UNIVERSITY
OF WOLLONGONG
AUSTRALIA

University of Wollongong
Research Online

Faculty of Engineering and Information Sciences - Papers: Part A

Faculty of Engineering and Information Sciences

2014

On the security of text-based 3D CAPTCHAs

Vu Duc Nguyen
University of Wollongong, dvn108@uowmail.edu.au

Yang-Wai Chow
University of Wollongong, caseyc@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Publication Details

Nguyen, V., Chow, Y. & Susilo, W. (2014). On the security of text-based 3D CAPTCHAs. Computers and Security, 45 (September), 84-99.

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

On the security of text-based 3D CAPTCHAs

Abstract
CAPTCHAs have become a standard security mechanism that are used to deter automated abuse of online services intended for humans. However, many existing CAPTCHA schemes to date have been successfully broken. As such, a number of CAPTCHA developers have explored alternative methods of designing CAPTCHAs. 3D CAPTCHAs is a design alternative that has been proposed to overcome the limitations of traditional CAPTCHAs. These CAPTCHAs are designed to capitalize on the human visual system's natural ability to perceive 3D objects from an image. The underlying security assumption is that it is difficult for a computer program to identify the 3D content. This paper investigates the robustness of text-based 3D CAPTCHAs. In particular, we examine three existing text-based 3D CAPTCHA schemes that are currently deployed on a number of websites. While the direct use of Optical Character Recognition (OCR) software is unable to correctly solve these textbased 3D CAPTCHA challenges, we highlight certain patterns in the 3D CAPTCHAs can be exploited to identify important information within the CAPTCHA. By extracting this information, this paper demonstrates that automated attacks can be used to solve these 3D CAPTCHAs with a high degree of success.

Keywords
text, 3d, security, captchas

Disciplines
Engineering | Science and Technology Studies

Publication Details
Nguyen, V., Chow, Y. & Susilo, W. (2014). On the security of text-based 3D CAPTCHAs. Computers and Security, 45 (September), 84-99.

This journal article is available at Research Online: http://ro.uow.edu.au/eispapers/3072

On the Security of Text-based 3D CAPTCHAs

Vu Duc Nguyen, Yang-Wai Chow, Willy Susilo*

School of Computer Science and Software Engineering, University of Wollongong, NSW 2522, Australia

Abstract

CAPTCHAs have become a standard security mechanism that are used to deter automated abuse of online services intended for humans. However, many existing CAPTCHA schemes to date have been successfully broken. As such, a number of CAPTCHA developers have explored alternative methods of designing CAPTCHAs. 3D CAPTCHAs is a design alternative that has been proposed to overcome the limitations of traditional CAPTCHAs. These CAPTCHAs are designed to capitalize on the human visual system's natural ability to perceive 3D objects from an image. The underlying security assumption is that it is difficult for a computer program to identify the 3D content. This paper investigates the robustness of text-based 3D CAPTCHAs. In particular, we examine three existing text-based 3D CAPTCHA schemes that are currently deployed on a number of websites. While the direct use of Optical Character Recognition (OCR) software is unable to correctly solve these text-based 3D CAPTCHA challenges, we highlight certain patterns in the 3D CAPTCHAs can be exploited to identify important information within the CAPTCHA. By extracting this information, this paper demonstrates that automated attacks can be used to solve these 3D CAPTCHAs with a high degree of success.

Keywords: 3D CAPTCHA, character extraction, optical character recognition, automated attack, security

*Corresponding author. School of Computer Science and Software Engineering, University of Wollongong, NSW 2522, Australia. Tel.: +61 2 4221 5535; fax: +61 2 4221 5550.

Email address: vdn108@uowmail.edu.au, caseyc@uow.edu.au, wsusilo@uow.edu.au (Vu Duc Nguyen, Yang-Wai Chow, Willy Susilo)

1. Introduction

CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) are essentially automated challenge-response tests that are used to ascertain whether or not an online transaction is being carried out by a human. While the concept of 'Automated Turing Tests' has been around for some time, the term CAPTCHA was originally introduced by von Ahn et al. (2003) in their seminal work on CAPTCHAs as hard Artificial Intelligence (AI) problems that can be exploited for security purposes. Since its inception, CAPTCHAs have now become a ubiquitous part of the Internet and are used on many web-based services as a standard security mechanism to deter automated abuse of online services intended for humans. For example, it is used to deter malicious bot programs from signing up for thousands of accounts from free email services and sending out thousands of spam messages every minute (Yan and Ahmad, 2007).

Over the years, various CAPTCHA schemes have been developed and deployed on numerous web services, including online services provided by major companies such as Google, Yahoo! and Microsoft, and social networks like Facebook, to provide some level of security against online abuse (Yan and Ahmad, 2008). Of the different types of CAPTCHAs (e.g. text-based, image-based, audio-based) that are currently used in practice, 2D text-based CAPTCHAs are the most widespread form in use. The popularity and pervasiveness of this form of CAPTCHA stems from its intuitiveness and low implementation cost (Chellapilla et al., 2005b). Text-based CAPTCHAs generally consist of a sequence of characters (i.e. alphabetical letters and/or digits) that are presented within an image. The image may contain some sort of visual noise (e.g. a textured background) and the characters are typically distorted to deter automated computer attacks.

However, many of these CAPTCHA schemes have been found to be vulnerable to attacks, including the use of machine learning, computer vision, pattern recognition or other techniques (Yan and Ahmad, 2007, 2008; Ahmad et al., 2012; Nakaguro et al., 2013). In addition, Optical Character Recognition (OCR) technology is ever improving and OCR programs are becoming increasingly effective in recognizing text. As such, in order to overcome the limitations of traditional 2D text-based CAPTCHAs and to produce CAPTCHA schemes that are more secure and robust against OCR programs, some CAPTCHA developers have explored the design of CAPTCHAs based on 3D. The perceived robustness of 3D CAPTCHAs stems from the under-

lying assumption that it is difficult for computer programs to identify 3D content, while at the same time, the perception of 3D is an inherent part of the human visual system. Hence, 3D CAPTCHAs satisfy the fundamental requirement of a CAPTCHA scheme, i.e. easy for humans to solve but difficult for computer programs.

While there is much research on the security of traditional 2D text-based CAPTCHAs, the robustness of 3D CAPTCHAs has not received much attention. This paper focuses on investigating the security of text-based 3D CAPTCHAs and their robustness against automated attacks. To this end, we have identified three existing text-based 3D CAPTCHAs that are currently being used on a number of websites. These 3D CAPTCHAs are assumed to be secure as they cannot be solved by OCR programs directly. However, this paper shows that by exploiting certain features in the CAPTCHA schemes it is possible to identify and extract key information about the CAPTCHA that can subsequently be used to solve the CAPTCHA challenge with a high success rate. This highlights the fact that these text-based 3D CAPTCHAs are indeed susceptible to automated attacks and are no more secure than their 2D counterparts.

In our previous work (Nguyen et al., 2011), we demonstrated an attack on the Teabag3D CAPTCHA (OCR Research Team, 2006). In that study we focused on attacking version 1.0.1 of Teabag3D and presented some preliminary findings on attacking Teabag3D version 1.2 using the same approach. This gave rise to the possibility that other text-based 3D CAPTCHAs were also vulnerable to automated attacks by extracting key information from the CAPTCHA. In this paper, we describe our approach to attacking two other text-based 3D CAPTCHAs. In addition, this paper also presents the essential pseudocode for attacking Teabag3D, as well as updated results on attacking Teabag3D version 1.2. These attacks are based on similar principles, in that we identified variations in the regular pattern of the text-based 3D CAPTCHA in order to extract information about the characters embedded within the pattern.

The rest of this paper is organized as follows. Section 2 presents existing work in the area of CAPTCHA security and also introduces various 3D CAPTCHA schemes that have been proposed by a number of researchers. In Section 3, three representative text-based 3D CAPTCHA schemes are described along with details of our approach on how to attack each of the respective schemes. Experimental results of automated attacks conducted for each of the 3D CAPTCHA schemes are presented in Section 4. Section

5 presents a summary of the key features highlighted in our approach along with a discussion of their limitations. Finally, Section 6 concludes the paper and summarizes the main findings of this research.

2. Related Work

2.1. CAPTCHA Security

The robustness and security of CAPTCHAs has been the topic of much scrutiny by researchers and practitioners alike. To date, a number of researchers investigating the security of various CAPTCHA designs have demonstrated that many existing CAPTCHA schemes are vulnerable to automated attacks. Much of this vulnerability is due to certain flaws in the design of these CAPTCHA schemes, several of which are described in this section.

The popular Gimpy family of CAPTCHAs developed at Carnegie Mellon University has been subject to a number of automated attacks. Mori and Malik (2003) were able to successfully break the EZ-Gimpy CAPTCHA (previously used by Yahoo!) 92% of the time, as well as the Gimpy CAPTCHA at a success rate of 33%. Their work was based on matching shape contexts of characters, in the midst of a background texture, using an image database of known objects. Using the knowledge that the text in this CAPTCHA scheme was based on a set of English words, they then proceeded by ranking a set of candidate words and selecting the one with the best matching score. They also demonstrated a holistic approach of recognizing entire words at once, instead of attempting to identify individual characters. This was because in severe clutter, attempting to identify characters itself was often not enough as parts of characters could be occluded or ambiguous (Mori and Malik, 2003). Among other things, this work highlights the fact that CAPTCHAs based on language models are susceptible to dictionary attacks.

In fact, with full knowledge of font and lexicon, the Mori-Malik attack also produced reasonably high success rates in solving two other CAPTCHAs schemes; namely, PessimalPrint (Baird et al., 2003) and BaffleText (Chew and Baird, 2003). Both of these pioneering CAPTCHAs were designed in the research community, and represent research effort exploring the question of how to design text-based CAPTCHAs properly (Yan and Ahmad, 2009).

Chellapilla and Simard (2004) demonstrated that machine learning algorithms could be used to break a variety of CAPTCHAs (or Human Interaction Proofs (HIPs)). In their work, they deliberately avoided exploiting language models to break these CAPTCHAs. The aim was to develop a

generic method that could automate the task of segmentation (i.e. finding the characters), thus reducing the challenge to a pure recognition problem which is a trivial task using machine learning. This work, by the research team in Microsoft (Chellapilla et al., 2005a), has led to the segmentation-resistant principle that is now widely accepted as a requirement in the design of more secure text-based CAPTCHAs (Ahmad et al., 2010).

Following on from their work, the team developed a well thought out CAPTCHA scheme that was deployed on a number of Microsoft's online services. While this CAPTCHA was meant to be segmentation-resistant, it was unfortunately shown to be susceptible to a low-cost attack (Yan and Ahmad, 2008). Among the lessons to be learnt from this work, is that it becomes easier to segment a CAPTCHA in which the total number of characters is known, or can be ascertained, a priori. Nonetheless, despite breaking the CAPTCHA, Yan and Ahmad (2008) pointed out that their attack did not overturn or negate the segmentation-resistant principle. Instead, upon closer examination certain CAPTCHAs that are designed to be segmentation-resistant, can actually be segmented after some pre-processing (Ahmad et al., 2010).

Yan and Ahmad (2007, 2009, 2011) also showed that a number of other CAPTCHAs could be defeated using novel attacks like pixel-count attacks, where characters could be distinguished by simply counting the number of pixels that constituted each individual character. Their work emphasized that in addition to segmentation-resistance, it is good practice to use local and global warping to distort characters in CAPTCHAs. Evidently, local and global distortions alone are not sufficient to deter effective attacks. Moy et al. (2004) demonstrated breaking EZ-Gimpy and Gimpy-r using distortion estimation techniques. The first step in their approach involved background removal, to separate the text from the background clutter without losing important information. This is also a step that many other attacks employ. Thus, the importance of making it hard to separate the text from the background is also highlighted as a factor that has to be considered when designing secure CAPTCHAs (Yan and Ahmad, 2009).

In a systematic study regarding the strengths and weaknesses of text-based CAPTCHAs, Bursztein et al. (2011b) observed that the segmentation-resistant principle alone is not enough to guarantee that a CAPTCHA scheme is secure against automated attacks. In other recent work, Li et al. (2010) demonstrated the use of image processing and pattern recognition algorithms, such as k-means clustering, digital image in-painting, character recognition

based on cross-correlation, etc. to successfully break a variety of e-Banking CAPTCHAs. The popular Google reCAPTCHA has also been broken using a holistic approach of recognizing shape contexts of entire words (Baecher et al., 2011) and an approach using heuristic character segmentation and recognition (Cruz-Perez et al., 2012). Furthermore, Gao et al. (2013) analyzed and discussed how simple attacks could be used to break a number of hollow CAPTCHAs, thus casting doubt on the viability of current hollow CAPTCHA designs.

While this paper focuses on text-based CAPTCHAs, other categories of CAPTCHAs are by no means immune to automated attacks. For example, an overview of attacks against a number of image-based CAPTCHAs can be found in Zhu et al. (2010), Hernandez-Castro and Ribagorda (2010) describe their attack against the 'Math CAPTCHA' and others have also worked on breaking audio-CAPTCHAs (Tam et al., 2008; Bursztein et al., 2011a). In addition, techniques for breaking CAPTCHAs based on moving objects are presented in Xu et al. (2012) and other animated CAPTCHAs have been broken in Nguyen et al. (2012a,b).

2.2. 3D CAPTCHAs

A number of attempts at designing and developing 3D CAPTCHAs have recently emerged in literature and in practice. These approaches typically generate CAPTCHA challenges by rendering 3D models of text-objects or of other objects.

Kaplan (n.d.) proposed a 3D CAPTCHA approach based on identifying labelled parts of 3D models. However, Ross et al. (2010) pointed out that this approach is unlikely to scale due to the manual effort involved in modelling and labelling parts. The social networking site YUNiTi (2013) adopts a CAPTCHA that uses Lambertian renderings of 3D models. Users are presented with an image containing 3D objects and are required to select matching objects, in the sequence that they appear in the CAPTCHA, from a provided set of images. The 3D objects in the CAPTCHA are rendered using different parameters (e.g. different orientation and colour) from those in the selection set. Unfortunately, this approach is likely to be susceptible to attacks using basic computer vision techniques (Ross et al., 2010).

The same method of attack applies to the approach proposed by Imsamai and Phimoltares (2010). In their paper, they presented a number of 3D CAPTCHA scheme variants based on renderings of 3D text-objects. It can be seen that the characters in their approach do not undergo any form

of distortion and, more importantly, the entire front face of characters are rendered using the same shade. Moreover, 3D object recognition is a well studied field, for example, Mian et al. (2006) presented an approach to viewpoint independent object recognition and segmentation of 3D model-based objects in cluttered scenes. It is possible that attacks adopting such computer vision techniques will be able to successfully defeat these 3D CAPTCHAs.

Among 3D CAPTCHA ideas that have been proposed in the research community, Mitra et al. (2009) proposed a technique of generating 'emerging images' by rendering extremely abstract representations of 3D models placed in 3D environments. This approach is based on 'emergence', the unique human ability to perceive objects in an image not by recognizing the object parts, but as a whole (Mitra et al., 2009). Ross et al. (2010) presented a pilot usability study and security analysis of a prototype implementation of their CAPTCHA called 'Sketcha'. Sketcha is based on oriented line drawings of 3D models and the user's task is to correctly orient images containing these 3D model line drawings.

3. Our Approach

To investigate the security of text-based 3D CAPTCHAs, three representative 3D CAPTCHA schemes were identified for the purpose of this study. The reason for selecting these particular 3D CAPTCHA schemes is primarily due to the fact that they are currently being used on several websites. As such, a large number of samples could be obtained from these websites. Other proposed 3D CAPTCHA schemes have a limited number of prototypes, as they are merely ideas for 3D CAPTCHAs that have not been fully developed or used in practice.

The 3D CAPTCHA schemes investigated in this study were essentially built from similar concepts, which involved perturbing a regular pattern in order to give rise to the perception of 3D text embedded within the pattern. While this approach is purported to be difficult for computers to solve, its human usability stems from the human cognitive ability to perceive 3D from cues present in a 2D image.

In general, the process that was taken to attack these CAPTCHA schemes can be divided into a number of stages. An overview of these stages is depicted in Fig. 1. The purpose of the pre-processing stage is to convert the image into a form that is suitable for character extraction, while the post-processing stage focuses on preparing the image for character recognition.

This is undertaken to improve the accuracy of the character recognition process. For example, removing impediments and noise from the image before passing it to the character recognition stage can increase the success rate of accurate recognition.

A description of each of the text-based 3D CAPTCHA schemes along a method of attacking the respective scheme is presented in the subsections to follow.

3.1. Super CAPTCHA

Super CAPTCHA is listed as a product of Goldsboro Web Development (2013). While previous versions of Super CAPTCHA were text-based 2D schemes, the scheme became a text-based 3D CAPTCHA as of version 2.3.0. At the time of writing, the latest version is version 2.4.2 and it is available as a plug-in for WordPress.org (2013), which is a popular online blogging and content management system. Consequently, this is the version that was used for this research.

The developers state that this type of 3D CAPTCHA requires human intervention by forcing the use of linear recognition algorithms that only the human mind can decipher (Goldsboro Web Development, 2013). In addition, the developers have tested its robustness against 37 different OCR programs (WordPress.org, 2013). Examples of the Super CAPTCHA scheme can be seen in Fig. 2. Fig. 2(a) is an example of the CAPTCHA with a plain background, whereas Fig. 2(b) shows an example that uses a textured background. As can be seen from the examples, for different CAPTCHAs the angle of the CAPTCHA challenge is slanted at different angles.

To attack this CAPTCHA, the pre-processing stage simply involves removing any background texture, if present, and converting the image into a black-and-white image. This can be done easily as the lines forming the CAPTCHA challenge are always rendered in a color that is distinct from the background texture. An illustration of this process is shown in Fig. 3, where Fig. 3(a) shows the original image and Fig. 3(b) shows the black-and-white image after background removal and binarization. The background removal and binarization process simply involves defining a color intensity threshold, and any pixel that has a color above this threshold are converted to white, i.e. the lines forming the CAPTCHA challenge, whereas pixels with colors below the threshold are converted to black.

After binarizing the CAPTCHA into a black-and-white image, the next stage involves extracting the text from the white lines in the image. A

human can perceive the text due to perturbations made to the lines. This also means that for a computer to attack the CAPTCHA, the perturbed sections of the lines can be separated from the linear sections. This can be done by first approximating the slope of the lines using the white corner pixels in the CAPTCHA challenge, since the slope for each CAPTCHA is different. Fig. 4 illustrates two lines that approximate the gradient of the CAPTCHA challenge. The pseudocode for approximating the gradient and obtaining the starting positions of the whites lines is provided in Algorithm 1.

Algorithm 1 Approximating the gradient and starting positions of the lines for Super CAPTCHA

```
function FindGradient(Image)
    /* Find the starting positions of the white lines */
    for all Pixels in Image do
        if Pixel = white then
            if Pixel has no neighboring white pixels on its left then
                LineStartList ← Pixel
            else if Pixel has no neighboring white pixels on its right then
                LineEndList ← Pixel
            end if
        end if
    end for
    /* Calculate the gradient from the lowest line */
    Gradient ← Calculated from the lowest pixels in LineStartList and
LineEndList
end function
```

Using the approximate gradient, parallel lines can then be produced to overwrite many of the white pixels that form the straight lines in the CAPTCHA. This is depicted in Fig. 5, where parallel lines were used to overwrite white pixels located along the straight lines. As can be seen from the figure, the parallel lines do not cover all the pixels along the straight lines. Therefore, the white pixels that are connected to, and in the direction of, the parallel lines also have to be identified. This is illustrated in Fig. 6. Once this is done, it is a simple matter of identifying potential pixels that belong to the characters. The end result of the character extraction phase is shown in Fig. 7, where one can clearly see that the remaining white pixels

are related to the characters, while the other pixels belong to the lines. Algorithm 2 depicts pseudocode for distinguishing between pixels belonging to the characters and pixels belonging to the lines.

Algorithm 2 Identifying pixels related to the characters, by using parallel lines to overwrite unrelated pixels for Super CAPTCHA

```
/* Note that Gradient and LineStartList are from Algorithm 1 */
function IdentifyCharPixels(Image, Gradient, LineStartList)
    /* Overwrite unrelated white pixels by constructing parallel lines */
    for all Pixels in LineStartList do
        Construct a line using Gradient
        for each pixel position along the constructed line do
            if Pixel in Image = white then
                Set Pixel to green
            end if
        end for
    end for
    /* Identify connected pixels in the direction of the line */
    repeat
        NoChange ← true
        for all Pixels in Image do
            if Pixel = white then
                if left or right neighbor of Pixel = green then
                    NoChange ← false
                    Set Pixel to orange
                end if
            end if
        end for
    until NoChange = true
    /* At this stage, all remaining white pixels are related to the characters,
while the other pixels belong to the lines */
end function
```

A post-processing stage was used to make the characters clearer and easier to recognize. For this, the average pixel distance, λ, between the consecutive lines was determined. Using this value, λ pixels below any character pixels were determined to belong the characters. This is depicted in Fig. 8. Additionally, pixels in large empty regions below the character pixels were

also determined as belonging to the characters, as shown in Fig. 9. The resulting image shown in Fig. 10(a) shows the pixels that were determined to constitute the characters. Fig. 10(b) shows the image after reorientation and filling in small holes.

The final stage is the character recognition stage. Character recognition can be performed by passing the post-processed image through any good OCR program. The ABBYY FineReader 11 Professional Edition (ABBYY, 2013), which is widely recognized to be one of the best OCR programs currently available on the market, was used in this research. The ABBYY FineReader uses a machine learning approach that can be trained from a training set of character samples. To improve the character recognition accuracy, a training set was created to be used in conjunction with the ABBYY FineReader's existing embedded training database.

3.2. 3dcaptcha

This is a text-based 3D CAPTCHA that is currently used on several websites (Cafe Milenium, 2009; Café Charlotte, 2006; EnterMir, 2013; SACHY-online.cz, 2010; HyperMedia, 2012). It is not known whether or not this CAPTCHA scheme was given an actual name. As such, this paper will simply refer to this CAPTCHA scheme as '3dcaptcha'. Two examples of 3dcaptcha are shown in Fig. 11. It can be seen that this CAPTCHA scheme uses diagonal lines to form a regular pattern. These diagonal lines are perturbed to give rise to the perception of 3D characters.

It should be noted that the CAPTCHA challenge for the scheme always appears within a certain region. Therefore, the pre-processing stage in attacking 3dcaptcha simply involves cropping the image to the specific region containing the challenge, as depicted in Fig. 12. This is not absolutely necessary, however, it helps to improve accuracy by potentially reducing peripheral noise in the processed image and speeds up the attack because of the reduced image size.

Unlike the lines in Super CAPTCHA where the distances between lines were relatively uniform, the distances between lines in 3dcaptcha appear to change with distance away from the viewpoint. This is because 3dcaptcha was rendered with a perspective camera in 3D space. As such, due the location and orientation of the camera where the image for 3dcaptcha was rendered, the distances between lines at the bottom of the image are further apart compared to distances between lines at the top of the image. Line distances also get closer and closer towards the right of the image.

Nevertheless, it can be seen that the distances between lines related to the characters are different from the regular pattern. Thus, key regions that are related to the characters can be identified by first calculating an average number of pixels between lines. Then by scanning the pixels in the vertical and horizontal directions, any section containing more white pixels than this average is determined to belong to a character. Results of a vertical and horizontal scan can be seen in Fig. 13(a) and Fig. 13(b) respectively. Fig. 13(c) in turn shows the combined results. The pseudocode presented in Algorithm 3 outlines the steps for performing a horizontal scan to identify pixels belonging to characters. For a vertical scan, the same steps in Algorithm 3 are performed for each column in the CAPTCHA image instead of each row.

Algorithm 3 Scanning pixels in the horizontal direction to identify pixels belonging to characters for 3dcaptcha

```
function HorizontalScan(Image)
    AvgNumOfPixels ← Average number of pixels between black pixels in the horizontal dimension
    /* Identify pixels belonging to characters */
    for all rows in Image do
        PreviousBlackPixel ← null
        for all Pixels in row do
            if Pixel = black and PreviousBlackPixel ≠ null then
                if (number of pixels between PreviousBlackPixel and Pixel) > AvgNumOfPixels then
                    TextPixelList ← all pixels between PreviousBlackPixel and Pixel
                end if
                PreviousBlackPixel ← Pixel
            end if
        end for
    end for
end function
```

In addition, since the regular pattern is made up of diagonal lines, regions related to the characters can be extracted based on changes in line direction. Fig. 14 illustrates this, where regions in between sections of lines that change direction are identified. Furthermore, perturbations to the regular pattern

result in some character boundaries having a higher pixel density. These regions can be identified based on areas with a high density of connected pixels, as depicted in Fig. 15. Finally, additional useful information can be obtained by drawing vertical lines to create a grid like pattern. An adaptive threshold is computed based on an average pixel count within the neighboring cells, and cells contain more pixels than this threshold are identified as belonging to characters. This is illustrated in Fig. 16. The pseudocode in Algorithm 4 shows the steps used to identify pixels belonging to the characters based on these changes in the regular pattern.

Fig. 17 shows an image resulting from a combination of all the key features that were identified. Before passing the image through an OCR program for character recognition, a post-processing stage was performed to reoriented the characters and to fill in small holes. Fig. 18 shows the post-processing results.

3.3. Teabag 3D

This 3D CAPTCHA scheme was designed by the OCR Research Team (2006). On their website (OCR Research Team, 2006), the team states that their aim is to break known CAPTCHAs to identify weaknesses and to create new secure CAPTCHAs. As such, their website gives a good overview of a number of text-based 2D CAPTCHAs along with their corresponding weaknesses. Additionally, some of the team's valuable experiences have been documented in Kolupaev and Ogijenko (2008). By analyzing the design flaws and weaknesses seen in traditional text-based 2D CAPTCHAs, Teabag 3D was designed to overcome these limitations.

While there are a number of versions of Teabag 3D (OCR Research Team, 2006), the research in this paper deals with the commercial version, version 1.0.1, as implemented on rediff.com (Rediff.com India, 2013), as well as version 1.2. Examples of Teabag 3D version 1.0.1 and version 1.2 are shown in Fig. 19(a) and Fig. 19(b) respectively.

As can be seen from Fig. 19(a), in Teabag 3D version 1.0.1 the CAPTCHA challenge emerges from a regular grid pattern in 3D space. There are small variations in terms of the grid direction between challenges. Teabag 3D version 1.2 is an improved version of the scheme. It can be seen from Fig. 19(b) that in this version, the grid is warped using a wave like pattern. This gives rise to greater variations in the cell sizes of the grid. In both versions of this CAPTCHA scheme, characters in close proximity may touch, thus connecting them together.

Algorithm 4 Identifying pixels that belong to the characters based on changes in the regular pattern for 3dcaptcha

```
function IdentifyTextPixels(Image, TextPixelList)
    /* Areas with changes in line direction */
    for all Pixels in Image do
        if Pixel = white then
            Perform a 4-connected flood-fill algorithm
            if number of connected pixels > 5 and (lowest connected pixel
- highest connected pixel) < threshold then
                TextPixelList ← all the connected pixels
            end if
        end if
    end for
    /* Identify areas with a high density of connected pixels */
    for all Pixels in Image do
        if Pixel = black then
            Perform a 4-connected flood-fill algorithm
            if number of connected pixels > 5 and close to pixels in
TextPixelList then
                TextPixelList ← all the connected pixels
            end if
        end if
    end for
    /* Create grid and identify cells belonging to characters */
    Construct grid pattern by drawing equidistant vertical lines
    for all Cells do
        Count the number of white pixels in the cell and its neighboring
cells
        AvgNumPixels ← Average number of pixels per cell
        if number of white pixels in cell > AvgNumPixels then
            TextPixelList ← all cell pixels
        end if
    end for
end function
```

Here, we will describe our approach to attacking Teabag 3D. Note that this can also be found in Nguyen et al. (2011). In this paper, we provide key pseudocode which outlines the steps involved in our attack. The pre-processing stage of attacking Teabag 3D involves determining the regions that contain text and separating these regions from the background grid. This can be done by identifying and extracting key features in the image. First, a binarization process is performed to convert the image into a black-and-white image. Then, the side surfaces of the characters are identified. This was done by determining the average number of white pixels per cell, and any cell containing a white pixel count above this average is identified as a side surface. Fig. 20(a) shows an example of the results obtained through this side surface identification process. The steps for performing side surface identification are shown in Algorithm 5.

Algorithm 5 Identifying the side surface pixels that belong to the text for Teabag3D

```
/* Side surface identification */
function IdentifySideSurfaces(Image)
    for all Cells in Image do
        Count the number of pixels in Cell
    end for
    AvgNumPixels ← Average number of pixels per cell
    for all Cells in Image do
        if number of pixels in Cell > AvgNumPixels then
            SideSurfacePixelList ← all pixels in Cell
        end if
    end for
end function
```

This is followed by a front surface identification process, where clusters of black pixels are determined to belong to the front surface of a character. This is because some of the borders around the characters are clearly darker than the rest of the image. Hence, if a black pixel is immediately connected to 4-neighboring black pixels (i.e. the pixel's top, down, left and right neighboring pixels are also black), then all these pixels are deemed to belong to the front surface of a character. This is illustrated in Fig. 20(b). The front surface identification process is outlined in Algorithm 6.

The pixels identified as belonging to the front and side surfaces are only

Algorithm 6 Identifying the front surface pixels that belong to the text for Teabag3D

```
/* Front surface identification */
function IdentifyFrontSurfaces(Image)
    for all Pixels in Image do
        if Pixel = black then
            if top, down, left and right pixels immediately connected to
Pixel = black then
                FrontSurfacePixelList ← Pixel and its 4-neighboring pix-
els
            end if
        end if
    end for
end function
```

the borders of the front character surfaces. To extract the full face of the text, each pixel column of the image was scanned from top to bottom. In each column, whenever a section that started with a front surface pixel and ended with either a front or side surface pixel was encountered, all the pixels in between would be filled in the as front surface of the text. Algorithm 7 shows the process of identifying full character surfaces and an example depicting the results from this process is shown in Fig. 20(c).

As can be seen from Fig. 20(c), the characters may be connected together. To improve character recognition, the text had to be segmented into their constituting characters. Segmentation was performed by trying to separate the characters using a vertical line, a diagonal line corresponding to the gradient of the grid, or by tracing the pixels along the side surfaces of the characters. Note that at this stage the side surfaces would have already been identified using the process depicted in Algorithm 5. Examples depicting these segmentation approaches are shown in Fig. 21(a), Fig. 21(b) and Fig. 21(c) respectively.

Post-processing involved reorienting the characters and refining the resulting image by removing noise and filling in small holes. This was followed by the character recognition stage, where the post-processed image was passed to an OCR program.

The processed used to attack Teabag 3D version 1.2 was similar to attacking version 1.0.1. The main difference was that since the grid was distorted

Algorithm 7 Extracting the full character faces using the front and side surface pixels for Teabag3D

/* Extract the full faces of the characters */
/* Note that $SideSurfacePixelList$ is from Algorithm 5 and $FrontSurfacePixelList$ is from Algorithm 6 */
function ExtractFullFaces(*Image, SideSurfacePixelList, FrontSurfacePixelList*)
  for all columns in $Image$ **do**
    $StartingPixel \leftarrow$ null
    for all $Pixels$ in column from top to bottom **do**
      if $StartingPixel$ = null **and** $Pixel$ is in $FrontSurfacePixelList$ **then**
        $StartingPixel \leftarrow Pixel$
      else if $StartingPixel \neq$ null **and** ($Pixel$ is in $FrontSurfacePixelList$ **or** $Pixel$ is in $SideSurfacePixelList$) **then**
        $FrontSurfacePixelList \leftarrow$ all pixels from $StartingPixel$ to $Pixel$
        $StartingPixel \leftarrow$ null
      end if
    end for
  end for
  $TextPixelList \leftarrow$ all pixels in $SideSurfacePixelList$ and $FrontSurfacePixelList$
end function

using a wave like pattern, there were greater variations in cell sizes. Thus, instead of using a fixed threshold for the side surface identification process, in which cell containing a number of pixels greater than the threshold were identified as belonging to a characters' side surface, an adaptive threshold was used. This adaptive threshold was determined by averaging the cell sizes of the neighboring cells. This can be seen in Fig. 22, where Fig. 22(a) shows an image of the original CAPTCHA challenge and Fig. 22(b) shows an image of the extracted front and side surfaces. The pseudocode for this adaptive local neighborhood averaging is depicted in Algorithm 8.

Algorithm 8 Adaptive threshold for identifying the side surface pixels for Teabag3D version 1.2

```
/* Adaptive side surface identification */
function IdentifySideSurfaces2(Image)
    for all Cells in Image do
        Count the number of pixels in the cell and its neighboring cells
        LocalAvgNumPixels ← Local neighborhood average of the number of pixels per cell
        if number of pixels in Cell > LocalAvgNumPixels then
            SideSurfacePixelList ← all pixels in Cell
        end if
    end for
end function
```

4. Results and Discussion

To test the effectiveness of our methods in attacking these text-based 3D CAPTCHAs, experiments using the techniques described in the previous section were conducted. For the experiments, 1000 samples for each of the respective CAPTCHA schemes were downloaded from actual websites. For the case of Super CAPTCHA, we collected samples from two different sources. The first was from a customer's website. In addition to this, WordPress provides a downloadable plug-in on their website (WordPress.org, 2013). Using this plug-in, another set of 1000 samples was generated using the default CAPTCHA settings. For the Teabag 3D CAPTCHA, we collected samples for each of the two versions, i.e. version 1.0.1 and version 1.2.

Table 1: Experimental results.

CAPTCHA Scheme	**Accuracy**	**Average Time**
Super CAPTCHA (from website)	27%	3 secs
Super CAPTCHA (from plug-in)	32%	3 secs
3dcaptcha	58%	4 secs
Teabag 3D version 1.0.1	76%	7 secs
Teabag 3D version 1.2	31%	4 secs

Table 1 shows the results of the CAPTCHA attacking experiments. In the table, accuracy refers to the success rate of correctly solving the entire CAPTCHA challenge, i.e. all characters successfully recognized, while average time refers to the average duration of an attack, i.e. the average period of time that it took to complete an attack on a single CAPTCHA of that particular scheme. The experiments were conducted on an Intel Core 2 Duo 3.33GHz PC.

As a measure of CAPTCHA effectiveness, Chellapilla et al. (2005a) suggest that automated bots should not be able to correctly solve a CAPTCHA at a success rate of higher than 0.01%. However, Bursztein et al. (2011b) state that this security goal is too ambitious and instead assert that a CAPTCHA scheme is broken when the attacker is able to reach a success rate of at least 1%. The results in Table 1 show that the results of our attacks are well above either of these benchmarks. In addition, the time that it takes to attack the CAPTCHAs is well within the duration that a normal human would take to solve a CAPTCHA.

In general, the success of our attacks stems from the fact that these 3D CAPTCHAs were designed based on the concept of perturbing a regular pattern in order to enable humans to perceive 3D characters amidst the pattern. While this may seem secure as it prevents OCR programs or automated bots from solving the CAPTCHA challenge directly, due to the noise created by the background pattern, in this paper we have shown that the perturbations made to a regular pattern can in fact be exploited. Our attacking techniques rely on the fact that key information that can be used to identify the character, and consequently be used to solve the CAPTCHA, can be extracted from these perturbations. Upon extracting this information, the resulting image can be fed into any good OCR program, which can then recognize the

characters with a high success rate. Some of the common 3D CAPTCHA features that were exploited in our approach where changes in line direction for patterns with linear lines, changes in pixel density which typically indicated the edges or corners of characters, as well as changes in cell size (for grid like patterns) or distances between lines (for patterns with parallel lines).

From the experimental results, it can be seen that our attack on Teabag 3D version 1.0.1 had the highest success rate. This was because in addition to the features mentioned above, our method could also identify the side and front character surfaces, making it easier to clean up the image during the post-processing stage before character recognition. Furthermore, the 3D character models in this CAPTCHA scheme were very rigid characters, as such it made it easier to identify characters with straight lines as these characters would uniformly change a number of connected cells in the grid pattern. This became more difficult in the updated version of this scheme, i.e. Teabag 3D version 1.2, as evident from the experimental results where the success rate dropped by around half. In this version, the wave like pattern distorted the grid, which in turn created more variation in the cell sizes and warped straight lines thus reducing the accuracy of our approach. Nevertheless, there is still a significant difference in the size and shape of the cells which form the characters compared to the background grid, and our method of attack still results a high success rate.

Overall, the success rate of attacking Super CAPTCHA was lower than the rest. This is because the lines that are used to build Super CAPTCHA are generated using Markov-chains. This gives rise to some randomness in the lines, which makes it more difficult to distinguish between pixels that belong to the parallel lines and those that belong to the characters. As such, using larger distortion parameters for the lines will certainly make the CAPTCHA more difficult to break. However, this also means potentially reducing the human usability of the resulting CAPTCHA.

It is widely recognized that the state of the art in traditional 2D CAPTCHA design suggests that crowding or overlapping characters is currently the most effective approach to deterring automated attacks (Ahmad et al., 2010). We believe that these principles also apply to the design of 3D CAPTCHAs, as it would make it more difficult to extract information about individual characters. Nevertheless, one has to bear in mind that crowding, overlapping or joining characters together also makes it more difficult for a human to recognize the characters in a 3D CAPTCHA.

5. Summary

The fundamental requirement of a practical CAPTCHA scheme necessitates that humans must be able to solve the CAPTCHA challenges with a high degree of success, while the likelihood that a computer program can correctly solve them must be very small. Unfortunately, many studies have shown that various traditional 2D text-based CAPTCHA schemes can be solved by automated computer programs at high success rates. As such, some CAPTCHA designers have proposed 3D CAPTCHA schemes to overcome this problem, as 3D CAPTCHAs are assumed to be secure because they cannot be solved by OCR programs directly.

However, we have shown approaches that can be used to attack 3D CAPTCHA schemes. The 3D CAPTCHA schemes examined in this research were in essence built from the concept that by perturbing a regular pattern, a human will be able to perceive the 3D text which is embedded in the pattern, while a computer will have difficulty in determining the text. By examining three different 3D CAPTCHA schemes, we have demonstrated techniques that can be used to identify and extract key information from such 3D CAPTCHAs. Essentially, the information that we wanted to identify and extract from the CAPTCHAs was the pixels that belong to the characters. This is because once the characters can be extracted from the pattern, the characters can simply be passed through any good OCR program which will then be able to recognize the text with a high degree of success.

5.1. Main methods

In general, the following methods were used to identify and extract key information from the respective 3D CAPTCHA schemes. Despite being used for different CAPTCHA schemes, it can be seen that the notions underlying some of these methods are relatively similar.

Super CAPTCHA

- Changes in line direction - by estimating the gradient of the parallel lines, we could identify pixels that did not conform to this gradient.
- Variation in the number of pixels between lines - since the lines were parallel, we could identify regions between consecutive lines that were greater than the average pixel distance.

3dcaptcha

- Variation in the number of pixels between lines - by calculating the average number of pixels between consecutive lines, we could identify irregular regions between lines which contained a greater number of pixels.
- Changes in line direction - the regular pattern consisted of straight lines, hence we could identify regions that belonged to the characters based on changes in the direction of the lines.
- Pixel densities - character boundaries could be determined based on regions which contained a high density of connected pixels.
- Variation in cell sizes - by dividing regions into grids, we could identify cells which had a higher pixel count compared to their neighboring cells.

Teabag 3D

- Variation in cell sizes - we could identify the side surfaces of characters based on cells which contained a higher number of pixels compared to their neighboring cells.
- Pixel densities - pixels belonging to the front surface of characters were determined based on regions which contained a high density of connected pixels.
- Segmentation - we could perform segmentation by trying to separate characters using a vertical line, a diagonal line which corresponded to the gradient of the grid, or by tracing the pixels along the side surfaces of the characters.

5.2. Limitations

The methods discussed above for extracting key information were all based on the notion of identifying variations amidst a regular pattern. It is anticipated that the success of our approaches can potentially be reduced by adding greater randomness to the patterns. For example, instead of using a pattern that consists of a collection of straight lines with similar gradients, if the text were to be embedded in a pattern which consisted of randomly

curving lines, this would significantly reduce the accuracy of some of our methods. In particular, changes in line direction as well as variation in the number of pixels between lines would not necessarily be due to an embedded character. As an example, it can be seen that the improved version of Teabag 3D, i.e. version 1.2, adopted a wave like pattern rather than a regular grid pattern. This meant that there was greater variation in the cell sizes between adjacent cells. While our attack was still able to break this version, the success rate of our attack dropped significantly.

However, one also has to consider the fact that adding greater randomness to the underlying pattern may adversely affect the human usability of the resulting 3D CAPTCHA scheme. This is because the human visual system may have difficulty perceiving text which is embedded in an irregular background pattern. This would undoubtedly make the resulting scheme frustrating to use as humans would not be able to solve it with a high degree of success.

6. Conclusion

This paper investigates the robustness of text-based 3D CAPTCHAs against automated attacks. By identifying key features in the construction of the 3D CAPTCHA scheme, it is possible to extract information about the CAPTCHA challenge. A number of methods for doing this have been described in this paper. Common methods include identifying variations in the background pattern, such as changes in line direction, distances between lines, pixel density, cell sizes, etc. This paper demonstrates that although the concept of text-based 3D CAPTCHA schemes may appear to be secure as OCR programs cannot be used to directly solve the 3D CAPTCHA, information can first be extracted from the CAPTCHA before passing it to an OCR program. The results presented in this paper show that by doing so, automated attacks can solve 3D CAPTCHAs with a high degree of success. While CAPTCHA developers constantly seek to produce new CAPTCHA alternatives, these results appear to negate the assumption that 3D CAPTCHAs are really more secure than their traditional 2D counterparts.

Acknowledgment

Willy Susilo would like to acknowledge the support of ARC Future Fellowship FT0991397.

References

ABBYY, 2013. ABBYY FineReader, http://finereader.abbyy.com, viewed 29 June 2013 .

Ahmad, A. S. E., Yan, J., Marshall, L., 2010. The robustness of a new CAPTCHA. In: Costa, M., Kirda, E. (Eds.), EUROSEC. ACM, pp. 36–41.

Ahmad, A. S. E., Yan, J., Ng, W.-Y., 2012. CAPTCHA design: Color, usability, and security. IEEE Internet Computing 16 (2), 44–51.

Baecher, P., Buscher, N., Fischlin, M., Milde, B., 2011. Breaking reCAPTCHA: A holistic approach via shape recognition. In: Camenisch, J., Fischer-Hubner, S., Murayama, Y., Portmann, A., Rieder, C. (Eds.), Future Challenges in Security and Privacy for Academia and Industry. Vol. 354 of IFIP Advances in Information and Communication Technology. pp. 56–67.

Baird, H. S., Coates, A. L., Fateman, R. J., 2003. PessimalPrint: a reverse turing test. IJDAR 5 (2-3), 158–163.

Bursztein, E., Beauxis, R., Paskov, H., Perito, D., Fabry, C., Mitchell, J. C., 2011a. The failure of noise-based non-continuous audio captchas. In: IEEE Symposium on Security and Privacy. IEEE Computer Society, pp. 19–31.

Bursztein, E., Martin, M., Mitchell, J. C., 2011b. Text-based CAPTCHA strengths and weaknesses. In: Chen, Y., Danezis, G., Shmatikov, V. (Eds.), ACM Conference on Computer and Communications Security. ACM, pp. 125–138.

Café Charlotte, 2006. Café Charlotte, http://www.cafe-charlotte.cz/en/fanclub, viewed 29 June 2013.

Cafe Milenium, 2009. Bar 21 Lounge, http://www.barlounge21.sk/kniha-navstev, viewed 29 June 2013.

Chellapilla, K., Larson, K., Simard, P. Y., Czerwinski, M., 2005a. Building segmentation based human-friendly human interaction proofs (HIPs). In: Baird, H. S., Lopresti, D. P. (Eds.), HIP. Vol. 3517 of Lecture Notes in Computer Science. Springer, pp. 1–26.

Chellapilla, K., Larson, K., Simard, P. Y., Czerwinski, M., 2005b. Designing human friendly human interaction proofs (HIPs). In: van der Veer, G. C., Gale, C. (Eds.), CHI. ACM, pp. 711–720.

Chellapilla, K., Simard, P. Y., 2004. Using machine learning to break visual human interaction proofs (HIPs). In: NIPS.

Chew, M., Baird, H. S., 2003. BaffleText: a human interactive proof. In: Kanungo, T., Smith, E. H. B., Hu, J., Kantor, P. B. (Eds.), DRR. Vol. 5010 of SPIE Proceedings. SPIE, pp. 305–316.

Cruz-Perez, C., Starostenko, O., Uceda-Ponga, F., Aquino, V. A., Reyes-Cabrera, L., 2012. Breaking reCAPTCHAs with unpredictable collapse: Heuristic character segmentation and recognition. In: Carrasco-Ochoa, J. A., Trinidad, J. F. M., Olvera-López, J. A., Boyer, K. L. (Eds.), MCPR. Vol. 7329 of Lecture Notes in Computer Science. Springer, pp. 155–165.

EnterMir, 2013. Enter Mir, http://entermir.com.ua/cs/reg.php, viewed 29 June 2013.

Gao, H., Wang, W., Qi, J., Wang, X., Liu, X., Yan, J., 2013. The robustness of hollow CAPTCHAs. In: Sadeghi, A.-R., Gligor, V. D., Yung, M. (Eds.), ACM Conference on Computer and Communications Security. ACM, pp. 1075–1086.

Goldsboro Web Development, 2013. Super CAPTCHA, http://goldsborowebdevelopment.com/product/super-captcha/, viewed 29 June 2013.

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

HyperMedia, 2012. xicht.cz, http://www.xicht.cz/index.php?stranka=registrace, viewed 29 June 2013.

Imsamai, M., Phimoltares, S., 2010. 3D CAPTCHA: A next generation of the CAPTCHA. In: Proceedings of the International Conference on Information Science and Applications (ICISA 2010), Seoul, South Korea, 21-23 April, 2010. IEEE Computer Society, pp. 1–8.

Kaplan, M. G., n.d. The 3D-CAPTCHA, http://spamfizzle.com/CAPTCHA.aspx, viewed 21 June 2010.

Kolupaev, A., Ogijenko, J., 2008. CAPTCHAs: Humans vs. bots. IEEE Security & Privacy 6 (1), 68–70.

Li, S., Shah, S. A. H., Khan, M. A. U., Khayam, S. A., Sadeghi, A.-R., Schmitz, R., 2010. Breaking e-banking CAPTCHAs. In: Gates, C., Franz, M., McDermott, J. P. (Eds.), ACSAC. ACM, pp. 171–180.

Mian, A. S., Bennamoun, M., Owens, R. A., 2006. Three-dimensional model-based object recognition and segmentation in cluttered scenes. IEEE Trans. Pattern Anal. Mach. Intell. 28 (10), 1584–1601.

Mitra, N. J., Chu, H.-K., Lee, T.-Y., Wolf, L., Yeshurun, H., Cohen-Or, D., 2009. Emerging images. ACM Trans. Graph. 28 (5).

Mori, G., Malik, J., 2003. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In: CVPR (1). IEEE Computer Society, pp. 134–144.

Moy, G., Jones, N., Harkless, C., Potter, R., 2004. Distortion estimation techniques in solving visual CAPTCHAs. In: CVPR (2). pp. 23–28.

Nakaguro, Y., Dailey, M. N., Marukatat, S., Makhanov, S. S., 2013. Defeating line-noise CAPTCHAs with multiple quadratic snakes. Computers & Security 37 (0), 91 – 110.

Nguyen, V. D., Chow, Y.-W., Susilo, W., 2011. Breaking a 3D-based CAPTCHA scheme. In: Kim, H. (Ed.), ICISC. Vol. 7259 of Lecture Notes in Computer Science. Springer, pp. 391–405.

Nguyen, V. D., Chow, Y.-W., Susilo, W., 2012a. Attacking animated CAPTCHAs via character extraction. In: Pieprzyk, J., Sadeghi, A.-R., Manulis, M. (Eds.), Cryptology and Network Security. Vol. 7712 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 98–113.

Nguyen, V. D., Chow, Y.-W., Susilo, W., 2012b. Breaking an animated CAPTCHA scheme. In: Bao, F., Samarati, P., Zhou, J. (Eds.), ACNS. Vol. 7341 of Lecture Notes in Computer Science. Springer, pp. 12–29.

OCR Research Team, 2006. TEABAG_3D, http://ocr-research.org.ua, viewed 29 June 2013.

Rediff.com India, 2013. Rediffmail, http://register.rediff.com/register/register.php, viewed 29 June 2013.

Ross, S. A., Halderman, J. A., Finkelstein, A., 2010. Sketcha: a CAPTCHA based on line drawings of 3D models. In: Rappa, M., Jones, P., Freire, J., Chakrabarti, S. (Eds.), WWW. ACM, pp. 821–830.

SACHYonline.cz, 2010. SACHY online.cz, http://www.sachyonline.cz/?action=register, viewed 29 June 2013.

Tam, J., Simsa, J., Hyde, S., von Ahn, L., 2008. Breaking audio CAPTCHAs. In: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (Eds.), NIPS. MIT Press, pp. 1625–1632.

von Ahn, L., Blum, M., Hopper, N. J., Langford, J., 2003. CAPTCHA: Using hard AI problems for security. In: Biham, E. (Ed.), EUROCRYPT. Vol. 2656 of Lecture Notes in Computer Science. Springer, pp. 294–311.

WordPress.org, 2013. Super CAPTCHA security suite - one and only 3D CAPTCHA, http://wordpress.org/extend/plugins/super-capcha/, viewed 29 June 2013.

Xu, Y., Reynaga, G., Chiasson, S., Frahm, J.-M., Monrose, F., Van Oorschot, P., 2012. Security and usability challenges of moving-object CAPTCHAs: decoding codewords in motion. In: Proceedings of the 21st USENIX conference on Security symposium. Security'12. USENIX Association, Berkeley, CA, USA, pp. 4–4.

Yan, J., Ahmad, A. S. E., 2007. Breaking visual CAPTCHAs with naive pattern recognition algorithms. In: ACSAC. IEEE Computer Society, pp. 279–291.

Yan, J., Ahmad, A. S. E., 2008. A low-cost attack on a Microsoft CAPTCHA. In: Ning, P., Syverson, P. F., Jha, S. (Eds.), ACM Conference on Computer and Communications Security. ACM, pp. 543–554.

Yan, J., Ahmad, A. S. E., 2009. CAPTCHA security: A case study. IEEE Security & Privacy 7 (4), 22–28.

Yan, J., Ahmad, A. S. E., 2011. CAPTCHA robustness: A security engineering perspective. Computer 44 (2), 54–60.

YUNiTi, 2013. YUNiTi - do something good, http://www.yuniti.com/register.php, viewed 29 June 2013.

Zhu, B. B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., Yi, M., Cai, K., 2010. Attacks and design of image recognition CAPTCHAs. In: Al-Shaer, E., Keromytis, A. D., Shmatikov, V. (Eds.), ACM Conference on Computer and Communications Security. ACM, pp. 187–200.

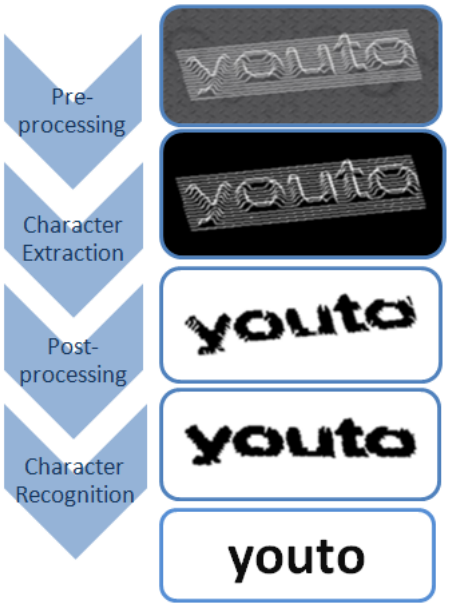
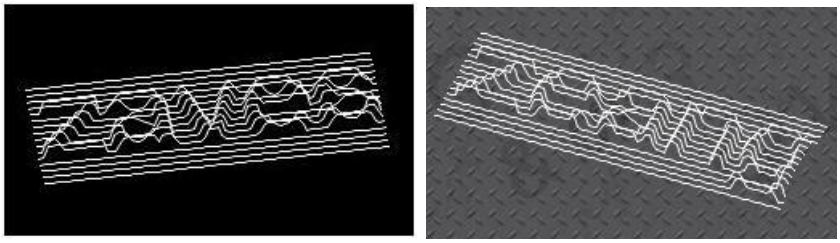
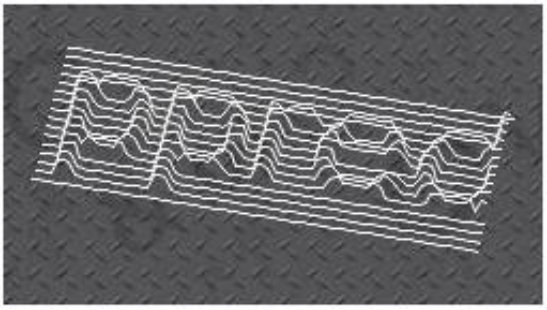


Figure 1: Overview of the stages.

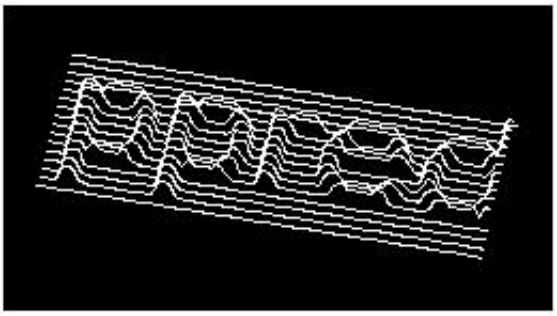


(a) Without background image (b) With background image

Figure 2: Examples of Super CAPTCHA



(a) Original image



(b) After background removal and binarization

Figure 3: Pre-processing stage in attacking Super CAPTCHA

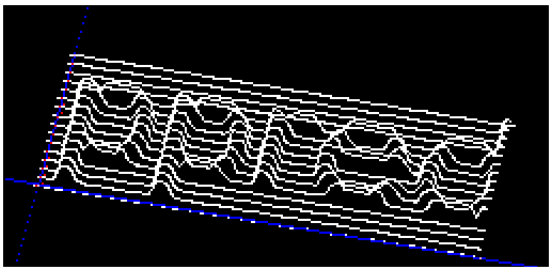


Figure 4: The approximate gradient of the white lines (shown in blue).

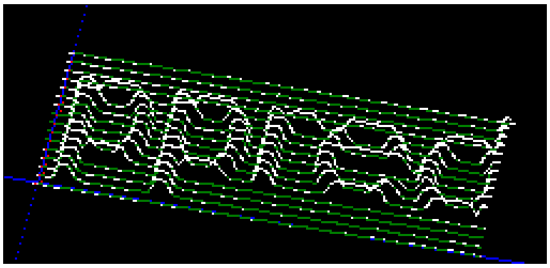


Figure 5: Parallel lines (illustrated in green) used to overwrite the white lines.

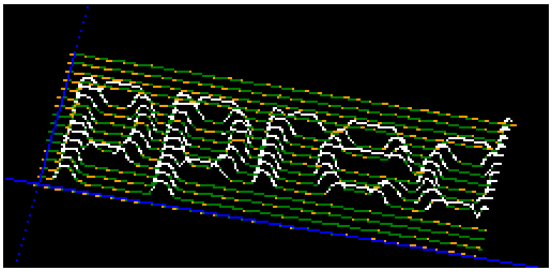


Figure 6: Connected pixels (illustrated in orange) in the direction of the parallel lines.

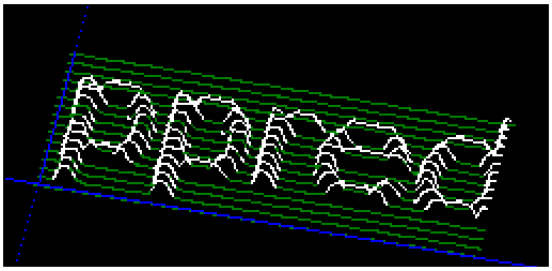


Figure 7: Separation between the pixels related to the characters (in white) and the pixels that belong to the lines (in green).

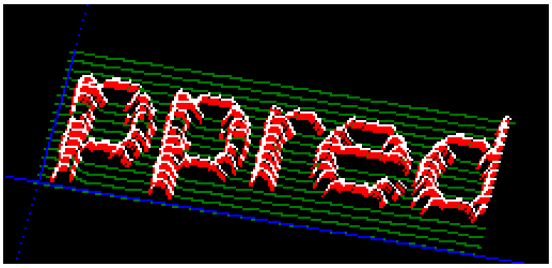


Figure 8: Pixels (in red) that were determined to belong to the characters.

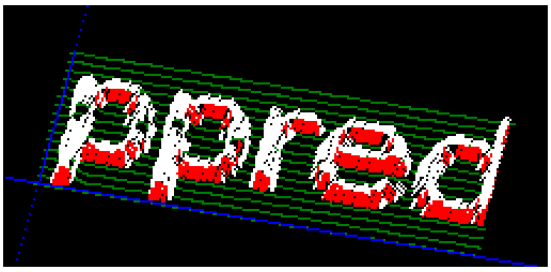


Figure 9: Additional pixels (in red) in large empty regions below the character pixels.

(a) Pixels determined to belong to the characters.

(b) Final post-processed image.

Figure 10: Results of the post-processing stage.

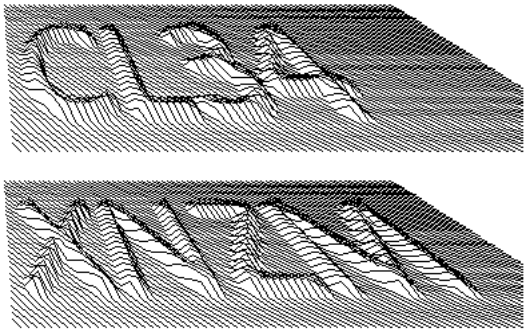


Figure 11: Examples of 3dcaptcha.

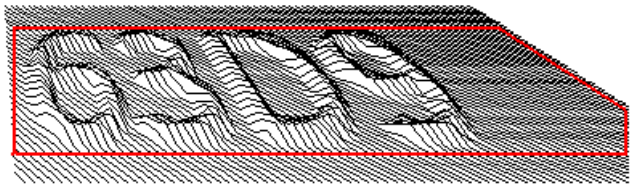
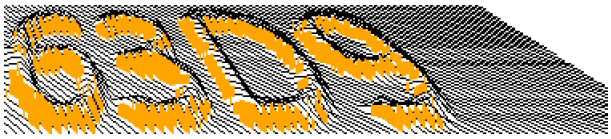
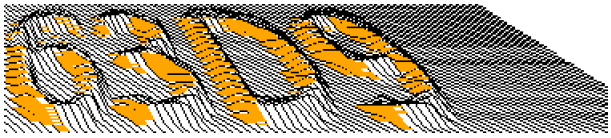


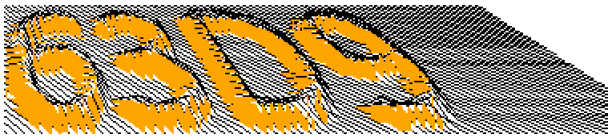
Figure 12: Processing area (inner red line).



(a) Result of a vertical scan.



(b) Result of a horizontal scan.



(c) Combined results

Figure 13: Extracting key features based on the distances between lines.

Figure 14: Extracting key features based on changes in line direction.

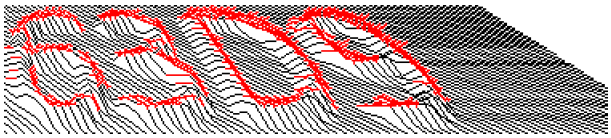


Figure 15: Character boundaries determined based on the high density of connected pixels.

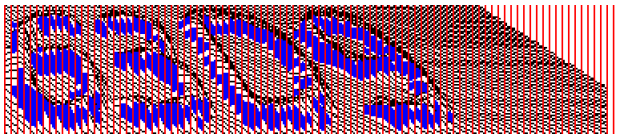


Figure 16: Cells containing more pixels than their neighbors.

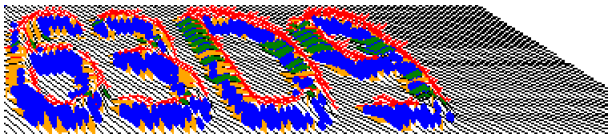
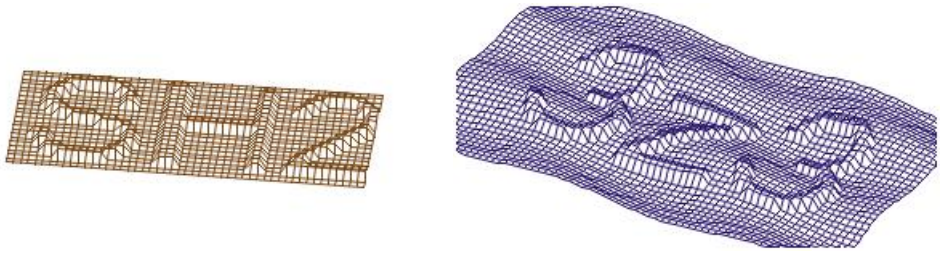


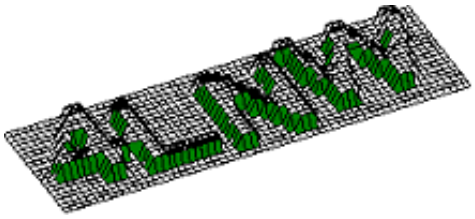
Figure 17: Combining all identified key features.

Figure 18: Post-processed image.

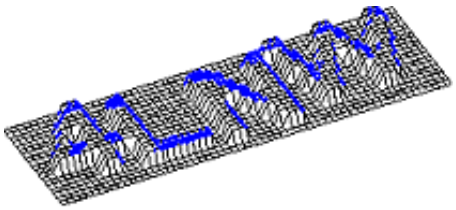


(a) Teabag 3D version 1.0.1 (b) Teabag 3D version 1.2

Figure 19: Examples of Teabag 3D CAPTCHA



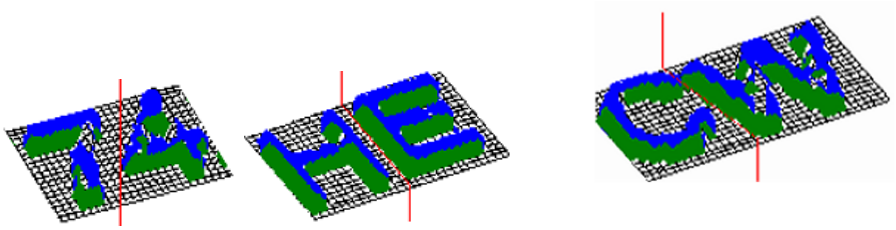
(a) Side surface identification.



(b) Front surface identification from clusters of black pixels.

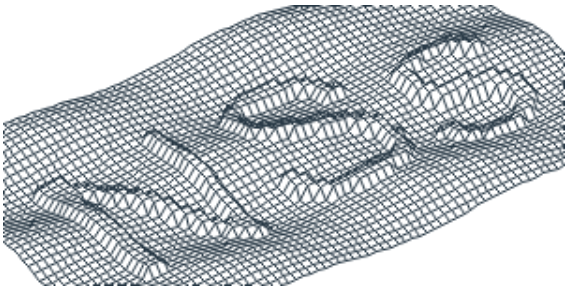
(c) Final results showing the front and side surfaces of the characters.

Figure 20: Extracting key features for a Teabag 3D version 1.0.1 challenge.



(a) Vertical segmentation. (b) Diagonal segmentation. (c) Side surface segmentation.

Figure 21: Different segmentation approaches.



(a) Original challenge.

(b) Final results showing the front and side surfaces of the characters.

Figure 22: Extracting key features for a Teabag 3D version 1.2 challenge.