronto's visual cryptogram. http://www.cronto.com/visual_cryptogram.htm, 2008.

[4] Volksbank Rhein-Ruhr eG. Bankgeschäfte online abwickeln: Mit Sm@rtTAN optic bequem und sicher im Netz. http://www.voba-rhein-ruhr.de/privatkunden/ebank/SMTop.html, 2009.

[5] T. Weigold, T. Kramp, R. Hermann, F. Höring, P. Buhler, and M. Baentsch. The Zurich Trusted Information Channel – an efficient defence against man-in-the-middle and malicious software attacks. In *TRUST'2008*, pages 75–91.

[6] G. A. F. Seber. *Multivariate Observations*. John Wiley & Sons, Inc., 2004.

[7] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *SIGGRAPH'2000*, pages 417–424.

[8] M. M. Oliveira, B. Bowen, R. McKenna, and Y.-S. Chang. Fast digital image inpainting. In *IASTED VII'2001*, pages 261–266.

[9] F. Y. Shin. *Image Processing and Mathematical Morphology*. CRC, 2009.

[10] S. J. Orfanidis. *Optimum Signal Processing*. 2 edition, 2007. http://www.ece.rutgers.edu/~orfanidi/osp2e.

[11] Z. Wang and E. P. Simoncelli. Translation insensitive image similarity in complex wavelet domain. In *ICASSP'2005*, pages 573–576.

[12] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *EUROCRYPT'2003*, pages 294–311.

[13] J. Elson, J. R. Douceur, J. Howell, and J. Saul. Asirra: A CAPTCHA that exploits interest-aligned manual image categorization. In *CCS'2007*, pages 366–374.

[14] A. Basso and F. Bergadano. Anti-bot strategies based on human interactive proofs. In *Handbook of Information and Communication Security*, chapter 15, pages 273–291. Springer, 2010.

[15] G. Mori and J. Malik. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In *CVPR'2003*, pages 134–141.

[16] G. Moy, N. Jones, C. Harkless, and R. Potter. Distortion estimation techniques in solving visual CAPTCHAs. In *CVPR'2004*, pages 23–28.

[17] K. Chellapilla and P. Y. Simard. Using machine learning to break visual Human Interaction Proofs (HIPs). In *NIPS'2004*, pages 265–272, 2005.

[18] S. Hocevar. PWNtcha: Pretend we're not a Turing computer but a human antagonist. http://caca.zoy.org/wiki/PWNtcha, 2004.

[19] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski. Computers beat humans at single character recognition in reading based human interaction proofs (HIPs). In *CEAS'2005*.

[20] J. Yan and A. S. El Ahmad. Breaking visual CAPTCHAs with naïve pattern recognition algorithms. In *ACSAC'2007*, pages 279–291.

[21] J. Yan and A. S. El Ahmad. A low-cost attack on a Microsoft CAPTCHA. In *CCS'2008*, pages 543–554.

[22] P. Golle. Machine learning attacks against the Asirra CAPTCHA. In *CCS'2008*, pages 535–542.

[23] J. Tam, J. Simsa, S. Hyde, and L. von Ahn. Breaking audio CAPTCHAs. In *NIPS'2008*, pages 1625–1632, 2009.

[24] E. Bursztein and S. Bethard. Decaptcha: Breaking 75% of eBay audio CAPTCHAs. In *WOOT'2009*.

[25] C. J. Hernandez-Castro and A. Ribagorda. Pitfalls in CAPTCHA design and implementation: The Math CAPTCHA, a case study. *Computers & Security*, 29(1):141–157, 2010.

[26] A. Hindle, M. W. Godfrey, and R. C. Holt. Reverse engineering CAPTCHAs. In *WCRE'2009*, pages 59–68.

[27] C. J. Hernandez-Castro and A. Ribagorda. Remotely telling humans and computers apart: An unsolved problem. In *iNetSec'2009*.

[28] B. Pinkas and T. Sander. Securing passwords against dictionary attacks. In *CCS'2002*, pages 161–170.

[29] C. J. Mitchell. Using human interactive proofs to secure human-machine interactions via untrusted intermediaries. In *Security Protocols'2006*, pages 164–170, 2009.

[30] I. Fischer and T. Herfet. Visual CAPTCHAs for document authentication. In *MMSP'2006*, pages 471–474.

[31] M. Szydlowski, C. Kruegel, and E. Kirda. Secure input for Web applications. In *ACSAC'2007*, pages 375–384.

[32] D. J. Steeves and M. W. Snyder. Secure online transactions using a CAPTCHA image as a watermark. US Patent 2007/0005500.

[33] W. Wieser. Captcha recognition via averaging. http://www.triplespark.net/misc/captcha, 2007.

[34] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice Hall, 2008.

[35] A. Criminisi, P. Pérez, and K. Toyama. Object removal by exemplar-based inpainting. In *CVPR'2003*, pages 721–728.

[36] P. Getreuer. tvreg: Variational imaging methods for denoising, deconvolution, inpainting, and segmentation. http://www.math.ucla.edu/~getreuer/tvreg.html, 2009.

[37] D. H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[38] J. Wilkins. Strong CAPTCHA guidelines: v1.2. http://bitland.net/captcha.pdf, December 2009.

[39] J. Yan and A. S. El Ahmad. Usability of CAPTCHAs or usability issues in CAPTCHA design. In *SOUPS'2008*, pages 44–52.

[40] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage. Re: CAPTCHAs – Understanding CAPTCHA-solving services in an economic context. In *USENIX Security'2010*.

[41] BBC News. PC stripper helps spam to spread. http://news.bbc.co.uk/2/hi/technology/7067962.stm, 2007.

[42] M. Jakobsson. CAPTCHA-free throttling. In *AISec'2009*, pages 15–21.