

# Breaking e-Banking CAPTCHAs\*

Shujun Li<sup>1</sup>, Syed Amier Haider Shah<sup>2</sup>, Muhammad Asad Usman Khan<sup>2</sup>,  
Syed Ali Khayam<sup>2</sup>, Ahmad-Reza Sadeghi<sup>3</sup> and Roland Schmitz<sup>4</sup>

<sup>1</sup> University of Konstanz, Germany

<sup>2</sup> National University of Science and Technology (NUST), Pakistan

<sup>3</sup> Ruhr-University of Bochum, Germany

<sup>4</sup> 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. Based on our results we believe that currently CAPTCHAs are incapable of offering adequate security for high-value applications like e-banking.

## 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<sup>1</sup> to protect their e-banking systems from automated attacks. In addition to traditional CAPTCHAs for preventing automated login attempts, some major financial institutions in Germany and China 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) based phishing attacks [2] and Man-in-the-Browser (MitB) attacks [3], 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 [4]. 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.<sup>2</sup>

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 [6, 7]; hardware for establishing encrypted channel over untrusted network and/or computer [8, 9]; hardware with trusted display/keypad for generating transaction-dependent TANs or signatures [10–12]; and solutions based on

\*Companion web page: <http://www.hooklee.com/default.asp?t=eBankingCAPTCHAs>.

<sup>1</sup>CAPTCHA is an acronym for “Completely Automated Public Turing test to tell Computers and Humans Apart”.

<sup>2</sup>A recent report released by the Anti-Phishing Working Group (APWG) [5] shows that the infection rate of malware has reached an alarming rate around 48% (among 21.5 million scanned computers) for the last two quarters of 2009, and that around 24% of all the detected malware can be linked to financial crimes.

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 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.<sup>3</sup> Our attacks are based on a new set of image processing and pattern recognition tools, including  $k$ -means clustering [13], digital image inpainting [14, 15], morphological image processing [16], character recognition based on cross-correlation [17] and an image quality assessment (IQA) method called CW-SSIM [18]. 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 in MATLAB, which is the programming language and environment we chose for our proof-of-concept implementations.

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. Our (conservative) suggestion is to replace e-banking CAPTCHAs with other alternative solutions.

*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, which will make automated online password attacks possible. 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

CAPTCHA is the acronym of “Completely Automated Public Turing test to tell Computers and Humans Apart”. The term was coined by von Ahn et al. in 2000 [19]. Some people prefer another term “Human Interactive Proof” (HIP). However, according to the topics discussed at the first workshop on HIPs in January 2002, the term “HIP” covers more human-involved interactive protocols like human authentication protocols against shoulder surfers. Yet another (less popular) term used is “Reverse Turing Test” (RTT). In this paper, we adhere to the term CAPTCHA.

The main application of CAPTCHAs [19] is to protect online resources from being abused by automated programs (i.e., web bots). The idea can be traced back to an unpublished manuscript of Naor in 1996 [20] and a system deployed by AltaVista to prevent automated URL submissions to their search engine [21]. Nowadays, CAPTCHAs have been widely deployed by many web sites.

A CAPTCHA scheme is a challenge-response authentication protocol based on a hard AI (artificial intelligence) 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

---

<sup>3</sup>These are *all* the e-banking CAPTCHA schemes that we had found when we submitted the final edition of this paper.

humans can. The hard AI problem here is text recognition. 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 [22], 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 [23, 24], video [25], and 3-D models [26–28]. Readers are referred to [29] 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 [30], 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 [31, 32] and one image-based CAPTCHA scheme called “Clock Face HIP” [33]. Moreover, Hocevar demonstrated results of his attacks on quite a number of CAPTCHA schemes on his web site [34], which reveals some common pitfalls of weak CAPTCHAs. In [35], Aboufadel et al. reported a recognition based attack on a CAPTCHA based on distorted texts, which was used by the Priority Club of the Holiday Inn chain of hotels. In [36], 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 [37], 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 [38], 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. In [39], Kluever reported recognition based attacks on PayPal CAPTCHAs.

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 [40], 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 [41, 42], attacks to some deployed audio CAPTCHA schemes were reported.

There are also attacks exploiting implementation flaws. In [43], Hernandez-Castro and Ribagorda proposed a side-channel attack on a CAPTCHA scheme based on solving mathematical problems. Similarly, in [44] it was shown that a side-channel attack on the Asirra CAPTCHA can be developed based on a randomness test tool ENT, which can achieve a better success rate than random guess. In [45], Caine and Hengartner pointed out a drawback of a distributed CAPTCHA protocol, which allowed an attacker to get multiple CAPTCHA images with the same solution for voting. In [46], Hindle et al. showed that reverse engineering can help to design new attacks based on comparing the results of a cracked CAPTCHA generator (or an emulator) against real CAPTCHAs. Recently, Hernandez-Castro and Ribagorda pointed out some common problems and design pitfalls of many CAPTCHA schemes [47].

## 2.2 CAPTCHAs for e-banking

One of the main applications of CAPTCHAs is to prevent automated online password attacks [48]. 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 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 around the world. 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. We have an interesting observation: financial institutions (of any kind) in China, and small/medium-sized financial institutions in other countries are more active in deploying e-banking CAPTCHAs.

Deployment of e-banking CAPTCHAs in China is so popular that it has become a standard component 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 in their e-banking systems.

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 [49], 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 [50], Fischer and Herfet proposed to use CAPTCHAs for e-banking transaction verification. In [51], Szydowski et al. proposed a CAPTCHA-based software keypad for securing web input of online transactions. In [52,53], 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 [54], Wieser described an attack on a login CAPTCHA scheme deployed by a German bank. This attack depends on a design flaw, which allows the attacker to collect multiple CAPTCHA images with the same texts. We noticed that this flaw had been fixed. In [55], Leung claimed an attack on a CAPTCHA-based e-banking solution deployed by the Bank of East Asia in Hong Kong. However, this e-banking solution is actually not based on CAPTCHA, but a scrambled software keyboard which cannot resist pattern recognition at all.

### 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. Figure 2 shows how these tools are used in our attacks to break different e-banking CAPTCHAs for both login and transaction verification. The dotted parts are only used for breaking CAPTCHAs for transaction verification. 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 [13]. 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 and it normally differs from the initial condition.

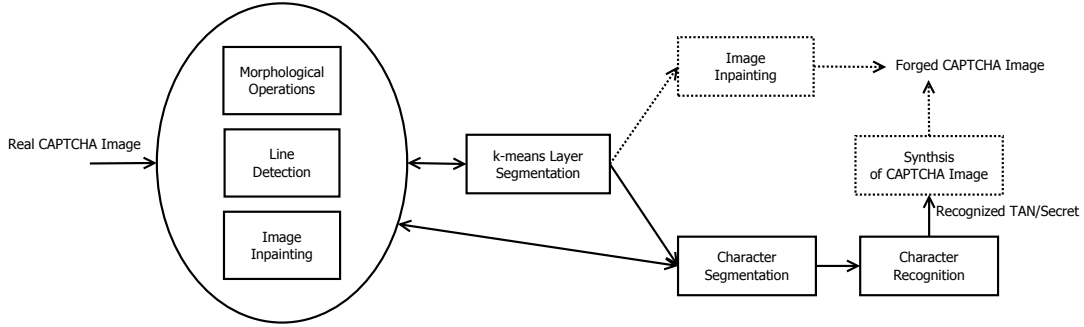


Figure 2: How different tools are used in our attacks on e-banking CAPTCHAs.

To apply  $k$ -means clustering, the number of clusters should be known in advance, otherwise it has to be guessed by obtained by a pre-processing step. It is also beneficial if a good initial condition is known. For some CAPTCHA schemes, we use  $k_1$ -means clustering to get part of the whole image for estimating the initial condition of a later  $k_2$ -means clustering process.

*Morphological image processing:* Mathematical morphology is a theory for analysis and processing of geometrical structures [16]. 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”. By defining different structure elements and probing the binary image in different operations, we can get different processing results of the binary image. 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. In our attacks on e-banking CAPTCHAs, morphological operations are heavily used to enhance performance of the results of many steps.

*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 in a binary image can be detected by the Hough transform [56]. 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 by checking points on the image boundary and the only two possible orientations.

*Digital image inpainting:* This is the technique to fix missing parts in a digital image [14]. 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 [15] to remove real transaction data and replace them with fake ones in the CAPTCHA images, and to remove unwanted objects like random grid lines. We avoided more advanced image inpainting methods because they are normally computationally expensive and therefore cannot be used in real-time attacks.

*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 [18] and cross-correlation [17]. 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. For instance, for the e-banking CAPTCHA scheme

GeCaptcha discussed in Sec. 4, the recognized characters should form a user’s birthday, which leads to a number of constraints and correlation between adjacent characters. For the e-banking CAPTCHA scheme ChCaptcha2 discussed in Sec. 5.2, the Euler numbers of recognized digits are checked to correct potential errors related to “5” and “6”.

To simplify implementations of our proposed attacks, we chose MATLAB as the main programming language and platform.<sup>4</sup> 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. 3, 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. 3,  $n = 158$ ) 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 3: 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  $TD_2 \neq TD_1$ . 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  $TD_1$ ; or
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. Figure 4 shows the overlapped histograms of three typical GeCaptcha images, from which we can clearly see the four separated layers: texts, birthday, random grid and the background. Since we know the number of layers, the  $k$ -means clustering method [13] can be applied to segment the GeCaptcha image. For the GeCaptcha image in Fig. 3, the segmentation result is shown in Fig. 5. Based on the successful segmentation of GeCaptcha images into several layers, two automated attacks can be developed using the two approaches enumerated above.

<sup>4</sup>Due to some essential functionality limitations of MATLAB, part of our code was written in Java and C.

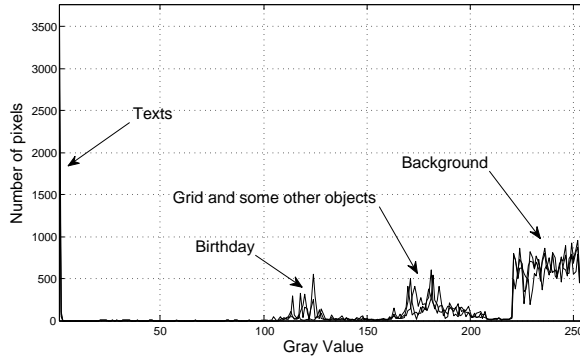


Figure 4: Histograms of three GeCaptcha images.

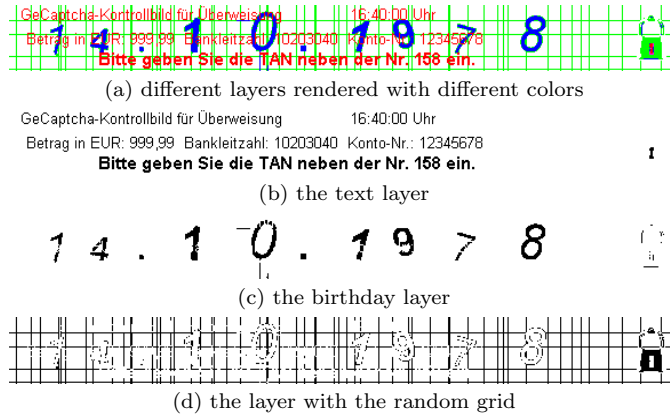


Figure 5: The segmentation result of the GeCaptcha image in Fig. 3.

## 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 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. To handle possible recognition errors, the searching process should be designed to search the keywords based on lexical similarity rather than an exact match. This method normally works quite well because the text layer is segmented nearly perfectly as shown in Fig. 5(b). The recognition task can be done by an existing OCR (optical character recognition) tool due to the nearly perfect segmentation of the text layer. We tested MODI(Microsoft Office Document Imaging), the OCR tool included in Microsoft Office 2007, and got the recognition result shown in Fig. 6.<sup>5</sup> We can see that the whole line of transaction data is correctly recognized and the overall recognition rate is  $(165 - 4)/165 \approx 97.6\%$ .

The result is stable for all GeCaptcha images we tested. Based on such a nearly perfect recognition rate, the malicious program can easily know which line the transaction data belongs to.

<sup>5</sup>Since the extracted layer is relatively small, we enlarged the image by 2 times to make the OCR tool more robust.



GeCaptcha-Kontrollbild für Überweisung 16:40:00 Uhr  
 Betrag in FUR: 999,99 Bankleitzahl: 10203040 Konto-Nr.: 12345678  
 Bitte geben Sie die TAN neben der Nr. 158 em.

Figure 6: The recognition result of the text layer shown in Fig. 5(b). The recognition errors are underlined.

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 between the phrase “GeCaptcha-Kontrollbild für: Überweisung” and the time. These observations imply that the average font weight (AFW) of the three lines can be very different. Note that the average font weight varies w.r.t. the font size, so different font weights should be compared after normalizing to a fixed font size. 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  $AFW = (b \cdot N_1) / (b_0 \cdot N_2)$ . Figure 7 shows the average font weights of the three text lines in 100 GeCaptcha images, from which we can see that different text lines indeed correspond to different average font weights.

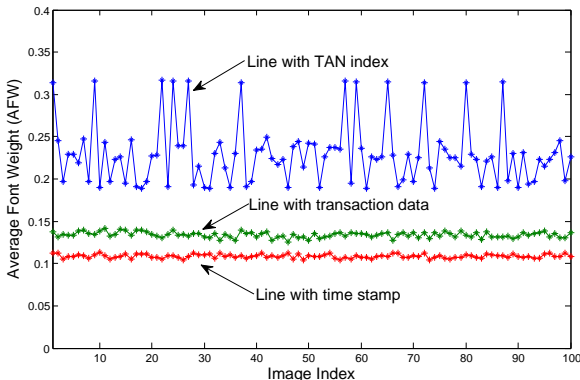


Figure 7: Average font weights of the three lines in 100 GeCaptcha images, when  $b_0 = 18$ .

Based on the above finding, a more light-weight method can be easily designed to locate all the three lines including the one with the transaction data. 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 8(a) shows the results of applying the image inpainting method in [15] to the GeCaptcha image in Fig. 3 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.<sup>6</sup> We tried two other image inpainting methods [57, 58] and got similar results as shown in Fig.8(b) and (c).

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 [15]. 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. Figure 9 shows the reconstructed grid based on the segmentation result shown in Fig. 5. 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 10 shows the result of the extended inpainting method. Comparing Fig. 10 with Fig. 8, we can see that our proposed inpainting method, while having the same level of complexity, creates significantly lesser distortion than general-purpose inpainting methods.

<sup>6</sup>Although users are not always sensitive to those subtle distortions, they can be easily trained to be more careful. The financial institutions may also issue auxiliary hardware/software tools to assist their customers.



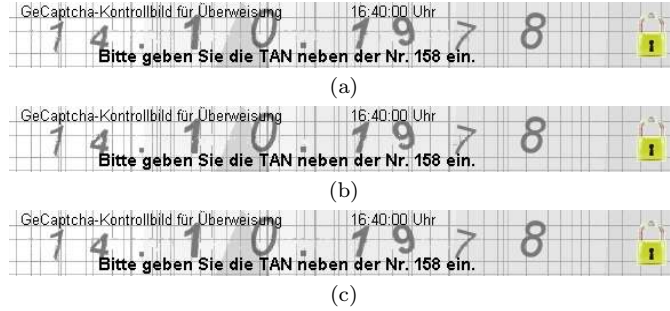


Figure 8: The results of removing the text line with transaction data by applying (a) the fast digital image inpainting method proposed in [15], (b) the image inpainting method reported in [57] and (c) the image inpainting method based on total variation (TV) regularization [58].

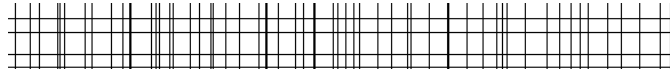


Figure 9: The reconstructed grid lines of the GeCaptcha image in Fig. 3.



Figure 10: The result of applying the extended inpainting method to the GeCaptcha image in Fig. 3.

#### 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 11 shows a forged GeCaptcha image by changing the transaction data from “Betrag in EUR: 999,99 Bankleitzahl: 10203040 Konto-Nr.: 12345678” 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 11: A forged GeCaptcha image from the GeCaptcha image in Fig. 3.

### 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 12 shows the digits segmented from the birthday layer shown in Fig. 5(c). We use morphological dilation to thicken all birthday digits, which is helpful in increasing the robustness and accuracy of birthday recognition.



Figure 12: The eight birthday digits segmented from the birthday layer shown in Fig. 5c.

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 [18] 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 [18], 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  $\pm 15$  degrees) and two different font styles (boldfaced, boldfaced italic). We only use boldfaced font styles because all the birthday digits are intentionally dilated. As a whole, there are  $10 \times 3 \times 2 = 60$  reference images. Based on such a database, the birthday in Fig. 12 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 13 shows the result of removing all those unwanted objects from the GeCaptcha image Fig. 3. 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. The result obtained from Fig. 13 is shown in Fig. 14. While this one is actually slightly worse than the one shown in Fig. 12, our experiments on 100 real GeCaptcha images have shown that the inpainting helps to produce better results in most case. With the improved method, the success rate of birthday recognition becomes 100%.



Figure 13: The inpainting result of the GeCaptcha image in Fig 3, by removing all unwanted objects.



(a) birthday layer



(b) extracted birthday

Figure 14: The segmented birthday layer and birthday digits from Fig. 13.

### 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 15 shows an example of the forged GeCaptcha image by our image generation engine.

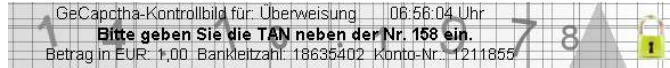


Figure 15: A GeCaptcha image synthesized from the image in Fig. 3 after the birthday “14.10.1978” is recognized.

It deserves noting that we can also try to recognize the TAN index. This will be helpful if the GeCaptcha scheme is enhanced to complicate perfect segmentation of the line with the TAN index. We applied GOCR [59], an open-source ORC tool, to recognize the whole line with the TAN index, and then search for the context to get the TAN index. The success rate turned out to be 100% for 100 GeCaptcha images.

#### 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. In case the malicious program has access to a web site with a large user base, this birthday recognition task may be done in real time. 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(e.g., switching from MATLAB to C).

## 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

#### 5.1.1 How does ChCaptcha1 work?

The ChCaptcha1 scheme for transaction verification is very similar to GeCaptcha, but there are some essential differences. First of all, there is no iTAN list issued to each user in advance. Instead, four digits are randomly selected from the receiver’s account number to form a transaction-dependent TAN. Second, the server-generated CAPTCHA images do not contain information for server authentication. Third, there are no random grid lines in the CAPTCHA image. The process of making an online transaction is similar to GeCaptcha: first the user sends transaction data to the e-banking server, then the server generates a CAPTCHA image (see Fig. 16) with the transaction-dependent TAN back to the user, and finally the user inputs the TAN to confirm the transaction.<sup>7</sup> Similar to GeCaptcha, all transaction data and the TAN is drawn on the top layer of the image. Instead of using

<sup>7</sup>The ChCaptcha1 e-banking system also supports user certificate stored in a USB-key. In case the user is possessing such a USB-key, she needs to connect her USB-key to the computer so that the transaction data will be signed. Such a mechanism does not enhance security against MitM attacks since the USB-key can be spoofed to sign the manipulated transaction.

random grid lines, multiple copies of the bank logo<sup>8</sup> and random strokes are used as decoy objects in the background. The color of the TAN digits is also randomly selected from red, green, blue, and yellow. Two different font faces are randomly chosen for the TAN digits. To enhance readability, the TAN digits are highlighted in color.



Figure 16: A ChCaptcha 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”.

### 5.1.2 Automated attack based on TAN recognition

Since the TAN is dependent on the transaction data, the automated attack based on image inpainting does not work any more. Instead, the malicious program has to recognize the TAN and send it to the server without any help from the user. In the meantime, the malicious program can directly forge a fake CAPTCHA image and show it to spoof the user.

To recognize the TAN, the first step is again to segment the TAN digits out of the CAPTCHA image. Observing the ChCaptcha image shown in Fig. 16, we can see that only the TAN digits and the two Chinese characters are in color. Therefore, a simple 2-means clustering is able to segment the whole image into two layers: pixels in color and all other pixels. Assuming a pixel value is represented as  $(R, G, B)$ , we can differentiate a color and a gray value from the standard deviation of the vector  $(R, G, B)$  or the distance from  $(R, G, B)$  to the 3-D line  $R = G = B$ . For the ChCaptcha image shown in Fig. 16, the 2-means segmentation result is given in Fig. 17. Although the segmentation result is already very good, we can further improve it by applying another 5-means clustering to differentiate the following layers: the black texts, the color texts, the multiple copies of the bank’s logo, the noises, and the background. A result of the 5-means segmentation is shown in Fig. 18.

**1 670**

红色

Figure 17: The result of the 2-means segmentation of the ChCaptcha image in Fig. 16.

**1 670**

红色

Figure 18: The result of the 5-means segmentation of the ChCaptcha image in Fig. 16.

After the layer of texts in color is segmented, we can easily detect the line of TAN digits from the line of two Chinese characters because the former is always wider than the latter and in most cases also contains more pixels. Then the TAN digits can be easily segmented with nearly perfect quality as shown in Fig. 19.

**1670**

Figure 19: The extracted TAN digits from segmented layer of Fig. 18.

After the successful segmentation of the TAN digits, we can try to recognize them. The nearly perfect segmentation result allows us to use a simpler character recognition method based on 2-D correlation between the segmented digits and reference images. Since there are only two different font faces without rotation and italic style, the total number of reference images is only  $10 \times 2 = 20$ . We tested the TAN recognition based attack on 100 ChCaptcha images and achieved a success rate of 100%. Due to the simplicity of 2-D correlation calculation process, the efficiency of the attack on the ChCaptcha scheme is very high. The average running time of our implementation is less than 150 ms.

<sup>8</sup>In Fig. 16 the logos are replaced by white disks with gray edge to avoid exposing the bank’s identity.

## 5.2 Breaking ChCaptcha2

### 5.2.1 How does ChCaptcha2 work?

The ChCaptcha2 scheme for transaction verification is also very similar to GeCaptcha. Like the ChCaptcha1 scheme, there is no TAN list issued to the user in advance. Instead, a 5-digit TAN is dynamically generated and embedded into the CAPTCHA image. Unlike the ChCaptcha1 scheme, the TAN is not transaction dependent. The TAN digits are mixed up with the transaction data as the birthday in the GeCaptcha scheme. Three different font faces and two different font styles (normal and italic) are used to render the TAN digits. The user is asked to input the transaction password<sup>9</sup> and the TAN to confirm the transaction data. The colors of the TAN digits and the transaction data remain relatively stable, but the color of the background is not uniform and differs for different CAPTCHA images. Some short strokes are added as decoy objects. The locations of the transaction data remain unchanged. Figure 20 shows a ChCaptcha2 image that we collected from a real bank account.



Figure 20: 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”.

### 5.2.2 Automated attack based on image inpainting

Due to its similarity to GeCaptcha, the automated attack based on image inpainting can also be applied to manipulate the transaction data in ChCaptcha2 images. Since there is no random grid involved, we can directly use the fast image inpainting technique in [15] to remove the transaction data and add forged ones to the inpainted image. The segmentation of the transaction data can still be achieved via  $k$ -means clustering since the color of the transaction data is different from other parts of the image. Figure 21 shows the result of this attack on the real ChCaptcha2 image in Fig. 20. Comparing the two images, we can see the forged image is not perfect, but the difference is small enough to spoof users. We tested the attack on 103 ChCaptcha2 images and it works for all of them, thus leading to a success rate of 100%. The efficiency of the inpainting based attack is very high. The average running time of our implementation is around 200 ms.



Figure 21: An image forged from the ChCaptcha2 image in Fig. 20 with the image inpainting based attack.

### 5.2.3 Automated attack based on TAN recognition

In addition to the above attack based on image inpainting, the malicious program can also try to recognize the TAN embedded in the CAPTCHA image. Then, it will be able to confirm a manipulated transaction without getting the TAN from the user. If the user is using a static transaction password, the malicious program can steal the transaction password and automate the whole process.

Again, the first step of TAN recognition is to segment all the TAN digits from the CAPTCHA image. Due to the non-uniformity and the dynamic color of the background, the  $k$ -means clustering algorithm does not work well if we apply it to the whole image. By horizontally dividing the image into five different parts of equal size and applying the  $k$ -means clustering algorithm to each part separately, this problem can be solved. Figure 22(a) shows the segmented layers from the ChCaptcha2 image in Fig. 20. We then apply the image inpainting algorithm to remove the text layer and perform a 2-means clustering process to extract the layer of TAN digits. Figure 22(b) shows the final segmentation result of the five TAN digits.

<sup>9</sup>Depending on the user’s choice, the transaction password can be either static or dynamic (i.e., one-time password on an OTP card or sent to the user’s mobile phone via SMS).

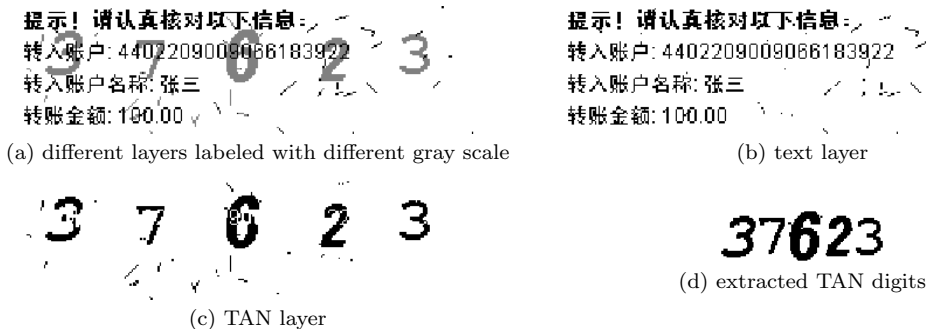


Figure 22: The segmented result of the ChCaptcha2 image in Fig. 20.

After segmenting the TAN digits, we can use CW-SSIM or correlation based methods to recognize them. Here, the number of reference images is  $10 \times 3 \times 2 = 60$ . For some ChCaptcha2 images, not all decoy strokes can be completely removed, so the correlation based recognition method does not work very well. For 103 ChCaptcha2 images, the success rate is around 24%. The CW-SSIM based recognition method achieves the ideal success rate of 100%.

As we mentioned before, the efficiency of the CW-SSIM based recognition method is relatively low. The average running time is about 7–8 seconds. Note, however, that the malicious program does not need to respond to the server in real time since the CAPTCHA images are supposed to be solved by human users who can be very slow.

## 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 [36,38], 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 all login CAPTCHA schemes. 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. For a few CAPTCHA schemes, sometimes (with a low probability) two adjacent characters are connected so that they cannot be segmented as two separate characters. Such connected characters cannot be distinguished by the segmentation attack alone, but they can be detected by the subsequent character recognition step. Therefore, we do not count these cases as failures of the segmentation attacks.

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

<i>Financial institution(s)/e-banking login CAPTCHA scheme</i>	<i>CAPTCHA image(s)</i>	<i>Segmentation result(s)</i>	<i>Tool(s) used, Weakness(es) exploited</i>	<i>Success rate</i>
13 German banks		<b>534261</b>	3-means clustering, morphological operations	60/60 = 100%
Hundreds other German banks		<b>Q4B254</b>	2-means clustering, line detection, image inpainting	60/60 = 100%
A Swiss bank with branches in Europe, Asia, North America and Africa (a Pakistani bank is also using the same system)		<b>49575</b>	2-means clustering	60/60 = 100%



Table 1: (continued)






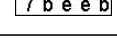

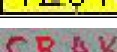
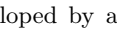
A bank based in Latin America with branches in Europe, Asia, Australasia and Africa		<b>31289 86773 53975 16391</b>	2-means clustering	63/63 = 100%
Old CAPTCHA scheme of the above bank		<b>METM</b>	2-means clustering	60/60 = 100%
US e-banking CAPTCHA 1*		<b>454e</b>	2/3-means clustering, line detection, image inpainting	209/209 = 100%
US e-banking CAPTCHA 2*		<b>79249</b>	4-means clustering, morphological operations	71/71 = 100%
US e-banking CAPTCHA 3*		<b>8V5R6</b>	morphological operations, static color of foreground	60/60 = 100%
US e-banking CAPTCHA 4*		<b>b6t3r</b>	3-means clustering	115/115 = 100%
US e-banking CAPTCHA 5*		<b>b49bs</b>	3-means clustering	92/92 = 100%
US e-banking CAPTCHA 6*		<b>2mbr5</b>	2-means clustering	61/61 = 100%
US e-banking CAPTCHA 7*		<b>3264</b>	2/3-means clustering	60/60 = 100%
A CU in Australia		<b>bQUM</b>	3-means clustering, morphological operations	60/60 = 100%
Another Two CUs in Australia		<b>eiFNtr</b>	3-means clustering, morphological operations	60/61 $\approx$ 98.4%
Yet another CU in Australia		<b>AYDG</b>	3-means clustering, morphological operations	60/60 = 100%
Chinese e-banking CAPTCHA 1		<b>3571</b>	3-means clustering	60/60 = 100%
Chinese e-banking CAPTCHA 2		<b>9YBZ</b>	2-means clustering, image inpainting, static color of noises	60/60 = 100%
Chinese e-banking CAPTCHA 3		<b>88071</b>	4-means clustering, morphological opening	62/62 = 100%
Chinese e-banking CAPTCHA 4		<b>WLV3</b>	morphological cleaning, character intensity < 120	61/61 = 100%
Chinese e-banking CAPTCHA 5		<b>byj9</b>	3-means clustering, morphological cleaning	59/60 $\approx$ 98.3%
Chinese e-banking CAPTCHA 6		<b>1694</b>	static colors of foreground, background and noises	61/61 = 100%
Chinese e-banking CAPTCHA 7		<b>3m9nXn</b>	grayscale foreground vs. colored noises	60/60 = 100%
Chinese e-banking CAPTCHA 8		<b>7582</b>	2-means clustering	60/60 = 100%
Chinese e-banking CAPTCHA 9		<b>6769</b>	2-means clustering, morphological opening	61/61 = 100%
Chinese e-banking CAPTCHA 10		<b>0369</b>	2-means clustering	60/60 = 100%
Chinese e-banking CAPTCHA 11		<b>smib</b>	2-means clustering	64/64 = 100%
Chinese e-banking CAPTCHA 12		<b>2595</b>	segmentation labels available as color indices	64/64 = 100%
Chinese e-banking CAPTCHA 13		<b>xjva3</b>	3-means clustering	60/60 = 100%



Table 1: (continued)

Chinese e-banking CAPTCHA 14		<b>w4na</b>	morphological operation, foreground intensity always darker than 128	60/60 = 100%
Chinese e-banking CAPTCHA 15		<b>Q8H6</b>	2-means clustering	110/110 = 100%
Chinese e-banking CAPTCHA 16		<b>MkJ9</b>	2/3-means clustering	60/60 = 100%
Chinese e-banking CAPTCHA 17		<b>14aa</b>	2-means clustering	62/62 = 100%
Chinese e-banking CAPTCHA 18		<b>mj4k</b>	2-means clustering, morphological operations	58/61 $\approx$ 95.1%
Chinese e-banking CAPTCHA 19 (deployed by two banks)		<b>0591</b>	3-means clustering, morphological filling	63/63 = 100%
Chinese e-banking CAPTCHA 20		<b>7beeb</b>	2-means clustering	60/60 = 100%
Chinese e-banking CAPTCHA 21 (deployed by four banks)		<b>10wzjh</b>	3-means clustering	61/61 = 100%
Chinese e-banking CAPTCHA 22		<b>5fbc</b>	2-means clustering	60/60 = 100%
Chinese e-banking CAPTCHA 23		<b>55624</b>	2-means clustering	60/60 = 100%
Chinese e-banking CAPTCHA 24		<b>s6dn</b>	static character color (white)	60/60 = 100%
Chinese e-banking CAPTCHA 25		<b>1287</b>	static character color (black)	60/60 = 100%
Chinese e-banking CAPTCHA 26		<b>CRAK</b>	clearly separable background and foreground	60/60 = 100%

\* All these CAPTCHA schemes are developed by an e-banking service provider in US, which serves several thousand American financial institutions.

## 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. Figure 23 shows the results for three JPEG compressed editions of the GeCaptcha image in Fig. 3. 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 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.

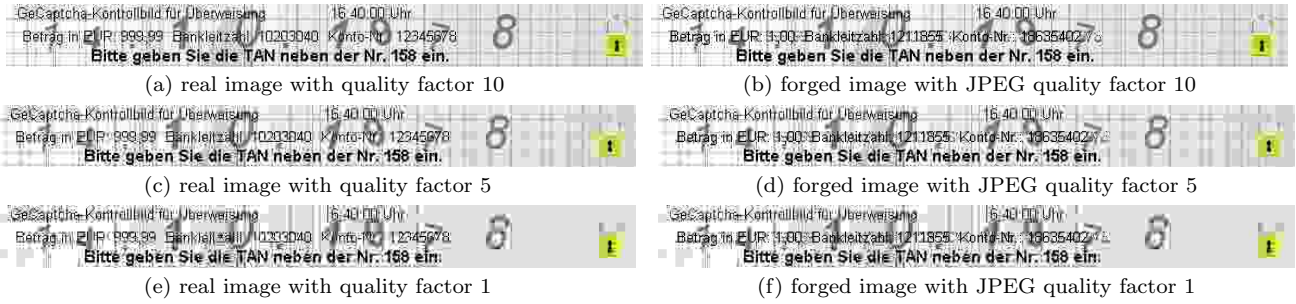


Figure 23: Three forged images when the GeCaptcha scheme is enhanced by lossy JPEG compression. The transaction data involved are the same those used in previous figures.

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 as different clusters are not well separated. For an enhanced GeCaptcha image shown in Fig. 24, none of our proposed attacks works.

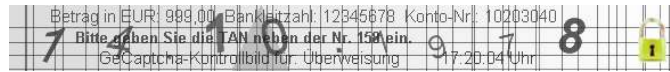


Figure 24: An enhanced GeCaptcha image in which different foreground layers have similar gray values.

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 [60], 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 (font face, font style, rotation, and different kinds of distortions). 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. 24. 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 25 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.

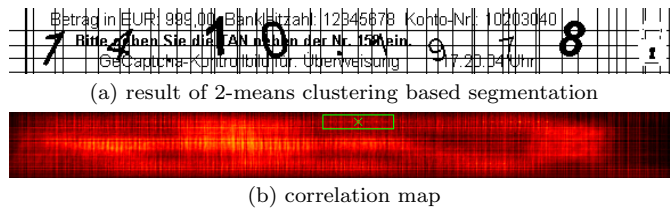


Figure 25: The result of searching for the account number “12345678” in Fig. 24. The green rectangle shows the location with the maximal correlation, which is exactly the location of the account number.

The template searching based attack is very powerful. It 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 and thus slow down the CAPTCHA generation process. 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. Note that the template searching

attack can also be generalized to break other CAPTCHA schemes. Similar ideas have been suggested in [32,46].

## 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 (i.e., the underlying AI problem) 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 [61,62]. 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 [63]. 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 [64,65]. 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 [66,67].

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. We are not the only people worrying about unsuitability of CAPTCHAs for long-term security applications. In [68], 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. Many financial institutions have already deployed different kinds of hardware security tokens, from simple OTP generator to smart card readers with trusted display and keypad. 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 of the hardware device.

## 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 Zukunftscolleg (“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.

## References

- [1] American Bankers Association. ABA survey: Consumers prefer online banking. <http://www.aba.com/Press+Room/092109ConsumerSurveyPBM.htm>, 2009.
- [2] Markus Jakobsson and Steven Myers, editors. *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. John Wiley & Sons, Inc., 2007.
- [3] P. Gühring. Concepts against man-in-the-browser attacks. <http://www2.futureware.at/svn/sourcerer/CAcert/SecureClient.pdf>, 2007.
- [4] RedTeam Pentesting GmbH. New banking security system iTAN not as secure as claimed. <http://www.redteam-pentesting.de/advisories/rt-sa-2005-014>, 2005.
- [5] Anti-Phishing Working Group. Phishing activity trends report, 4th quarter / 2009. [http://www.antiphishing.org/reports/apwg\\_report\\_Q4\\_2009.pdf](http://www.antiphishing.org/reports/apwg_report_Q4_2009.pdf), 2010.
- [6] Bank Austria. mobileTAN information. <http://www.bankaustria.at/de/19741.html>, 2007.
- [7] Postbank. mTAN now free for all customers. [http://www.postbank.com/pbcom\\_ag\\_home/pbcom\\_pr\\_press/pbcom\\_pr\\_press\\_archives/pbcom\\_pr\\_press\\_archives\\_2008/pbcom\\_pr\\_pm1063\\_19\\_05\\_08.html](http://www.postbank.com/pbcom_ag_home/pbcom_pr_press/pbcom_pr_press_archives/pbcom_pr_press_archives_2008/pbcom_pr_pm1063_19_05_08.html), 2008.
- [8] Cronto Limited. Cronto’s visual cryptogram: A simple portable solution for enhanced security. [http://www.cronto.com/visual\\_cryptogram.htm](http://www.cronto.com/visual_cryptogram.htm), 2008.
- [9] Axsionics. Personal AXS-token. <http://www.axsionics.ch/tce/frame/main/414.htm>, 2009.
- [10] CEN (Comité Européen de Normalisation, European Committee for Standardization). Financial transactional IC card reader (FINREAD). CEN Workshop Agreements (CWA) 14174, 2004.
- [11] 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.
- [12] Thomas Weigold, Thorsten Kramp, Reto Hermann, Frank Höring, Peter Buhler, and Michael Baentsch. The Zurich Trusted Information Channel – An efficient defence against man-in-the-middle and malicious software attacks. In *Trusted Computing – Challenges and Applications (Proc. TRUST’2008)*, volume 4968 of *Lecture Notes in Computer Science*, pages 75–91. Springer, 2008.
- [13] G. A. F. Seber. *Multivariate Observations*. John Wiley & Sons, Inc., 2004.
- [14] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics (SIGGRAPH’2000)*, pages 417–424. ACM, 2000.
- [15] Manuel M. Oliveira, Brian M. Bowen, Richard McKenna, and Yu-Sung Chang. Fast digital image inpainting. In *Proceedings of IASTED International Conference on Visualization, Imaging, and Image Processing (VII’2001)*, pages 261–266, 2001.
- [16] F. Y. Shin. *Image Processing and Mathematical Morphology: Fundamentals and Applications*. CRC, 2009.
- [17] Sophocles J. Orfanidis. *Optimum Signal Processing*. 2 edition, 2007. <http://www.ece.rutgers.edu/~orfanidi/osp2e>.
- [18] Zhou Wang and Eero P. Simoncelli. Translation insensitive image similarity in complex wavelet domain. In *Proceedings of 2005 IEEE International Conference on Acoustics, Speech & Signal Processing (ICASSP’2005)*, volume 2, pages 573–576. IEEE, 2005.

- [19] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: Using hard AI problems for security. In *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311. Springer, 2003.
- [20] M. Naor. Verification of a human in the loop or identification via the Turing test. <http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.pdf>, 1996.
- [21] M. D. Lillibridge, M. Abadi, K. Bharat, and A. Z. Broder. Method for selectively restricting access to computer systems. US Patent 6195698, 2001.
- [22] Jeremy Elson, John R. Douceur, Jon Howell, and Jared Saul. Asirra: A CAPTCHA that exploits interest-aligned manual image categorization. In *Proceedings of 14th ACM Conference on Computer and Communications Security (CCS'2007)*, pages 366–374. ACM, 2007.
- [23] Elias Athanasopoulos and Spiros Antonatos. Enhanced CAPTCHAs: Using animation to tell humans and computers apart. In *Communications and Multimedia Security: 10th IFIP TC-6 TC-11 International Conference, CMS 2006, Heraklion, Crete, Greece, October 19-21, 2006. Proceedings*, volume 4237 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 2006.
- [24] Niloy J. Mitra, Hung-Kuo Chu, Tong-Yee Lee, Lior Wolf, Hez Yeshurun, and Daniel Cohen-Or. Emerging images. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia'2009)*, 28(5):Article No. 163, 2009.
- [25] Kurt Alfred Kluever and Richard Zanibbi. Balancing usability and security in a video CAPTCHA. In *Proceedings of 4th Symposium on Usable Privacy and Security (SOUPS'2009)*, page Article No. 14. ACM, 2009.
- [26] Mohammed E. Hoque, David J. Russomanno, and Mohammed Yeasin. 2D Captchas from 3D models. In *Proceedings of the IEEE SoutheastCon 2006*, pages 165–170, 2006.
- [27] T. Hayward. 3D images as a CAPTCHA. <http://taylorhayward.posterous.com/3d-images-as-a-captcha>, 2009.
- [28] M. G. Kaplan. The 3-D CAPTCHA. <http://spamfizzle.com/CAPTCHA.aspx>.
- [29] 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.
- [30] Greg Mori and Jitendra Malik. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2003)*, volume 1, pages 134–141. IEEE Computer Society, 2003.
- [31] Gabriel Moy, Nathan Jones, Curt Harkless, and Randall Potter. Distortion estimation techniques in solving visual CAPTCHAs. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*, volume 2, pages 23–28. IEEE Computer Society, 2004.
- [32] Kumar Chellapilla and Patrice Y. Simard. Using machine learning to break visual Human Interaction Proofs (HIPs). In *Advances in Neural Information Processing Systems 17 (NIPS'2004)*, pages 265–272. MIT Press, 2005.
- [33] Zhenqiu Zhang, Yong Rui, Thomas Huang, and Cem Paya. Breaking the Clock Face HIP. In *Proceedings of 2004 IEEE International Conference on Multimedia and Expo (ICME)*, volume 3, pages 2167–2170. IEEE, 2004.
- [34] S. Hocevar. PWNtcha: Pretend we're not a Turing computer but a human antagonist. <http://caca.zoy.org/wiki/PWNtcha>, 2004.
- [35] E. Aboufadel, J. Olsen, and J. Windle. Breaking the Holiday Inn Priority Club CAPTCHA. *The College Mathematics Journal*, 36(2):101–108, 2005.
- [36] Kumar Chellapilla, Kevin Larson, Patrice Simard, and Mary Czerwinski. Computers beat humans at single character recognition in reading based human interaction proofs (HIPs). In *Proceedings of the 2nd Conference on Email and Anti-Spam (CEAS'2005)*, 2005.

- [37] Jeff Yan and Ahmad Salah El Ahmad. Breaking visual CAPTCHAs with naïve pattern recognition algorithms. In *Proceedings of 23rd Annual Computer Security Applications Conference (ACSAC'2007)*, pages 279–291. IEEE Computer Society, 2007.
- [38] Jeff Yan and Ahmad Salah El Ahmad. A low-cost attack on a Microsoft CAPTCHA. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'2008)*, pages 543–554. ACM, 2008.
- [39] K. A. Kluever. Breaking the PayPal HIP: A comparison of classifiers. <https://ritdml.rit.edu/handle/1850/7813>, 2008.
- [40] Philippe Golle. Machine learning attacks against the Asirra CAPTCHA. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'2008)*, pages 535–542. ACM, 2008.
- [41] Jennifer Tam, Jiri Simsa, Sean Hyde, and Luis von Ahn. Breaking audio CAPTCHAs. In *Advances in Neural Information Processing Systems 21 (NIPS'2008)*, pages 1625–1632. MIT Press, 2009.
- [42] Elie Bursztein and Steven Bethard. Decaptcha: Breaking 75% of eBay audio CAPTCHAs. In *Proceedings of 3rd USENIX Workshop on Offensive Technologies (WOOT'2009)*. USENIX, 2009.
- [43] 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.
- [44] C. J. Hernandez-Castro, A. Ribagorda, and Y. Saez. Side-channel attack on labeling CAPTCHAs. <http://arxiv.org/abs/0908.1185>, 2009.
- [45] Allan Caine and Urs Hengartner. The AI hardness of CAPTCHAs does not imply robust network security. In *Trust Management: Proceedings of IFIPTM 2007: Joint iTrust and PST Conferences on Privacy, Trust Management and Security, July 30-August 2, 2007, New Brunswick, Canada*, volume 238 of *IFIP International Federation for Information Processing*, pages 367–382. Springer, 2007.
- [46] Abram Hindle, Michael W. Godfrey, and Richard C. Holt. Reverse engineering CAPTCHAs. In *Proceedings of 15th Working Conference on Reverse Engineering (WCRE'2009)*, pages 59–68. IEEE Computer Society, 2009.
- [47] Carlos Javier Hernandez-Castro and Arturo Ribagorda. Remotely telling humans and computers apart: An unsolved problem. In J. Camenisch and D. Kesdogan, editors, *iNetSec 2009 – Open Research Problems in Network Security: IFIP WG 11.4 International Workshop, Zurich, Switzerland, April 23-24, 2009, Revised Selected Papers*, volume 309 of *IFIP Advances in Information and Communication Technology*. Springer, 2009.
- [48] Benny Pinkas and Tomas Sander. Securing passwords against dictionary attacks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'2002)*, pages 161–170, 2002.
- [49] Chris J. Mitchell. Using human interactive proofs to secure human-machine interactions via untrusted intermediaries. In *Security Protocols: 14th International Workshop, Cambridge, UK, March 27-29, 2006, Revised Selected Papers*, volume 5087 of *Lecture Notes in Computer Science*, pages 164–170. Springer, 2009.
- [50] Igor Fischer and Thorsten Herfet. Visual CAPTCHAs for document authentication. In *Proceedings of IEEE 8th Workshop on Multimedia Signal Processing (MMSP'2006)*, pages 471–474. IEEE, 2006.
- [51] Martin Szydłowski, Christopher Kruegel, and Engin Kirda. Secure input for Web applications. In *Proceedings of 23rd Annual Computer Security Applications Conference (ACSAC'2007)*, pages 375–384. IEEE Computer Society, 2007.
- [52] D. J. Steeves and M. W. Snyder. Secure online transactions using a CAPTCHA image as a watermark. US Patent 2007/0005500, 2007.
- [53] Chun-Ming Leung. Depress phishing by CAPTCHA with OTP. In *Proceedings of 3rd IEEE International Conference on Anti-counterfeiting, Security, and Identification in Communication (ASID'2009)*, pages 187–192. IEEE, 2009.
- [54] W. Wieser. Captcha recognition via averaging. <http://www.triplespark.net/misc/captcha>, 2007.

- [55] Chun-Ming Leung. Visual security is feeble for anti-phishing. In *Proceedings of 3rd IEEE International Conference on Anti-counterfeiting, Security, and Identification in Communication (ASID'2009)*, pages 118–123. IEEE, 2009.
- [56] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice Hall, 3 edition, 2008.
- [57] A. Criminisi, P. Pérez, and K. Toyama. Object removal by exemplar-based inpainting. In *Proceedings of 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2003)*, volume 2, pages 721–728. IEEE Computer Society, 2003.
- [58] P. Getreuer. tvreg: Variational imaging methods for denoising, deconvolution, inpainting, and segmentation. <http://www.math.ucla.edu/~getreuer/tvreg.html>, 2009.
- [59] J. Schulenburg. GOCR: open-source character recognition. <http://jocr.sourceforge.net>, 2009.
- [60] D. H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.
- [61] J. Wilkins. Strong CAPTCHA guidelines: v1.2. <http://bitland.net/captcha.pdf>, December 2009.
- [62] Chad W. Houck. Decoding reCAPTCHA. presented at DEF CON 18 Hacking Conference, <http://n3on.org/projects/reCAPTCHA/docs/reCAPTCHA.docx>, 2010.
- [63] Jeff Yan and Ahmad Salah El Ahmad. Usability of CAPTCHAs or usability issues in CAPTCHA design. In *Proceedings of 4th Symposium On Usable Privacy and Security (SOUPS'2008)*, pages 44–52. ACM, 2008.
- [64] Marti Motoyama, Kirill Levchenko, Chris Kanich, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. Re: CAPTCHAs – Understanding CAPTCHA-solving services in an economic context. In *Proceedings of 19th USENIX Security Symposium*, 2010.
- [65] Elie Bursztein, Steven Bethard, Celine Fabry, John C. Mitchell, and Dan Jurafsky. How good are humans at solving CAPTCHAs? A large scale evaluation. In *Proceedings of 31st IEEE Symposium on Security & Privacy (S&P'2010)*, 2010.
- [66] BBC. PC stripper helps spam to spread. <http://news.bbc.co.uk/2/hi/technology/7067962.stm>, 2007.
- [67] J. Baltazar, J. Costoya, and R. Flores. The real face of KOOFACE: The largest Web 2.0 botnet explained. [http://us.trendmicro.com/imperia/md/content/us/trendwatch/researchandanalysis/the\\_real\\_face\\_of\\_koobface\\_jul2009.pdf](http://us.trendmicro.com/imperia/md/content/us/trendwatch/researchandanalysis/the_real_face_of_koobface_jul2009.pdf), 2009.
- [68] Markus Jakobsson. CAPTCHA-free throttling. In *Proceedings of the 2nd ACM Workshop on Security and Artificial Intelligence (AISec'2009)*, pages 15–21. ACM, 2009.